VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

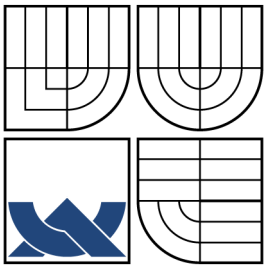
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

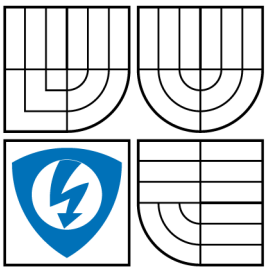
KOMPRESE XML DAT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MICHAL JELEN
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

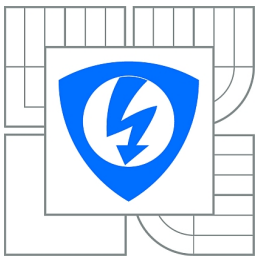
KOMPRESE XML DAT
XML DATA COMPRESSION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MICHAL JELEN
AUTHOR

VEDOUCÍ PRÁCE — Ing. DOMINIK KOVÁČ
SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Michal Jelen	***ID:***	125469
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Komprese XML dat

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku komprese XML dat. Následně zpracujte a zdokumentujte dosavadní mechanizmy/nástroje pro kompresi XML dat. V další části práce se seznamte s formátem AIXM, který je určený pro výměnu aeronautických informaci ve formátu XML. Vytvořte nástroj, který otevře XML dokument obsahující aeronautické informace ve formátu AIXM a provede analýzu a rozklad těchto dat do vybraných kategorií. Nástroj umožní vybrat jednotlivé objekty XML souboru a provede kompresi výsledného souboru pomocí uživatelsky zvoleného kompresního mechanizmu.

DOPORUČENÁ LITERATURA:

[1] BENZ, B., DURANT, J.: XML Programming Bible. New York: Wiley Publishing, Inc., 2003, ISBN: 0-7645-38292.
[2] RAY, E.: Learning XML, Second Edition. Sebastopol: O'Reilly Media, 2003, ISBN: 978-0596004200.

Termín zadání:	10.2.2014	***Termín odevzdání:***	28.5.2014

Vedoucí práce: Ing. Dominik Kováč
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se zabývá problematikou komprese XML dat. Popisuje charakteristiku a důležité pojmy XML jazyka, stejně tak AIXM, jakožto formát XML. Dokumentuje současné kompresní mechanizmy XML a rozděluje je do patřičných skupin. Dále se realizuje testování komprese XML za použití zmíněných metod a vyhodnocení výsledků. S přihlédnutím na zpracované údaje je vytvořena a prezentována aplikace, která za pomoci analýzy dat jednotlivých elementů umožňuje ztrátovou i bezeztrátovou kompresi AIXM. To vše s uživatelskými možnostmi výběru AIXM objektů a způsobu výsledné komprese.

KLÍČOVÁ SLOVA

XML, AIXM, komprese, dekomprese, XMILL, LZMA, parsování, NOTAM

ABSTRACT

This master's thesis is dealing with problems of XML data compression. It describes a characterization and important concepts of XML language as well as AIXM as XML data format. It documents contemporary compression methods of XML and divides them into appropriate groups. Thereinafter a testing of XML data compression using mentioned methods and evaluation of results is implemented. With a consideration of processed data there is created and presented an application which enables lossy and lossless compression of AIXM with the help of particular elements data analysis. All of it with user selection options of AIXM objects and way of resultant compression.

KEYWORDS

XML, AIXM, compression, decompression, XMILL, LZMA, parsing, NOTAM

JELEN, Michal *Komprese XML dat*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 83 s. Vedoucí práce byl Ing. Dominik Kováč

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Komprese XML dat“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych tímto chtěl poděkovat vedoucímu diplomové práce panu Ing. Dominikovi Kováčovi za odborné vedení, přínosné konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Komprese, neboli zpracování dat za účelem snížení jejich objemu, je v informatice, a nejen v ní, důležitým pojmem. Využívá se především k docílení menšího datového toku nebo zmenšení potřeby zdrojů při práci s danými daty. Využití komprese jsou nezměrná. Přes upotřebení při archivaci nebo přenosu dat přes síť, až například po využití v telekomunikacích.

Výhodnost použití komprese lze pozorovat na takřka všech druzích dat. Tato práce se soustřeďuje na kompresi jazyka XML, kterému se věnuje úvodní kapitola. Ta otevírá teoretickou část práce a popisuje jeho charakteristiku a důležité pojmy, které jsou nezbytné k dalšímu porozumění. Věnuje se jak syntaxi nebo kódování, tak například i parsování tohoto značkovacího jazyka. V závěru pak popisuje několik výhod a nevýhod XML.

Další kapitola upřesňuje zaměření práce a popisuje tak model AIXM, jeho charakteristiku a konvence modelování. Dále seznamuje s typickými pojmy modelu jako jsou například model dočasnosti nebo NOTAM. Model AIXM je důležitý při využití v letectví, kde je trend předávat informace nejenom v papírové podobě, ale stále více i v té elektronické. Při přenosu dat v této podobě je velmi vhodné omezit přenášená data pouze na aktuální informace, jelikož kapacita přenosových kanálů je omezena. Proto je prospěšné využít i kompresi AIXM, které je věnována hlavní praktická část práce.

Následující kapitola se věnuje již samotné kompresi XML dat. I když se zabývá i obecnou charakteristikou komprese, stěžejním této kapitoly jsou především druhy dělení, kterým se věnuje, dále je dělí, a popisuje jednotlivé kompresní mechanizmy, které řadí do patřičných skupin. Důležitým bodem je vnitřní struktura XML dokumentu a její vliv na celkovou kompresi, dalším je pak otázka tzv. dotazování nad komprimovanými daty, resp. přístupu k nim.

Čtvrtá kapitola se zaměřuje na jednu ze součástí praktické části práce. Tato se zabývá testováním komprese různých XML souborů při využití různých kompresních technik. Zprvu je popsána metodika testování a jednotlivé soubory. V závěru se pak nacházejí výsledky celého testování s popisem veškerých jeho testovaných variant. Na tuto kapitolu navazují i dvě přílohy, které rozšiřují dosažené výsledky o zajímavé informace v podobě tabulek a jejich grafického vyhodnocení.

Poslední kapitola se zabývá hlavní praktickou částí, a to aplikací vytvořenou za účelem ztrátové a bezeztrátové komprese AIXM dat. Je zmíněn princip fungování jednotlivých typů kompresí v programu a použité kompresní mechanizmy. Dále jsou podrobněji popsány jednotlivé programové části aplikace, jejichž možnosti jsou vyobrazeny také na obrázcích. Nemohou chybět ani ukázky důležitých částí kódu. Na ty je odkazováno do třetí přílohy práce.

1 XML

XML (eXtensible Markup Language) je obecný a otevřený značkovací jazyk, standardizovaný konsorciem W3C (World Wide Web Consortium) [34], k reprezentování datové struktury. Stejně jako HTML (HyperText Markup Language) vychází ze staršího a složitějšího metajazyka[1] SGML (Standard Generalized Markup Language), resp. je jeho podmnožinou (profilem). XML je používán pro popis dokumentů a dat ve standardizované, textově orientované formě. Proto může být přeposílán přes různé tradiční přenosové protokoly.

1.1 Charakteristika

Cílů při návrhu XML v roce 1998 bylo hned několik, dají se ale shrnout do jednoduchosti, obecnosti a použitelnosti [34]. Mimo cíle, které by měl XML splňovat, byly také vytyčeny jisté pilíře, na kterých se má stavět – rozšiřitelnost, struktura a validita.

Původní validační standard XML souborů byl DTD (Data Type Definition), který byl nahrazen jinou alternativou, také založenou na tzv. *XML schématech*, XSD (XML Schema Definition).

XML se soustřeďuje na věcný obsah, nezabývá se vzhledem. Ten lze upravovat stylovými jazyky, kterých existuje celá řada. Mezi nejznámější patří asi kaskádové styly CSS (Cascading Style Sheets). Ty se většinou využívají pouze v případech, kdy chceme jednoduché formátování. Komplexnější možností je rodina jazyků XSL (eXtensible Stylesheet Language). Ta umožňuje dokument již více upravovat, transformovat, vybírat jednotlivé části nebo generovat obsahy a rejstříky pomocí XSLT (XSL Transformations), XSL-FO (XSL - Formatting Objects) a XPath (XML Path Language) [35]. Příklad formátování XML pomocí XSL je na obrázku 1.1.

Použití XML je nepřeberné množství. Od serializace[2] dat, použití při tvoření metadat[3] a konfiguračních souborů, přes výměnu dat mezi aplikacemi v prostředí internetu, B2B[4] aplikace, inteligentní webové stránky, elektronické publikovaní prostřednictvím univerzálního datového formátu až například po převod XML do HTML s použitím XSLT.

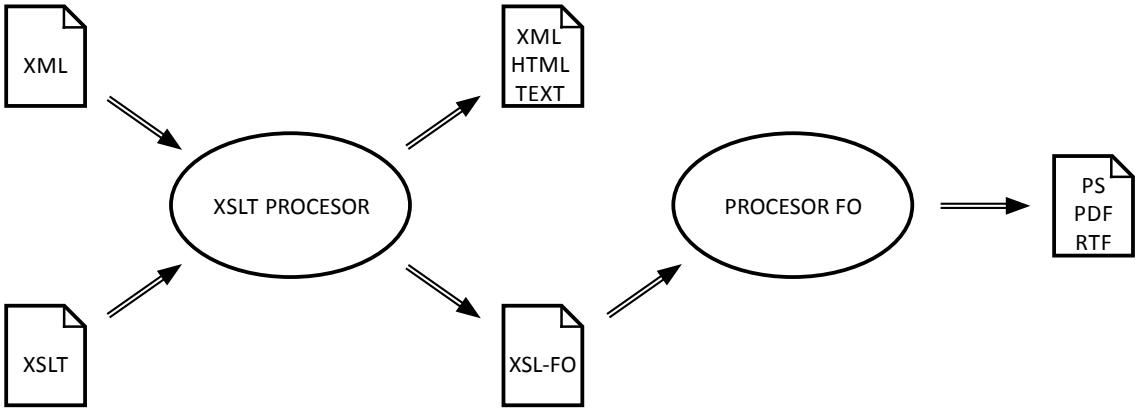
Je vhodný zejména pro dokumenty, které obsahují strukturované informace. Téměř v každém dokumentu se dá vypozorovat jistá struktura dat.

[1] Nadřazený značkovací jazyk, v rámci něhož je možné vytvářet vlastní jazyky.

[2] Převedení datové struktury uložené ve vnitřní paměti počítače na posloupnost bitů.

[3] Strukturované informace o datech.

[4] Business-to-business (obchodní vztahy mezi obchodními společnostmi).



Obr. 1.1: Princip formátování XML dokumentu pomocí XSLT a XSL-FO [18]

1.2 Syntaktická analýza

Je třeba mít na paměti, že se na dodržení syntaxe XML dat, i přes svou rozšiřitelnost, klade velký důraz (ve srovnání s jinými jazyky).

XML dokument je tvořen posloupností Unicode[5] znaků. Ve své podstatě je to textový dokument, ve kterém se rozlišují dva základní prvky – elementy (značky) a obsah (text). Elementy mohou být ještě dále popsány vnořenými atributy. Takto formátovaný dokument spolu s doporučeními W3C můžeme považovat za tzv. *well-formed XML*, tedy správně strukturovaný způsob zápisu [17]. S takovýmto zápisem si pak poradí všechny aplikace podporující XML.

Syntaxe XML obsahuje mnoho prvků a i jejich speciální případy. Níže jsou popsány ty nejzákladnější.

1.2.1 Element

Základní prvek, ze kterého se skládá každý XML dokument. Elementy se vyznačují tzv. *tagy*[6]. Většinou má každý element svůj počáteční a ukončovací tag, jejichž názvy se zapisují mezi znaky `<` a `>`, přičemž ukončovací tag má před svým názvem ještě navíc znak `/`, aby byl jednoznačně rozpoznatelný. Jejich využití je velmi důležité při analýze a parsování XML dat (kap. 1.3). Nedodržení správné syntaxe může mít na strukturu dokumentu a obsažená data fatální důsledky. Data pak nemusí být zpětně rozpoznatelná. Příklad správně zapsaného elementu může vypadat následovně [3].

```
<element> </element>
```

[5]Standard pro konzistentní kódování a reprezentaci znaků všech existujících abeced.
[6]Značka definující jistým způsobem část kódu.

Názvy elementů mohou obsahovat písmena, číslice, pomlčky, podtržítka, tečky a dvojtečky. To vše při použití tzv. *namespaces*. Více v kapitole 1.2.4.

Naopak nemohou obsahovat mezery. Dále pak nesmí začínat číslicí, speciálním znakem mimo abecedu ani příkazem `xml` [3]. I přes povolené názvy je třeba uvažovat nad budoucím zpracováním dokumentu a následnou kompatibilitou mezi ostatními systémy. Nedoporučují se tedy pomlčky, tečky, apod.

Speciální případ elementu je tzv. *prázdný element*. Ukázka syntaxe:

```
<element/>
```

Nemá žádný obsah a přidává se samostatně, bez žádných dalších tagů. Obvykle slouží k popsání předdefinované datové struktury.

1.2.2 Atribut

Atributy se obvykle používají k upřesnění významu elementu [17] a jsou obsaženy v počátečním tagu elementu za jeho názvem.

```
<element atribut="hodnota"> </element>
```

Element může obsahovat hned několik atributů oddělených mezerou. Jejich hodnota se píše do uvozovek nebo apostrofů za názvem atributu a znakem `=`. Hodnota atributu může také obsahovat uvozovky nebo apostrofy, vždy ale s pravidlem, že jeden způsob zápisu je použit k vymezení hodnoty atributu a ten druhý následně k použití uvnitř dané hodnoty [3]. Nikdy se nesmí tato označení syntakticky křížit.

1.2.3 Text

Textový obsah se zapisuje mezi počáteční a ukončovací tag elementů a obvykle reprezentuje data, která jsou asociována s daným elementem.

```
<element> textový obsah </element>
```

Text není omezen stejnými syntaktickými pravidly jako například elementy a jejich atributy, tudíž lze mezi tagy elementu zapsat jakýkoliv obsah [3]. Zde je ale potřeba mít na paměti, že se určité znaky (`<`, `>`, `&`, `;`, apod.) používají k oddělení tagů od okolního textu. Pro jejich zápis musíme využít tzv. *znakové entity*, což jsou v zásadě kombinace jiných znaků, které jednoznačně reprezentují požadovaný znak či symbol. Tyto kombinace již nejsou XML parserem (kap. 1.3.3) chápány jako součást tagů, nýbrž jako uživatelský obsah.

Stejných principů lze využít i v případě, kdy zvolené kódování (kap. 1.2.6) nepodporuje námi požadovaný znak. Díky znakovým entitám můžeme tedy využívat jiných znakových sad.

1.2.4 Namespaces

XML namespaces poskytují metodu zabraňující vzniku duplicitních názvů ve jménech elementů [37] a následným konfliktům. Je to nepovinná součást XML dokumentů.

V mnoha případech se stává, že jména prvků, která si tvůrce XML dokumentu zvolil v praxi kolidují s jinými XML dokumenty, v nichž se nachází stejné názvy. Následně nelze adekvátně parsovat soubor, aniž bychom tyto shody nějak neupravili. K tomuto účelu slouží namespaces, která si lze představit jako obory názvů nebo jmenné prostory, které upravují dokumenty do akceptovatelného tvaru. K tomu se většinou využívají tzv. *xmlns atributy* ve spojení s prefixy[7] [3] jejichž názvy se odvíjí od toho, co je uvedeno za příkazem `xmlns:`. Na příkladu lze vidět obojí.

```
<prefix:element xmlns:prefix="http://www.web.cz"> </element>
```

Při shodě lze tedy měnit prefix a hodnotu xmlns atributu. V takovém případě již budou veškeré opakující se elementy dokumentu jednoznačně identifikovatelné. Je dobrým zvykem používat hodnotu atributu jako URL (Uniform Resource Locator) ukazatel odkazující na web s uloženými namespaces informacemi. Takto se nemusí kontrolovat změny v jednotlivých dokumentech, postačí změna pouze na webu.

1.2.5 Bílé znaky

Dokumenty jsou často psány s jistou strukturou, která je pro člověka lépe čitelná. Do ní obvykle patří mezery, tabulátory, prázdné řádky, apod. Souhrnně se těmto znakům říká bílé znaky neboli tzv. *whitespaces*.

To, co autor textu třeba ani neřeší, musí parser nějakým způsobem deklarovat a ošetřit, aby za každých okolností věděl, co má v daný moment udělat s následným kusem kódu. I přestože si mnoho parserů dokáže s touto situací poradit, je vhodné využít deklarovaného namespace atributu (kap. 1.2.4) `xml:space` s hodnotou `preserve`, který zajistí zachování bílých znaků napříč dokumentem [24].

```
<element xml:space="preserve"> </element>
```

Bílé znaky mohou mít v dokumentu více stupňů důležitosti. Jednou mohou být vypuštěny nebo nahrazeny, jindy jsou zase plnohodnotným obsahem. Je třeba dodat, že různé parsery se chovají rozdílně a mají různé možnosti, jak pracují nejen s bílými znaky. Nepřihlédnutí k bílým znakům může mít za následek i znehodnocení původního dokumentu z pohledu koncové aplikace, která s XML pracuje.

Pokud dokument používá DTD, musí být parametr `xml:space` deklarován [17].

[7] Předpona měnící význam slova.

1.2.6 Deklarace a kódování

Většina XML dokumentů začíná tagem `<?xml ?>`, kde do volného místa před poslední otazník se vkládají udaje o verzi XML, použitém kódování, apod. To vše pak tvoří deklaraci dokumentu, která je nepovinná dle specifikací W3C XML 1.0, ale přesto slouží jako výhodná identifikace dat [3].

```
<?xml version="1.0" encoding="windows-1250" standalone="yes"?>
```

Na příkladu lze vidět vyplnění tří základních parametrů. Verze XML, kódování dokumentu a parametr (standalone) informující parsery, zda se nachází nějaká další deklarace i mimo daný XML soubor (předvolená hodnota je `"no"`). Při uvedení nějakých parametrů v deklaraci se automaticky stává parametr `version` povinným [24]. V dnešní době se nejvíce využívá verze 1.0, i přes existenci verze 1.1, která obsahuje několik obměn (např. rozšíření Unicode standardů nebo změny v ukončování řádků, které se projeví hlavně při parsování) [3].

Posledním uvedeným parametrem je `encoding` – kódování. Ten specifikuje typ kódování použitý v daném dokumentu. V případě, že není definovaný, parser bere v úvahu přednastavený typ `UTF-8`[8] (UCS Transformation Format), který je ideální pro anglicky psaný obsah dokumentu, případně `UTF-16`, který se ale musí specificky nastavit tzv. *příkazovou značkou bajtu xFEFF* na začátku dokumentu [24].

Nastavení kódování je jedna z nejdůležitějších věcí. Protože XML dokument nemusí být psaný pouze ASCII (American Standard Code for Information Interchange) znaky, je třeba kódování specifikovat nebo uložit soubor jako Unicode [37].

V případě, že je jeden XML dokument fyzicky uložen v několika souborech, dílčí soubory se načítají jako tzv. *externí entity*. Každá externí entita může mít v deklaraci nadefinované různé kódování [17].

1.3 Parsování

Operace parsování ve své nejobecnější podobě odpovídá syntaktickému analyzování dokumentu a následnému převodu XML dat do podoby, která bude čitelná pro ostatní aplikace, které chtějí s daty zacházet. Parsování provádí tzv. *parsery* (kap. 1.3.3).

Je-li nutné jistou část kódu z parsování z jakéhokoliv důvodu vyloučit, lze využít například komentářů nebo tzv. *sekce CDATA* (Character Data).

Komentáře se ohraničují pomocí znaků `<!--` a `-->`. Mezi ně se zapisuje text, který má zůstat skrytý.

[8] Způsob kódování řetězců znaků Unicode/UCS (Universal Character Set) do sekvencí bajtů.

```
<!-- komentář -->
```

Zakomentování se nesmí aplikovat do ostatního značkování, nelze tedy např. komentovat atributy [17].

Sekce CDATA slouží hlavně ke vkládání větších částí kódu obsahujícího množství speciálních znaků, které je třeba zachovat. Bez použití této sekce by bylo nutné psát znakové entity (kap. 1.2.3), což by v některých případech byla jen zbytečná komplikace.

```
<![CDATA[text]]>
```

Bez parsování těchto oddílů tedy nedochází k chybám způsobeným jejich obsahem a zároveň plní i svou specifickou funkci.

Oba z výše uvedených příkladů nesmí uvnitř svého tagu obsahovat sekvenci znaků, která by tvořila vždy danou ukončovací značku.

1.3.1 DTD a XSD

Tyto dva standardy byly již zmíněny v úvodu (kap. 1.1). Lze na ně nahlížet jako na schémata definující soustavu specifikací a pravidel, která určují jak má vypadat XML dokument, aby byl validní.

DTD:
Jedná se o starší jazyk, který definuje množinu použitelných tagů a atributů, případně jejich možné umístění v XML dokumentu, resp. vzájemné uspořádání a vnoření elementů. DTD může také obsahovat definice entit nebo implicitní hodnoty některých atributů.

Nevýhodou je, že samotný DTD se nezapisuje pomocí jazyka XML [29].

XSD:
XSD, někdy také XML schéma, zvládá vše, co umí i DTD a přidává k tomu navíc další součásti. Například definuje typovost dat, zvládá namespaces (kap. 1.2.4) nebo říká, zda může být element prázdný či nikoliv. Dále pak umožňuje pracovat s daty uloženými v databázi, konvertovat data mezi různými datovými typy a další.

Soubory definující konkrétní schéma se zapisují v XML [30].

1.3.2 DOM a SAX

Tato kapitola je věnována dvěma platformě nezávislým programovým rozhraním, která přebírají XML data od parseru a řeší, jak se budou dokumenty zpracovávat. Na tuto vrstvu zpracování XML dat lze pohlížet jako na tzv. *API* (Application

Programming Interface). DOM (Document Object Model) je standard W3C, kdežto SAX (Simple API for XML) byl vyvinut členy XML-DEV[9] (XML-Development). I přestože není rozhraní SAX oficiálně doporučováno W3C, je v dnešní době již považováno také za standard.

DOM:
Původně byl DOM vytvořen zejména proto, aby prohlížeče podporující XML používaly stejný objektový model pro přístup k dokumentu ze skriptových jazyků[10] [17]. Je tedy, mimo jiné, i přínosem v oblasti kompatibility webových prohlížečů.

Rozhraní DOM je postaveno na principu, kdy je XML dokument reprezentován jako stromová hierarchická struktura. Každému elementu pak odpovídá jeden uzel stromu. Jednotlivé uzly samozřejmě mohou obsahovat i komentáře, instrukce pro zpracování, atd. DOM načte všechna XML data do paměti a umožňuje tak procházet celou stromovou strukturu za účelem mazání, přidávání a modifikování jednotlivých uzlů dle potřeby. To je vhodné pro aplikace, které s dokumentem provádí náročnější operace.

DOM má výhodu v tom, že se s ním lehce pracuje. Nevýhodou pak je, že objekty v paměti zabírají až třikrát více místa než původní XML dokument [19]. Vytvoření takovéto reprezentace trvá také určitý čas.

SAX:
Toto rozhraní je založeno na řízení pomocí událostí. Pomocí SAX se vytváří vazba mezi kódem a událostmi, které generuje parser. XML dokument se postupně čte a provádí se požadované operace jako volání množství funkcí, kterým se předávají potřebné parametry. Funkce se volají při nalezení počátečního tagu, koncového tagu (kap. 1.2.1), souvislého textu uvnitř nich, komentáře a mnoha dalších prvků dokumentu.

Výhoda událostmi řízeného přístupu je v jeho rychlosti a malé spotřebě paměti [19]. Oproti tomu ale SAX nemá tak široké spektrum možností, které může s daty provádět a dovoluje pouze jejich čtení.

1.3.3 Parsery

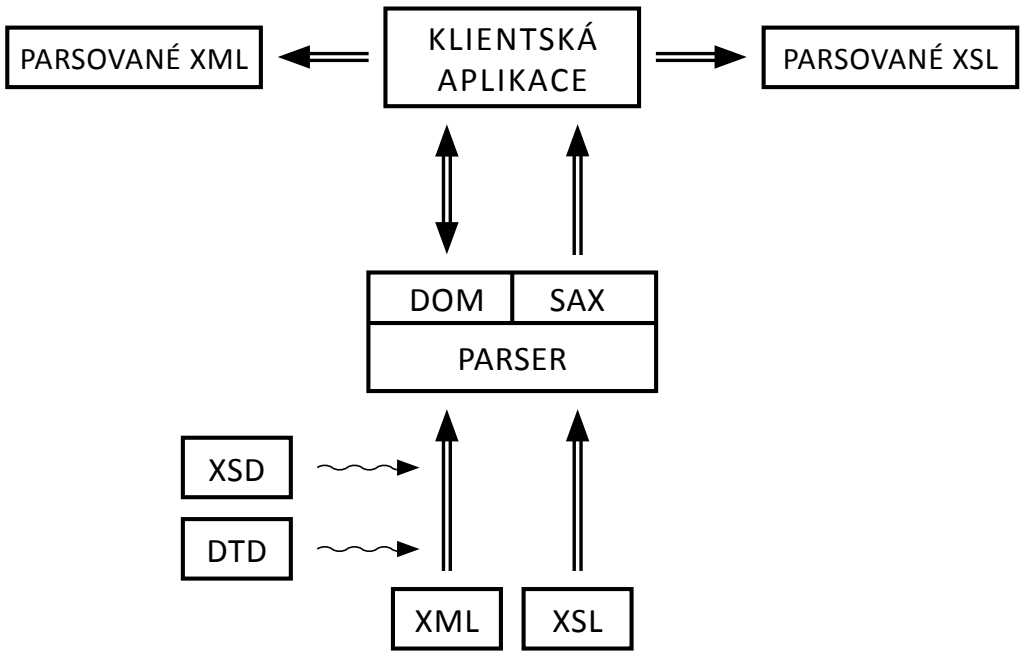
Ve své podstatě jsou parsery komplexní nástroje, které, mimo jiné, komunikují i s hardwarem[11] počítače [13]. Lze si je představit jako programové knihovny, pomocí nichž se kontroluje syntaktická správnost dokumentu a zároveň se jimi i dokument programově převádí, aby se dále mohl předat jiným aplikacím k dalšímu zpracování.

[9] Společenství vývojářů komunikujících prostřednictvím mailů.
[10] JavaScript, C#, apod.
[11] Fyzicky existující technické vybavení počítače.

To vše s přihlédnutím ke standardům popsaným výše. Bez parseru by koncová aplikace neměla všechny důležité informace k tomu, aby dokázala XML soubor správně přečíst a například i zjistit, zda se nesnaží zpracovat kód s chybami [17].

Parser muže mít mnoho podob. Samostatný program, integrální součást prohlížeče nebo editoru, knihovna nebo specifická třída pro vyšší programovací jazyk, atd.

Výše zmíněné pojmy principiálně shrnuje obrázek 1.2.



Obr. 1.2: Princip parsování XML

Parsery zpravidla spadají do dvou kategorií:

Validující parsery:
Provádí hlubší kontrolu XML dat, porovnávají řadu specifických pravidel definovaných v DTD nebo XSD, dále pak také kontrolují datové typy nebo tvoří rozhodnutí o přednastavených hodnotách [13]. Kontrola well-formed XML (kap. 1.2) je povinná.

Nevalidující parsery:
Oproti validujícím provádí jen rychlou kontrolu ustanovené syntaxe XML a struktury kódu, resp. zda je dokument well-formed XML. Při této kontrole splnění základních předpokladů je nezajímají datové typy, DTD nebo XSD, ani unikátní instrukce [13].

V přehledu dále je uveden bodový soupis názvů nejznámějších parserů dnešní doby. Jelikož jsou stále vyvíjeny a mění se jejich specifikace, není u nich uveden žádný popis. Podrobnější informace lze nalézt na webových stránkách jednotlivých parserů.

Expat	
HaXml	
JAXP	(Java API for XML Processing)
Libxml2	
LlamaXML	
MSXML	(Microsoft XML Core Services)
pugixml	
RapidXml	
SimpleXML	
StAX	(Streaming API for XML)
TinyXML-2	
VTD-XML	(Virtual Token Descriptor for XML)
Xerces	

1.4 Výhody a nevýhody

Výhody:
Mezi jednu z nejdůležitějších výhod patří fakt, že pomocí XML lze psát svůj vlastní značkovací jazyk. Vývojář není limitován žádnými konkrétními elementy, tagy nebo hodnotami atributů. Navíc je vytváření uživatelských značkovacích jazyků pro různé účely a různé typy dat snadné. Kdokoliv si tedy může vymyslet vlastní tagy a pravidla, které mohou velice dobře definovat přesnou strukturu každého XML dokumentu podle aktuální potřeby [1]. U vhodně napsaného kódu taky nemusí mít nezasvěcená osoba hlubší předchozí znalost jazyka.

Další důležitou vlastností je přenositelnost XML a z toho plynoucí i kompatibilita. S well-formed dokumentem splňujícím doporučení W3C tedy není žádný problém v komunikaci s ostatními institucemi nebo aplikacemi, které podporují stejná specifika přenášených XML dat.

Obvykle jsou dokumenty i snadno zpracovatelné parsery (kap. 1.3.3), stejně jako dalšími nástroji uzpůsobenými k práci s XML.

Možnost naprogramovat jazyk inteligentněji také dopomáhá vyhledávačům, které mnohdy samy o sobě inteligentnější pohled na zpracovávaná data postrádají [28].

V neposlední řadě je to i datový formát se silnou podporou Unicode.

Nevýhody:
Je třeba přiznat, že i přes četné výhody není XML zrovna nejkompaktnější. Díky své výřečnosti jsou reprezentovaná data obvykle mnohem větší než jejich originální forma. XML neřeší záležitosti ohledně místa na disku nebo například šířky pásma přenosových médií.

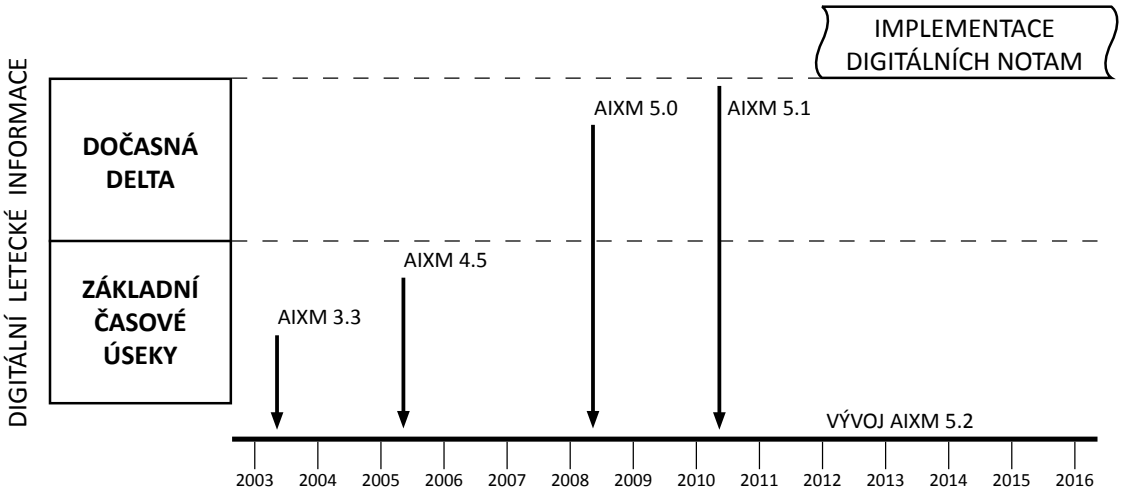
2 AIXM

Pojem AIXM (Aeronautical Information eXchange Model) představuje model pro výměnu leteckých informací v digitální podobě [7]. Jeho základním cílem je umožnit efektivní správu a distribuci leteckých informačních služeb AIS (Aeronautical Information Service) v mezinárodním civilním letectví, díky čemuž zvyšuje, mimo jiné, i bezpečnost a efektivitu navigace letového provozu. AIXM spadá pod AIM (Aeronautical Information Management), to vše zaštiťuje pak ICAO (International Civil Aviation Organization).

2.1 Charakteristika

AIXM vychází z koncepčního modelu AICM (Aeronautical Information Conceptual Model), přičemž využívá jazyků UML (Unified Modeling Language), GML (Geography Markup Language) a XML. AICM staví na definovaných vazbách a vztazích mezi objekty a jejich vlastnostmi. Obecně je založen na doporučeních z ICAO i praxe, průmyslových standardech a datových konceptech publikovaných v aeronautických zprávách. Je vhodné zdůraznit, že AIXM je pouze jedna z možných implementací AICM, která nevyužívá veškerých jeho možností. Při použití vhodných formátů pro výměnu dat lze tedy implementovat i jiná řešení [8].

Na vývoji AIXM spolupracuje evropská organizace pro bezpečnost letové navigace EUROCONTROL s americkou FAA (Federal Aviation Administration). Počátky návrhu AIXM se datují do 90. let minulého století a jak ukazuje obrázek 2.1, vývoj neustále pokračuje a model je i nadále vylepšován.



Obr. 2.1: Vývoj AIXM z pohledu modelu dočasnosti [8]

Poslední dostupná verze AIXM 5.1 přinesla hned několik klíčových principů. Model je nyní modulární[1] a rozšiřitelný, úzce využívá územní standardy organizace ISO (International Organization for Standardization), pracuje s obsáhlým tzv. *modelem dočasnosti* (kap. 2.3) včetně podpory informací obsažených v digitálních NOTAM (NOtice To AirMen) zprávách (kap. 2.4), mimo jiné podporuje aktuální identifikační kódy letišť a uživatelská data obsahující například druhy překážek oproti běžnému provozu, terminálové procedury nebo databázové mapování letišť [7].

AIXM sestává ze dvou hlavních komponent. Koncepčního modelu AIXM a AIXM XML schématu. Koncepční model se věnuje letecké doméně, popisuje vnitřní objekty AIXM, jejich vlastnosti, atributy a vzájemné asociace. Může být použit jako logický základ pro AIM databáze. Oproti tomu schéma se již více přiklání k výměně leteckých dat a samotné implementaci koncepčního modelu. Právě díky schématu je AIXM tím, čím je. Oběma těmto prvkům se více věnuje kapitola 2.2.

2.2 Konvence modelování AIXM

Koncepční model AIXM je logický informační model vyjádřený v UML [8]. Jedním jeho prvkem je i tzv. *abstraktní model*, který pojednává o zjednodušeních, které jsou v modelu provedeny. Například o situaci, kdy by měl koncepční model obsahovat sady abstraktních AIXM tříd, které jsou stavebními bloky schématu, ale ve skutečnosti nejsou zobrazeny na žádném z diagramů a ani v UML opravdu neexistují. Jejich existence se pouze navenek předpokládá, a to vše právě pro zjednodušení konverze UML modelu na XML schéma [12].

Formát pro výměnu dat je v AIXM tvořen sérií XML schémat. Mezi koncepčním modelem a schématy je přímé spojení. Toho využívá AIXM a vzniká konverzí koncepčního modelu na AIXM XML schéma.

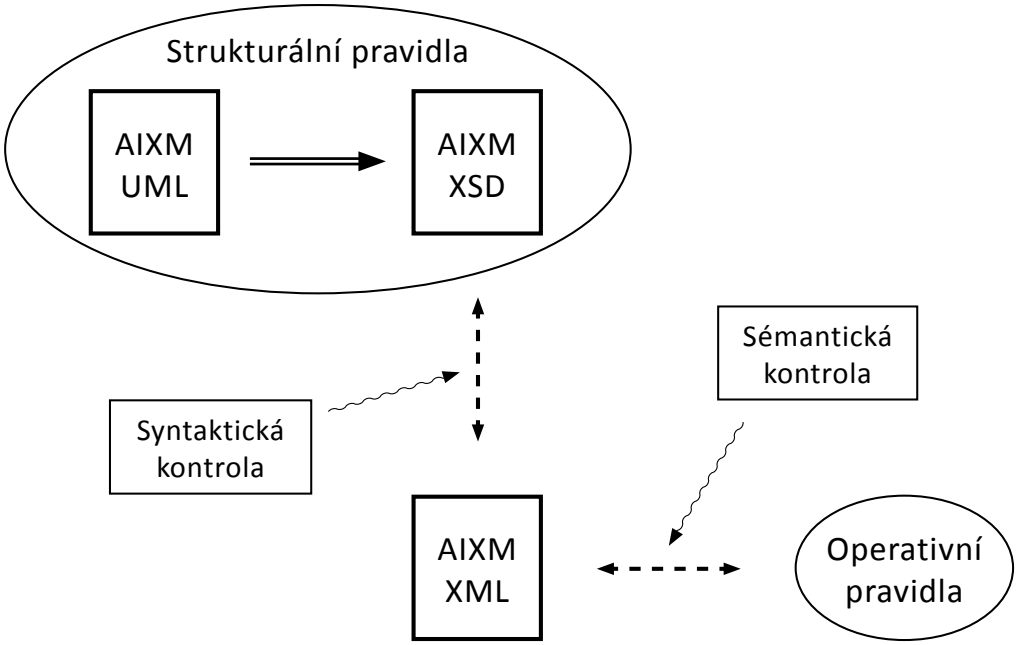
Celý pracovní proces v AIXM podléhá mnohým validacím a kontrolám, ať už sémantickým nebo syntaktickým, které se označují jako SBVR (Semantics of Business Vocabulary and Business Rules). Obrázek 2.2 popisuje princip modelování AIXM.

SBVR rozlišuje dva druhy mezi sebou kontrastujících pravidel. Prvním jsou tzv. *strukturální* pravidla, která se zabývají strukturou a jejím uspořádáním. Druhým je soubor tzv. *operativních* pravidel. Ta upravují chování uživatelských aktivit a mohou být přímo změněna nebo porušena odpovědným personálem.

2.2.1 Prvky koncepčního modelu AIXM

Stručný přehled nejdůležitějších prvků vycházejících z [12]. Prvky a jejich pravidla dohromady tvoří doménu leteckých dat.

[1] Možnost připojení přídavných součástí za účelem zdokonalení stávajícího modelu.



Obr. 2.2: Modelování AIXM [23]

Typy diagramů

Používají se dva typy diagramů, diagramy třídy a balíčku. První případ reprezentuje součásti, vlastnosti, vztahy a dědičnost. Druhý pak rozděluje koncepční model do více dílčích modulů a identifikuje závislosti mezi sadami tříd.

Stereotypy

Jednotlivé třídy se rozlišují svými stereotypy. Stereotypy blíže určují a rozšiřují standardní UML koncepty.

Příklady stereotypů jsou `feature`, `object`, `choice`, `datatype` nebo `codelist`.

Abstraktní třídy

Některé třídy mohou být navíc abstraktní. V modelu se vyznačují tím, že se píší kurzívou. Abstraktní třídy nelze realizovat v klasické implementaci (např. XML dokument), namísto toho se jich využívá jakožto základních tříd v hierarchii dědičnosti.

Například, základní vlastnosti každé součásti (viz další prvek) jsou popsány právě abstraktní třídou, od té také daná součást může dědit (viz Dědičnost).

Součásti

Jsou základem AIXM a popisují objekty ze skutečného světa, přičemž mohou být jak konkrétní, tak abstraktní. Mohou se také měnit v čase. Součásti jsou reprezentovány jako třídy se stereotypem `feature`. Příkladem může být ranvej či procedura.

Součásti jsou dynamické prvky. K popisu změn, které mají vliv na součásti v průběhu času, se používají tzv. *časové úseky* (kap. 2.3).

Objekty
Jedná se o abstrakci entit z reálného světa, resp. vlastností těchto entit, které existují uvnitř součástí. Objekty se v AIXM tvoří ze dvou důvodů. A to v případě, že je násobnost entit větší než jedna, a když má objekt vlastní atributy, které jsou opětovně používány napříč modelem.

V obou případech jsou objekty reprezentovány s patřičnými asociačními UML vztahy mezi ostatními prvky.

Výběry
Jsou to druhy tříd, které se využívají při modelování XOR[2] vztahů mezi prvky. Umožňují specifikovat použití konkrétního prvku v případě více způsobů vyjádření.

Vlastnosti
Spadají do nich atributy a vztahy charakterizující objekty a součásti.

Atributy jsou použity k popisu jednoduchých vlastností objektů a součástí, kdežto vztahy určují druh asociace mezi nimi. Vztahy se používají, když je třeba vyjádřit více vlastností. V případě použití atributu se vyjadřuje vlastnost jedna.

Dědičnost
Schopnost jedné třídy zdědit vlastnosti třídy druhé.

V AIXM je dědičnost povolena pouze v rámci jednotlivých prvků, nelze tedy například dědit vlastnosti objektu od součásti.

2.2.2 AIXM XML schéma

Důležitým prvkem AIXM je XML schéma, mnohdy reprezentované jako `*.xsd` soubor. Schéma si lze představit ale i jako tok bajtů mezi aplikacemi nebo například záznamy v polích databáze. Každopádně může být chápáno jako šablona AIXM dokumentu definující přípustnost jeho struktury, komplexních typů, prostých typů a elementů. Představuje také samotný výměnný model pro data.

Výše zmíněné se uvádí i ve W3C doporučeních s dodatkem, že cílem schématu je, mimo jiné, definice tříd XML dokumentu, přičemž k jeho popisu se zavádí termín tzv. *instance dokumentu*. Při použití úplného nebo třeba i částečného schématu na dokumentu, který odpovídá AIXM požadavkům, se uvádí pojem tzv. *AIXM zpráv*.

Mezi typy AIXM zpráv lze zařadit například tzv. *AIXM-Update*, což je zpráva, která umožňuje zaktualizovat dané části dříveji používaných dat, aniž by se musela tato data opětovně celá přenášet [8].

Hlavní jádro formátu pro výměnu dat v AIXM je tvořeno třemi základními soubory. Těmi jsou `AIXM_AbstractGML_ObjectTypes.xsd`, `AIXM_Datatypes.xsd`

[2]Exkluzivní logický součet. Pravdivý při unikátní hodnotě každého ze vstupů.

a `AIXM_Features.xsd`. První definuje základní konstrukce AIXM objektů a součástí. Zároveň podléhá schématu metadat podle ISO19139. Druhý soubor obsahuje XML reprezentace všech datových typů definovaných v UML modelu. Třetí soubor také obsahuje XML reprezentace. Zde ale v podobě všech součástí AIXM včetně jejich vlastností.

2.2.3 Výhody použité konvence

V následujícím souhrnu je uvedeno několik větších výhod používané konvence [10].

Výhody koncepčního modelu:

- Reprezentuje koncepty z reálného světa, jakožto teoretické pojmy srozumitelné automatizovaným systémům.
- Poskytuje základ pro logické datové struktury použité během implementace.
- Umožňuje dosažení nestrannosti vůči aplikacím a jejich pohledu na data.
- Standardizuje koncepční pojetí leteckých údajů všech zúčastněných stran, tzn. dorozumívání se stejným jazykem.

Výhody schématu:

- Zvýšení bezpečnosti (např. díky redukci nesrovnalostí v datech). Navíc, počítačově interpretované údaje znamenají méně chyb pro piloty a ostatní členy letového systému.
- Snížení nákladů. AIXM model může být opětovně použit ve více softwarových řešeních. Při využití průmyslových standardů lze využít komerčních i tzv. *open-source*[3] nástrojů. Využití digitálních vstupů a výstupů umožňuje zredukovat kontrolu kvality dat a zároveň náklady na jejich integraci.

Pozn.: Samozřejmě lze ve výše popsané konvenci nalézt i jisté nevýhody. Ty přecházejí hlavně z použitých technik, jako například UML, apod.

2.3 Model dočasnosti

Čas je jeden z nejpodstatnějších aspektů v letecké informační doméně, kde každé ohlášení o změně přichází s patřičným předstihem její platnosti. Po leteckých informačních systémech se obvykle vyžaduje jak zajištění současné situace, tak příprava na budoucí změny. Informace s prošlou lhůtou platnosti musí být navíc archivovány pro účely případného šetření.

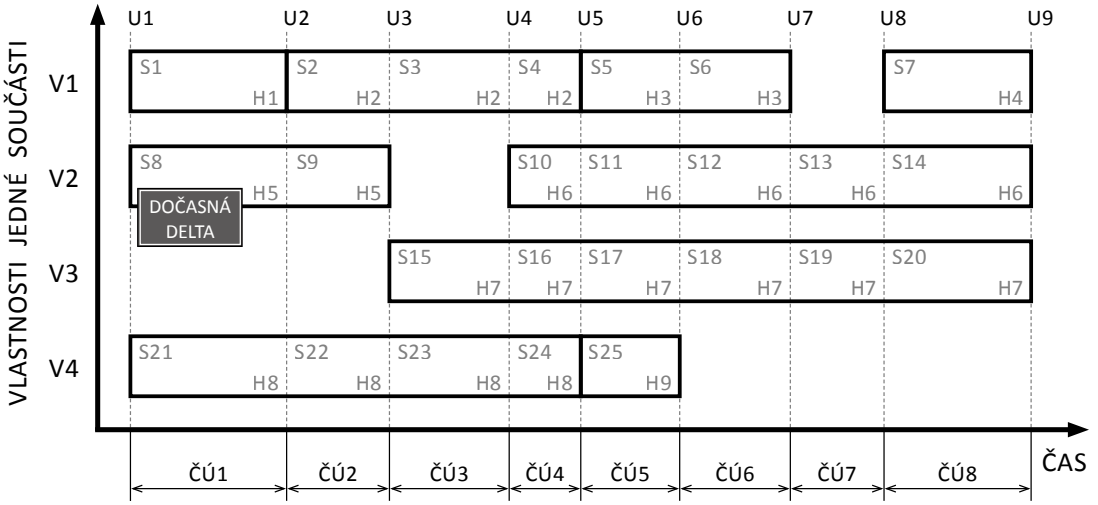
[3] Software s otevřeným zdrojovým kódem, zajišťujícím jak technickou dostupnost, tak tu legální.

Kvůli operativním důvodům se obvykle uplatňuje rozdíl mezi *permanentními*[4] *změnami* a tzv. *dočasným statusem*. Permanentní změny trvají do následující permanentní změny nebo do konce životnosti součásti (kap. 2.2.1). Oproti tomu dočasný status představuje změny s omezeným trváním. Považuje se za překrytí stavu[5] součásti, resp. její vlastnosti, v čase. To lze označit jako tzv. *koncept překrytí*. Jakmile dočasná změna skončí, dále se neuplatní a pomocí tzv. *konceptu navrácení* se změní stav dané součásti, resp. její vlastnosti, na původní hodnoty [11].

Model dočasnosti se tedy stará o přesné vyjádření stavů jednotlivých součástí a událostí v čase. Umožňuje také implementaci digitálních NOTAM zpráv do AIXM.

Klíčovým předpokladem modelu dočasnosti je, že se vlastnosti součástí mohou měnit v čase. Výjimku zde ale tvoří případ globálního unikátního identifikátoru, který tímto způsobem u součástí měnit nelze.

Nejen shrnutí výše popsaných pojmů zobrazuje obrázek 2.3. Ten popisuje vlastnosti jedné součásti (V), jejich hodnoty v čase (H) a časové úseky (ČÚ), přičemž ty lze chápat jako stavy (S) jednotlivých vlastností v jistých časových intervalech, které vymezují události (U). Časové úseky jsou de facto abstraktní jednotky modelu dočasnosti v podobě kontejnerů se zapouzdřenými časově proměnnými vlastnostmi.



Obr. 2.3: Konstrukce modelu dočasnosti v AIXM

Díky míře všestrannosti modelu dočasnosti mohou být časové úseky dále děleny. A to na časové úseky *základní* nebo *dočasné*. Základní se aplikují na všechny vlastnosti dané součásti a popisují stav součásti jakožto výsledek permanentní změny, kdežto dočasné se aplikují pouze na jednu z vlastností a popisují přechodné překrytí

[4]Trvalými, stálými.

[5]Sada všech vlastností součásti v daném časovém okamžiku.

stavu součásti v souladu s konceptem překrytí a navrácení. Dočasným časovým úsekům se někdy též říká tzv. *dočasná delta* [11]. Právě díky nim lze aplikovat digitální NOTAM zprávy do AIXM.

2.4 Digitální NOTAM

Klasické NOTAM zprávy jsou ve formě upozornění distribuované prostřednictvím telekomunikačních služeb. První z nich vznikaly již v polovině minulého století. V dnešní době obsahují informace zabývající se organizací, stavem a změnami jakéhokoliv aeronautického zařízení. Dále popisují služby, procedury a možná nebezpečí, jejichž včasná znalost je nezbytná pro personál letového provozu. Vytváří je státní letecké autority, definované v leteckých zákonech. Tyto zodpovědné instituce jednotlivých států si je pak vyměňují mezi sebou. Jejich platnost může být několik hodin nebo například do další případné změny.

Digitální NOTAM zprávy jsou tvořeny převodem obyčejného textu z klasických NOTAM do strukturované a digitalizované podoby, která umožňuje automatizované zpracování informace [11]. Jsou ve formě aktualizací datových souborů, a oproti klasickému zpracování personálem se zpracovávají pomocí specializovaných systémů [9].

Příklad digitální NOTAM zprávy:

```
280847 EBBRYNYN
(A1143/02 NOTAMN
Q) EBBU/QOBCE/IV/M/A/000/999/5027N00427E002
A) EBCI
B) 0208280800
C) 0210301400EST
E) CRANE ERECTED 25M AGL - 200M AMSL AT 1200M ARP ON A MAG BRG
OF 40 DEG. AND AT 450M RIGHT SIDE OF THR25. NO ICAO MARKINGS.)
```

Soupis několika možných informací obsažených v NOTAM:

- varování před nebezpečím (letecké dny, seskoky padákem, aktivity modelářů)
- významné lety (státníků, humanitární pomoc)
- uzavření letiště nebo některé jeho součásti
- porucha na navigačním prostředku (světelné návěstidlo)
- vojenské cvičení omezující jinak volný vzdušný prostor
- dočasné překážky v okolí letiště (jeřáby)
- zvýšený výskyt migrujícího ptactva (tzv. *BIRDTAM*)
- problémy na letišti způsobené sněhem nebo ledem (tzv. *SNOWTAM*)
- problémy na letišti způsobené vulkanickou činností (tzv. *ASHTAM*)

Jak ukazuje obrázek 2.1, je digitální NOTAM stále ve vývoji a postupné implementaci. Mnoho zemí světa ale těží z jeho výhod již dnes, hlavně díky praktickému rozšíření aplikace modelu dočasnosti v AIXM.

Pozn.: NOTAM zprávy pro Českou republiku spravuje Řízení letového provozu České republiky. České veřejné NOTAM zprávy jsou díky webové letecké informační službě dostupné na: notam.rlp.cz.

3 KOMPRESE XML

Kompresí lze obecně nazývat operaci, při které se identifikuje a odstraňuje redundance[1] z libovolných dat.

Jak již vybízí závěr kapitoly 1, lze mnohdy na XML data využít jistá komprese, abychom původní velikost dokumentu co možná nejvíce zmenšili. Tato redukce vytváří pro různá řešení různé výhody. Zmenšení originálního souboru se promítne například na velikosti místa na úložném médiu nebo třeba na přenosové kapacitě linky.

3.1 Charakteristika

Kompresi dat lze klasifikovat do dvou základních druhů:

Ztrátová komprese:
Je efektivnější než komprese bezeztrátová, avšak za cenu snížení přesnosti rekonstrukce. Její algoritmy[2] obvykle zmenšují objem dat na zlomek původní velikosti. Praktikuje se typicky při zmenšování audia a videa, případně obrazových materiálů. Jedná se o pouhou aproximaci[3] originálních dat, přičemž se většinou využívá nedokonalosti lidských smyslů [14]. Při této kompresi se ztrácejí méně důležité informace, které již nelze zpětně zrekonstruovat.

Bezeztrátová komprese:
V závislosti na typu vstupních dat obvykle nezmenšuje rapidně velikost souboru, ale vždy zachovává kompletní informaci. Po dekompresi[4] se tedy rekonstruují originální data. Využívá se převážně tam, kde by ztráta i jediného znaku mohla znamenat nenávratné poškození souboru. Lze aplikovat i na výše zmíněná multimediální data, obvykle se ale používá ke kompresi textových dat [31].

Tato práce se soustřeďuje právě na tuto kompresi.

V odvětví komprese lze najít mnoho vztahů popisujících kvalitativní vlastnosti provedených kompresí. Asi jako nejdůležitější je bezesporu tzv. *kompresní poměr*, který se dá vyjádřit vztahem

$$\texttt{kompresní poměr} = \frac{\texttt{velikost komprimovaných dat}}{\texttt{velikost nekomprimovaných dat}} \tag{3.1}$$

a vyjadřuje velikost dat výstupních ku vstupním. Je to bezrozměrný parametr. Někdy se lze setkat s vyjádřením v procentech. V takovém případě pak vyjadřuje

[1] Nadbytečné množství informace.
[2] Teoretický princip řešení problému.
[3] Nahrazení přibližnou hodnotou.
[4] Opačný proces k operaci komprese.

kompresní poměr, kolik procent velikosti z originálních dat má zkomprimovaný soubor. Převrácená hodnota kompresního poměru ze vztahu 3.1 se nazývá *kompresní faktor* [31]. Jestli je kompresní faktor větší než jedna, došlo ke kompresi.

Ke kompresnímu poměru se váže i úspora místa, kterou popisuje vztah 3.2.

$$\texttt{úspora místa} = 1 - \texttt{kompresní poměr} \tag{3.2}$$

Tento vztah říká, jaká část velikosti originálních dat se díky kompresi zredukovala, resp. o kolik se původní soubor zmenšil. Opět lze vyjádřit i v procentech.

Kterákoliv komprese se vždy týká dvou algoritmů. Kompresního, který odstraňuje redundanci v původních datech. A k němu opačnému, dekompresnímu, nebo někdy také rekonstrukčnímu, který se snaží o obnovení originálních dat.

Komprese obecně zpracovává vstupní data ve dvou krocích ustálenou metodikou kompresních technik. Prvnímu kroku se říká *modelování*. Ve vstupních datech se hledá redundance, jejímž popisem se vytváří tzv. *model*. Kompresor (kodér) i dekompresor (dekodér) musí model znát. Model může být *statický* nebo *adaptivní*. Parametry kódování vstupních dat u statického modelu jsou neměnné, zatímco u adaptivního se mohou měnit. Druhému kroku se říká *kódování*. V této fázi se již provádí generování komprimovaných dat. Navíc se binárně kóduje popis modelu a popis odlišností vstupních dat od modelu [31].

3.2 Kompresní mechanizmy a struktura

Kompresní mechanizmy se dělí do dvou hlavních skupin v závislosti na tom, jak pohlíží na XML dokument v závislosti na struktuře [25].

3.2.1 Komprese XML jako textu

Při kompresi XML jako textu je na dokument nahlíženo jako na běžný soubor, který se komprimuje jako celek bez ohledu na svou vnitřní strukturu. Jelikož je XML v podstatě textový soubor, lze aplikovat některý z efektivních algoritmů specializujících se na kompresi textu.

GZip:
Jedná se o bezeztrátovou metodu komprese založenou na algoritmu *Deflate*[5] využívající při kompresi dále LZ77 (Lempel-Ziv 1977) a Huffmanovo kódování. Samotná komprese metodou GZip probíhá ve dvou bodech. V prvním se pomocí LZ77 odstraní opakující se řetězce, v druhém pak nastupuje Huffmanovo kódování.

[5] Úspěšný kompresní mechanizmus z roku 1996.

GZip je moderní a populární kompresor, který je implementovaný v mnoha vývojových prostředích a operačních systémech. Využívá ho například i známý formát `*.zip`. Odtud také pochází slangový výraz „zazipování“ souboru.

Jeho výhodou je poměrně rychlá komprese a dekomprese. Zároveň také není příliš paměťově náročný. Komprimovaná data jsou tvořena bloky, kde každý blok může být uložen bez kódování, kódován předem dohodnutým Huffmanovým stromem nebo kódován Huffmanovým stromem, který je již součástí daného bloku [5].

BZip2:
Jedná se o otevřený, bezeztrátový kompresní algoritmus. Celkově je účinnější než GZip, ale lehce pomalejší. Nevýhodou by se mohlo zdát, že neumí pracovat s více soubory. To vychází ze základů systému UNIX[6], který využívá svých programů, aby spojil více souborů dohromady a ty pak mohl BZip2 v jednom souboru zkomprimovat.

Po prvních verzích bylo původní aritmetické kódování nahrazeno Huffmanovým. Komprimuje bloky dat velké od 100 do 900 kB s krokem 100 kB, přičemž kombinuje techniky MTF (Move-To-Front transform), RLE (Run-Length Encoding), Huffmanovo kódování a BWT (Burrows–Wheeler Transform) [27].

PPM:
PPM (Prediction by Partial Matching) je adaptivní, statistická kompresní metoda, která je založená na vytváření kontextových modelů a předpovědi znaků [4]. I když je tato metoda paměťově i časově náročná, již od svého vzniku v 90. letech je vysoce efektivní. Jisté varianty tohoto mechanizmu se v dnešní době používají v každém lepším komprimačním programu.

Každý kontextový model si udržuje statistiky o prošlých znacích a zároveň má definováno, kolik těchto znaků si bude pamatovat. Tato metoda pracuje hned s několika modely zároveň, kde každý si udržuje část obsahu. V závislosti na počtu znaků v modelech se určuje *stupeň* PPM. Existují varianty, kde stupeň není nijak pevně stanovený. Ty pak nejsou limitovány délkou kontextu a označují se jako PPM*.

Modely slouží k výpočtu predikcí jednotlivých pravděpodobností výskytu následujících znaků. Prostřednictvím vypočítaných pravděpodobností se následně kódují aritmetickým kódováním dané znaky. Po zpracování každého znaku se model upravuje tak, aby zase zachytil následující znak [15].

Určování pravděpodobností řeší různé varianty PPM různě.

[6]Operační systém podporující přístup více uživatelů a zpracovávání více procesů ve stejný čas.

LZMA:
LZMA (Lempel-Ziv-Markov-Chain Algorithm) je bezeztrátový kompresní algoritmus, založený na vylepšené a optimalizované verzi LZ77. Navíc se skládá ještě z Markovových řetězců[7] a range kodéru. Stejně jako LZ77 využívá slovníku, přičemž ten je uživatelsky nastavitelný až do velikosti 4 GB.

V mnoha případech je komprese lepší než u LZ77. Cena za to je pomalejší zpracování a značná spotřeba paměti. Oproti tomu dekomprese je několikanásobně rychlejší než komprese[22].

LZMA je výchozím kompresním mechanizmem formátu `*.7z` programu 7-Zip.

3.2.2 XML-aware komprese

Pojem vychází z anglického *aware* neboli *být si vědom* [6]. Jak už název napovídá, tyto kompresní metody využívají ve svůj prospěch znalost vnitřní struktury XML dokumentu, tudíž se mohou zaměřit přímo na sémantické informace uložené v XML datech. Ty následně používají k přípravě XML dat k další fázi komprese. V té se aplikují již běžné kompresní algoritmy, aby se tak docílilo co nejlepších výsledků.

V závislosti na dostupnosti XML schématu lze také členit tyto metody na závislé na schématu a nezávislé na schématu. V případě závislosti musí kompresor i dekompresor mít přístup k XML schématu daného souboru, aby byla komprese adekvátně provedena. Nezávislé metody pro svou plnou funkčnost toho schéma nevyžadují. Ačkoliv jsou závislé mechanizmy schopny dosahovat lepších kompresních poměrů, v praxi nejsou kvůli nejistotě trvalé dostupnosti schématu preferovány [26].

Kompresní mechanizmy v této kapitole lze zařadit i do kapitoly 3.3.2.

XMill:
Patří mezi nejznámější kompresory specializované na XML. Jako jeden z prvních pracoval s myšlenkou oddělení struktury od dat, které by pak mohl seskupit podle sémantické příbuznosti.

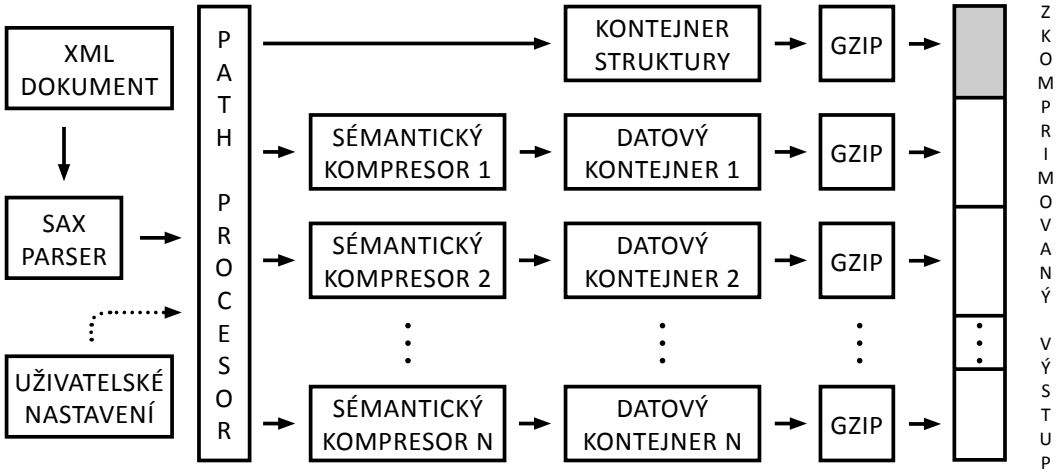
Strukturu kóduje pomocí slovníku, do kterého odkazuje názvy tagů, elementů a atributů, místo aby neustále vypisoval jejich názvy. To samo o sobě již snižuje datovou náročnost zpracovávaných dat.

XMill je založený na knihovně *zlib*, což mu dovoluje pracovat s metodami jako GZip a BZip2. Stejně tak využívá i svých vlastních sémantických nástrojů. Ty lze pro specifičtější data rozšířit i uživatelsky definovanými kompresory.

Princip fungování XMillu má tři základní body. Prvním je *oddělení struktury od obsahu*. Na oba tyto prvky se následně aplikuje komprese odděleně. Druhým bodem je *seskupení sémanticky podobných dat*, která se rozdělují do tzv. *kontejnerů* [32].

[7] Matematická metoda pro statistické modelování, která původně nesouvisela s kompresí dat.

Vždy unikátně v závislosti na označení prvků XML dat. Například opakující se elementy obsahující sémanticky podobné významy se ukládají do stejného kontejneru. Posledním bodem komprese je *použití existujících kompresních algoritmů*. Na výběr jsou kompresory pro základní datové typy, kombinované kompresory nebo uživatelsky definované kompresory. Navíc mohou být na různé kontejnery použity různé typy kompresorů. Na obrázku 3.1 je vidět architektura kompresoru XMill.



Obr. 3.1: Architektura kompresoru XMill [32]

MHMPPM:
MHMPPM (Multiplexed Hierarchical Modeling based on PPM) pracuje na podobném principu jako XMill. Rozdíl je hlavně ve způsobu kódování struktury a samotných dat.

MHMPPM komprimuje data ve dvou krocích. V prvním kóduje vstupní XML dokument metodou ESAX (Encoded SAX) a ve druhém je výstup ESAX kódován mechanizmem PPM. ESAX je založený na SAX modelu, u kterého se jednotlivé prvky dokumentu zpracovávají jako sekvence bajtů. Při jejich tvorbě si kompresor a dekompresor pamatují seznamy znaků, na které již narazily. V případě, že kompresor narazí na nový znak, přiřadí mu jedinečný kód, který zároveň zapíše na výstup. Za kódy se uvádí jednotlivé hodnoty znaků. Při opakovaném kódování stejných znaků se pak pracuje již jen se zástupnými kódy.

Původně používal ESAX pouze jeden model. Tento počet se díky MHM zvýšil na čtyři, mezi kterými se lze přepínat. Jedná se o modely pro *názvy elementů a atributů*, *strukturu prvků XML dokumentu*, *samotné atributy* a *řetězce*. Každý model si udržuje svůj vlastní stav, přičemž ale mají sdílený aritmetický kompresor.

Použitím několika od sebe oddělených modelů se dosahuje lepších výsledků komprese. MHMPPM dokáže ještě svůj mechanizmus optimalizovat vkládáním pomocných informací do jednotlivých modelů [15].

3.3 Podpora dotazování

Na kompresi XML dokumentu lze nahlížet i z pohledu přístupu ke komprimovanému souboru. K tomu slouží dotazovací jazyky jako například XPath nebo XQuery.

3.3.1 Komprese XML s podporou dotazování

Kompresní mechanizmy v této kapitole podporují dotazování na zpracování zkomprimovaných XML dat. Kompresní poměr je obvykle horší než při kompresi bez podpory dotazování [25]. Zato ale tato komprese zanechává možnost dále s daty pracovat i po zkomprimování. Nemusí se tedy soubor dekomprimovat jen kvůli tomu, aby se v něm provedla požadovaná změna. Tato schopnost se vyplatí především při práci na zařízeních s omezenými výpočetními možnostmi, při zpracovávání velkých souborů nebo při práci s enormním počtem dokumentů.

V zásadě všechny kompresní metody s podporou dotazování jsou také XML-aware (kap. 3.2.2).

Je možné nalézt i další dělení. A to na kompresory *homomorfní* a *nehomomorfní*. Homomorfní[8] zachovávají strukturu originálního XML dokumentu. Je možné tedy přistupovat ke zkomprimovanému formátu stejně jako k původním datům a například data parsovat (kap. 1.3). Oproti tomu nehomomorfní kompresory oddělují strukturu od samotného obsahu. Díky tomu neumožňují stejný princip přístupu k datům a jejich využití je tím limitováno.

XGrind:
Považuje se za jeden z prvních mechanizmů, který se začal zabývat problematikou komprese XML s podporou dotazování. Zároveň se jedná o homomorfní kompresor.

Podobně jako XMill odděluje obsah od struktury, přičemž na tu uplatňuje slovníkové kódování. Jednotlivé elementy označuje T a názvy atributů A. Za každým takovýmto kódovým označením následuje dále vždy unikátní identifikátor. Pomocí těchto identifikátorů se odkazuje do slovníku, který obsahuje originální názvy elementů a atributů. Ke kódování koncových značek se pak využívá symbolu /. Pro tyto značky není nutné vytvářet záznamy ve slovníku, jelikož se vždy obnovují z kontextu za pomoci počátečního tagu (kap. 1.2.1).

XGrind umožňuje pracovat s výčtovými typy. Ty rozpoznává s využitím DTD schémat, které kóduje pomocí $log_2 K$ kódování, kde K vyjadřuje celkový počet hodnot výčtového typu.

Obsah je zpracováván tzv. *bezkontextovou kompresí*. Díky ní se přiřazují identifikátory jednotlivým řetězcům tak, že nejsou identifikátory závislé na aktuální pozici

[8] Stejnotvarý, stejnorodý.

daných řetězců v XML datech. Zároveň také tato komprese umožňuje nalézt řetězce přímo v komprimovaných datech bez nutnosti dekomprese. To vše díky tomu, že komprimuje hledaný řetězec stejnou metodou jako vstupní soubor.

Využívá se neadaptivní Huffmanovo kódování. Pro zvýšení efektivity komprese se při kódování používají navíc i rozdílné tabulky pravděpodobnosti výskytu znaků. Tím se zohledňuje sémantika XML, jelikož ve stejných strukturách bývají data sémanticky příbuzná.

Podpora dotazování je závislá na typu dotazu. Dotaz může být na *přesnou shodu*, *shodu prefixu*, *částečnou shodu* nebo na *rozsah*. U prvních dvou se hledá element nebo atribut, který se přesně shoduje s hledaným výrazem, případně odpovídající prefix hledané hodnoty. Využívá se bajtové zarovnání namísto bitového, což přináší rychlejší porovnávání. U zbylých dvou typů dotazů se komprimuje pouze cesta dotazu. Při porovnávání se tedy musí porovnávat cesty jednotlivých prvků i s cestami daných dotazů. Tyto typy jsou náročnější na vyhodnocení, protože se obvykle musí při porovnávání projít celý soubor a všechny cesty [33].

XPress:
Při vývoji tohoto mechanizmu se vycházelo především z vlastností nástroje XGrind. XPress představuje efektivnější metody komprese XML a optimalizované principy dotazování.

XPress využívá automatické odvození datových typů s podporou provádění jejich efektivního kódování. I když se autoři inspirovali také u XMill kompresoru, na rozdíl od něj provádí XPress automatickou detekci bez nutnosti jakéhokoli uživatelského nastavování. Obsahuje také šest sémantických kompresorů, z čehož čtyři jsou rozdílové kodéry číselných hodnot a zbylé dva slouží ke kódování textu.

Stejně jako u XGrind se data komprimují bezkontextově, výstup je homomorfní a odděluje se obsah od struktury. XPress ale přistupuje naprosto odlišně ke kódování dat. To provádí *reverzním aritmetickým kódováním*, při němž přiřazuje každé cestě dotazu[9] (nebo její podmnožině) jednoznačný identifikátor z intervalu $\langle 0, 1)$. Reverzní aritmetické kódování rozděluje daný interval na subintervaly, přičemž ty jsou přiřazeny jednotlivým prvkům v závislosti na pravděpodobnosti výskytu prvku v poměru k celkové četnosti všech prvků. Každému prvku je přiřazen právě jeden subinterval.

Při vyhodnocování cesty dotazu se tedy vyhodnocují intervaly, resp. subintervaly. A to tak, že se vyhledávají prvky, jejichž označení cesty spadá do intervalu cesty dotazu. Dotaz se vyhodnotí jen u prvků, které odpovídají dané cestě. Tento postup je efektivnější než u XGrind metody, u které se vyhodnocují všechny cesty [21].

[9]Posloupnost prvků od kořene stromového vyjádření k aktuálnímu prvku.

XQzip:
Vývoj XQzip započal hlavně kvůli jistým nedostatků předchozích kompresních technik v této podkapitole. XQzip řeší efektivnějším způsobem jak samotnou kompresi, tak i dotazování. K tomu mu navíc dopomáhá i využívání vyrovnávací paměti technikou *buffer-pool*[10], která zrychluje provádění stejných nebo podobných úkonů a omezuje režii.

Problematika XGrind mechanizmu tkví v porovnávání všech cest jednotlivých prvků a dotazů. U XPress se prochází jen podmnožina intervalů. XQzip tyto problémy řeší a nastupuje s myšlenkou zavedení struktury SIT (Structure Index Tree), díky níž dochází k odstraňování duplicitních struktur. Z tohoto důvodu je vyhodnocování dotazů efektivnější, protože se prohledává pouze optimalizovaná struktura v podobě SIT. Ta přiřazuje pomocí hashovacích[11] tabulek komprimované bloky dat jednotlivým prvkům dané struktury.

Navíc podporuje širší škálu XPath dotazů nabízející tak rozsáhlejší možnosti dotazování v komprimovaných XML datech [16].

3.3.2 Komprese XML bez podpory dotazování

Kompresory této kategorie nezvládají dotazování nad komprimovanými daty. Kvůli potřebným změnám se musí soubor dekomprimovat, upravit a následně opět zkomprimovat.

Hlavní zaměření je na získání co nejlepšího kompresního poměru (kap. 3.1). Proto se na finální kompresi využívají metody z kapitoly 3.2.1 nebo například XMill.

SCMPPM:
SCMPPM (Structural Contexts Model based on PPM) kombinuje obecný model pro kompresi strukturovaných dokumentů spolu s technikou PPM. SCM vychází z myšlenky, že data uložená ve stejné struktuře budou mít podobná slovníková vyjádření a naopak, data z rozdílných struktur budou mít tato vyjádření odlišná. Opět se předpokládá sémantická příbuznost dat ze stejné struktury. Při seskupení takovýchto dat se bude dosahovat lepších kompresních výsledků.

SCM navíc ale provádí i *heuristické*[12] *slučování* sémanticky příbuzných dat do společného kontextu, protože i data z jiných struktur mohou být sémanticky příbuzná a je tedy výhodné využít i této možnosti.

Při průchodu XML dokumentem se vytváří PPM modely mezi nimiž se přepíná a zapisují se do nich data. Přepínání závisí na přiřazené struktuře danému modelu.

Ke kódování výstupu se používá aritmetický kodér [2].

[10] Technika popisující možnosti zacházení s přidělenou pamětí.
[11] Hash je krátký řetězec znaků, jakožto výstup algoritmu pro převod vstupních dat.
[12] Zkusmé řešení problémů, pro něž není znám algoritmus nebo přesnější metoda.

Komprese XML struktury:
Od běžnějších kompresních mechanizmů se odlišuje tím, že se nezaměřuje na využívání sémantických informací v XML datech za účelem sjednocení sémanticky podobných dat a následné kompresi některým z běžných kompresních algoritmů.

Zabývá se především možnostmi, jak efektivněji kódovat strukturu XML dat při využití schémat DTD (kap. 1.3.1). Tato schéma lze vložit přímo do XML dokumentu nebo na ně odkázat, jako na externí soubor, pomocí referenční odkazů [36].

Princip komprese je postaven na eliminaci redundantních informací, které obsahuje jak schéma, tak samotný dokument. Dále pak na analyzování DTD schématu kvůli zjištění charakteristiky XML dokumentu. Pro správný průběh komprese musí být dokument vždy vůči danému schématu validní, tzn. musí spolu korespondovat.

Zjištěné informace přispívají k lepším komprimačním výsledkům. Komprimovaný výstup se skládá ze tří částí. A to DTD schématu, kódované struktury XML dokumentu a samotného obsahu. K samotné kompresi lze použít některý z dříve popsaných kompresních mechanizmů.

Při dekompresi se k odvození použitých pravidel opět využije DTD schéma, které je uložené v komprimovaných datech [20].

4 TESTOVÁNÍ KOMPRESE XML

Tato část práce se zaměřuje na využití některých z dříve uvedených kompresních mechanizmů při kompresi XML dat. Veškeré důležité informace, jako je výběr testovaných souborů, metodika testování nebo volba kompresních metod, jsou popsány v kapitole 4.1.

Po ní následuje kapitola 4.2, ve které jsou prezentovány jednotlivé varianty testování a veškeré výsledky.

Celkové zhodnocení je pak uvedeno v závěru práce (kap. 6).

4.1 Parametry testovaní

Byly vybrány kompresní mechanizmy GZip, BZip2, PPM, LZMA a XMill, přičemž tato metoda byla vždy prováděna prostřednictvím unikátně vytvořeného dávkového souboru obsahující potřebné parametry a samotného programu XMill ve verzi 0.7 s výstupním formátem komprimovaných dat `*.xmi`. Zbývající čtyři metody prováděla aplikace 7-Zip 9.2 Portable[1], která byla zvolena hlavně kvůli podpoře vhodných možností nastavení. Zde byl jako výstupní formát použit `*.zip`. Jednak z důvodu kompatibility a univerzálnosti, jednak díky vykazování lepších hodnot komprese na testovacích vzorcích než formát `*.7z`, který je výchozím formátem aplikace 7-Zip. Formát `*.zip` také podporuje čtyři výše zmíněné metody, je tu navíc tedy jednotnost výstupních dat. Obě použitá programová řešení jsou open-source.

Ostatní kompresní metody nebylo možno vyzkoušet z důvodu nenalezení zdrojových kódů ani aplikací, které by daný mechanizmus zvládaly.

Hardwarové vybavení se promítá pouze na rychlosti komprese a výsledky jiným aspektem neovlivňuje. To bylo dokázáno testováním na třech různých počítačích, jejichž stáří se pohybovalo v intervalu přibližně 9 let, tzn. starý počítač (2004), starší notebook (2009) a nový počítač (2013). Na čas potřebný ke zpracování tedy nebylo téměř přihlíženo, pouze subjektivně pro potřeby závěrečného hodnocení.

Relevantní parametry testovací stanice jsou vypsány níže.

Zařízení:	notebook HP ProBook 4710s
Procesor:	Intel Core 2 Duo T6570 2,1 GHz
Operační paměť:	3 GB
Pevný disk:	Fujitsu MHZ2320BH G2 ATA
Operační systém:	32bitový Windows 7 Ultimate SP1

[1] Aplikace přenositelná mezi počítači bez nutnosti instalace.

4.1.1 Popis testovaných XML souborů

Všech pět kompresních mechanizmů, resp. jejich variace (kap. 4.2), bylo postupně použito na deset XML souborů, které byly vybrány v závislosti na jejich různorodé vnitřní struktuře a velikosti. Podrobnější soupis všech testovaných souborů lze nalézt dále v této podkapitole.

Některé ze souborů byly navíc natolik obsáhlé, že je běžné textové editory nedokázaly otevřít. V případě nutnosti byl použit program GVim 7.1.42 Portable, který je limitován až mnohem vyšší velikostí vstupních souborů (řádově GB).

Pozn.: Statistické údaje o souborech byly zjištěny z:
cs.washington.edu/research/xmldatasets/www/repository.html

aixm.xml
Souhrn deseti XML souborů obsahujících letecké informace splňující specifikace AIXM.
Více informací na: aixm.aero

dblp.xml
Databáze serveru DBLP (Digital Bibliography and Library Project). Ten poskytuje bibliografické informace o periodikách z oblasti počítačových věd.
Počet elementů: 3 332 130, počet atributů: 404 276, max. hloubka zanoření: 6.
Více informací na: dblp.uni-trier.de

epa.xml
Obsahuje geografická data o umístění zařízení spravovaných americkou agenturou EPA (Environmental Protection Agency).
Více informací na: epa.gov/envirofw/geo_data.html

mondial.xml
Databáze německého projektu Mondial, jež sbírá geografická data z různých webových zdrojů. Spolupracují například i se CIA (Central Intelligence Agency).
Počet elementů: 22 423, počet atributů: 47 423, max. hloubka zanoření: 5.
Více informací na: dbis.informatik.uni-goettingen.de/Mondial

nasa.xml
Starší datové soubory obsahující astronomická data NASA (National Aeronautics and Space Administration).
Počet elementů: 476 646, počet atributů: 56 317, max. hloubka zanoření: 8.
Více informací na: opensource.gsfc.nasa.gov

psd.xml
Název vychází z anglického označení Protein Sequence Database. Obsahuje kolekci funkčně uspořádaných údajů o proteinových sekvencích.
Počet elementů: 21 305 818, počet atributů: 1 290 647, max. hloubka zanoření: 7.
Více informací na: pir.georgetown.edu

shakespeare.xml
Soubor 37 her Williama Shakespeara v originálním znění převedených do XML.
Více informací na: xml.coverpages.org/bosakShakespeare200.html

sigmod.xml
Část bibliografických údajů o článcích ze stránky sigmod.org (Sigmod Record).
Počet elementů: 11 526, počet atributů: 3 737, max. hloubka zanoření: 6.
Více informací na: dia.uniroma3.it/Araneus/Sigmod

swissprot.xml
Databáze proteinových sekvencí s mnoha doplňujícími informacemi.
Počet elementů: 2 977 031, počet atributů: 2 189 859, max. hloubka zanoření: 5.
Více informací na: web.expasy.org/docs/swiss-prot_guideline.html

treebank.xml
Částečně zašifrované anglické texty určené především pro experimentální účely.
Počet elementů: 2 437 666, počet atributů: 1, max. hloubka zanoření: 36.
Více informací na: cis.upenn.edu/~treebank

4.2 Výsledky testování

Samotné testování probíhalo tak, že na každý z XML souborů bylo aplikováno 26 variací kompresních metod vycházejících z pěti úvodních, zmíněných v kapitole 4.1. I přes možnost dalšího nastavování nepřineslo zvýšení počtu variací výraznější změny, proto právě tento počet. Ten byl zvolen tak, aby napříč spektrem nastavitelných možností dokázal adekvátně poukázat na vliv jednotlivých parametrů na výsledky testování.

Pro jednoznačnou identifikaci jednotlivých variací kompresních metod bylo vytvořeno intuitivní kódové označení sestávající ze čtyř parametrů oddělených pomlčkami. První parametr je název kompresního mechanizmu. Druhý parametr je číselný a udává nastavenou kvalitu komprese. Může nabývat hodnot 1 a 9, přičemž 1 je nejrychlejší komprese a 9 odpovídá nejefektivnější kompresi. Třetí parametr vyjadřuje velikost slovníku, který dané metody využívají ke kompresi dat. Vždy se za tímto

parametrem uvádí jednotka. Poslední, čtvrtý parametr, je opět číselný s tím, že zobrazuje velikost použitého kódového slova dané metody. V případě, že nebyla možnost některý z parametrů z jakéhokoliv důvodu nastavit, je místo tohoto parametru uveden zástupný znak $\times$.

Ve výsledných tabulkách, které se nachází v příloze A, lze pozorovat vždy označení použitého kompresního mechanizmu, velikost XML souboru před a po komprimaci a následně vypočítaný kompresní poměr (viz vztah 3.1). Navíc každá tato tabulka obsahuje zvýrazněný řádek s nejlepším kompresním výsledkem pro daný soubor. Dále pak také vždy jednu méně zvýrazněnou buňku kompresního poměru, která odpovídá nejméně efektivnímu výsledku. Ke každému z testovaných XML souborů byla vyhotovena vždy jedna takováto tabulka.

Z těchto tabulek mohou být vypozorovány jisté podobnosti. A to například, že se kompresní poměry mechanizmů GZIP, LZMA a XMILL v průběhu testování mění, kdežto metody BZIP2 a PPM mají výsledky v jistých případech stejné, resp. u BZIP2 má vliv na kompresní poměr pouze velikost slovníku a u PPM je to zase pouze zvolená efektivita komprese. Ostatní nastavené parametry se u těchto dvou metod neprojeví.

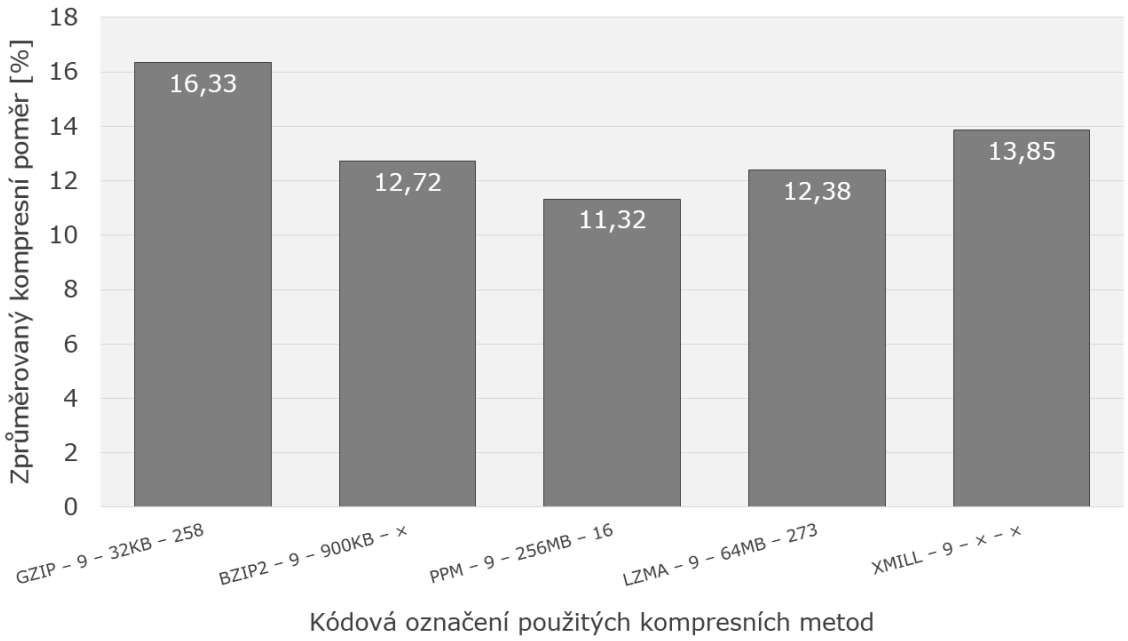
Celkově nejúspěšnější kompresní metodou byla PPM, jelikož dosahovala u šesti souborů nejlepších výsledků. Metoda LZMA takovéto výsledky měla tři a XMILL jeden. Soubor s celkově nejlepším kompresním poměrem byl `epa.xml` (2,8 %). (Toho bylo dosaženo také díky vnitřní struktuře daného dokumentu, která ho v kompresi zvýhodňovala mezi ostatními díky velkým blokům opakujících se dat, které se neměnily tak často, nebo například díky dlouhým názvům elementů, které jsou v tomto souboru mnohdy několikanásobně delší než samotný obsah uvnitř nich. To vše dopomáhá k bezkonkurenčním výsledkům tohoho dokumentu. Všechny tyto prvky dohromady mohou také jistým způsobem zkreslovat testování daného souboru.) Tento soubor měl zároveň i nejmenší rozptyl mezi svým nejlepším a nejhorším kompresním poměrem, který činil 5,5 %. Zato soubor `treebank.xml` měl tento rozptyl největší, a to 13,8 %. Průměrný rozptyl mezi nejlepšími a nejhoršími hodnotami kompresních poměrů všech testovaných XML souborů byl 9,75 %. Jednotlivé rozptyly daných souborů jsou uvedeny vždy pod patřičnou tabulkou. Průměr nejlepších kompresních poměrů všech souborů je 10,77 %. Průměr těch nejhorších pak odpovídá 20,52 %.

V příloze B jsou grafická vyjádření výsledků popisující kompresní poměry XML dokumentů při zvolených kompresních mechanizmech. Ty byly reprezentovány vždy nejvyšším možným (nejefektivnějším) nastavením parametrů v rámci zvolených variací. Stejné variace kompresních metod lze vidět i na obrázku 4.1. Ten popisuje hodnoty kompresních poměrů zprůměrovaných ze všech testovaných souborů při použití dané kompresní metody. Jasně z něj vyplývá výše zmíněné, a to že metoda PPM byla v testování nejefektivnější. Dále lze vyvodit i rozptyl, pohybující se kolem 5 %.

Patrný je i nejhorší výsledek mechanizmu GZIP, který potvrzují i hodnoty z tabulek v příloze A, kde měla obecně tato metoda devět z deseti nejhorších kompresních poměrů.

Jako poslední byl testován vliv vnitřní struktury dokumentu na kompresi XML dat. K tomu bylo opět vybráno pět výše zmíněných variací kompresních mechanizmů a tři XML soubory, které byly zmenšeny na jednotnou velikost 288 585 bajtů při zachování well-formed XML (kap. 1.2). Přehled výsledků je zobrazen v tabulce 4.1, ze které vyplývá, že i při stejné velikosti XML dokumentů záleží na vnitřní struktuře, jelikož všechny velikosti komprimovaných souborů byly rozdílné. Zvýraznění v tabulce označuje soubor s nejlepším kompresním poměrem. Soubor `modial.xml` toho dosáhl díky typické XML struktuře, zatímco `shakespeare.xml` obsahoval spíše více textového obsahu a méně elementů bez atributů a `treebank.xml` má oproti ostatním názvy elementů krátké a jejich obsah je šifrovaný, což není pro kompresi výhodné.

A na úplný závěr ještě malá zajímavost. Ze stránek *dumps.wikimedia.org/enwiki*, na kterých jsou uloženy zálohy databáze webu *wikipedia.org*, bylo zjištěno, že řádově dosahují jejich kompresní poměry desetinásobně lepších hodnot než nejlepší výsledky v této práci. Vzhledem k velikosti jejich záloh každý měsíc (desítky TB) je to pochopitelné, stále je to ale obdivuhodný výsledek, který stojí za zmínku. Zřejmě je toho dosaženo upraveným `*.7z` formátem a výborně tvořenými databázemi.



Obr. 4.1: Zprůměrované kompresní poměry použitých kompresních metod

Tab. 4.1: Výsledky komprese tří stejně velkých XML souborů

Název XML souboru	**Kódové označení kompresní metody**	**Komprimovaná velikost [B]**	**Kompresní poměr [%]**
mondial	GZIP – 9 – 32KB – 258	31 366	10,9
	BZIP2 – 9 – 900KB – ×	23 021	8,0
	PPM – 9 – 256MB – 16	21 886	7,6
	LZMA – 9 – 64MB – 273	25 100	8,7
	XMILL – 9 – × – ×	25 222	8,7
shakespeare	GZIP – 9 – 32KB – 258	75 530	26,2
	BZIP2 – 9 – 900KB – ×	57 708	20,0
	PPM – 9 – 256MB – 16	52 342	18,1
	LZMA – 9 – 64MB – 273	69 092	23,9
	XMILL – 9 – × – ×	75 002	26,0
treebank	GZIP – 9 – 32KB – 258	105 199	36,5
	BZIP2 – 9 – 900KB – ×	94 538	32,8
	PPM – 9 – 256MB – 16	96 508	33,4
	LZMA – 9 – 64MB – 273	98 959	34,3
	XMILL – 9 – × – ×	90 617	31,4

5 KOMPRESE AIXM

Jakožto výstupu diplomové práce je tato kapitola věnována hlavní praktické části, a to aplikaci, která umožňuje kompresi AIXM dat. S přihlédnutím k dosavadním poznatkům bylo vhodné využít textovou podobu těchto dat a aplikovat metody pro ztrátovou i bezeztrátovou kompresi. Při ztrátové kompresi se prostřednictvím programu odstraňují nedůležité tagy a jejich obsah (kap. 1.2), kdežto při bezeztrátové se využívají kompresní mechanizmy LZMA a XMILL (kap. 3.2). Ty reprezentují moderní open-source řešení rychlých a přesto kvalitních kompresních metod. Obě tyto dostupné metody byly vybrány s přihlédnutím na dřívější výsledky testování komprese XML dat. LZMA reprezentuje efektivnější kompresní mechanizmus, kdežto XMILL zastupuje ten rychlejší.

5.1 Program s uživatelským rozhraním

Výsledný program byl vytvořen v programovacím jazyce C# prostředí .NET[1], jehož některé využité funkce jsou zmíněny níže. Z toho plyne jediný relevantní minimální požadavek na spuštění aplikace, a to nainstalovaný tzv. *.NET Framework*[2] ve verzi alespoň 3.5.

Pro správnou funkčnost programu je třeba mít právo zápisu do adresáře, ze kterého se spouští `*.exe` soubor aplikace. Dále je také nutné, aby se ve stejném umístění nacházela složka `AIXMcompress`, která obsahuje potřebné soubory pro kompresi a dekompresi. Konkrétněji `lzma.exe` (komprese XML a dekomprese LZMA), `xmill.exe` (komprese XML) a `xdemill.exe` (dekomprese XMILL). V případě nedodržení těchto požadavků aplikace informuje uživatele ihned po spuštění, případně v některé z klíčových fází běhu programu. V každém případě je aplikace stabilní a na nastalé situace umí adekvátně reagovat.

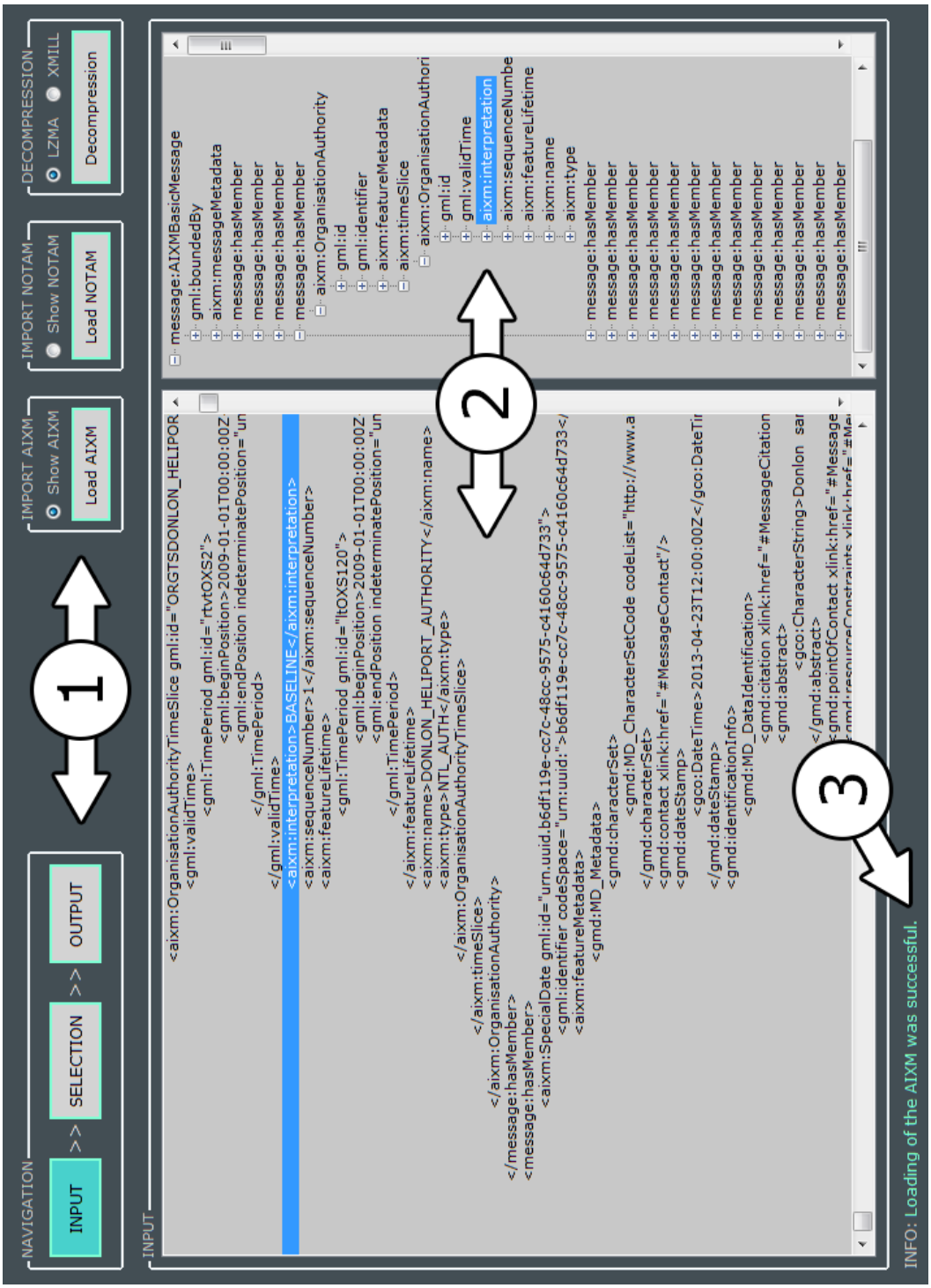
Z důvodu zvýšení zpětné kompatibility bylo zvoleno rozlišení GUI (Graphical User Interface) v klasickém poměru 4:3 a velikosti 1000×750 pixelů. Díky tomu mohou uživatelé bez problému spouštět aplikaci i v rozlišení obrazovky 1024×768 pixelů a vyšším.

Jednotlivé prvky programu lze v GUI rozdělit do tří oblastí. Ty jsou pod číselným označením 1, 2 a 3 zobrazeny na obrázku 5.1.

První oblast (označena jako 1) tvoří ovládací panel, na kterém lze vždy v jeho levé části nalézt prvek navigace (`NAVIGATION`), který slouží k průchodu programovými částmi `INPUT` (kap. 5.1.1), `SELECTION` (kap. 5.1.2) a `OUTPUT` (kap. 5.1.3). V případě, že programová část má nějaké ovládací prvky, jsou vykresleny v pravé části.

[1] Soubor softwarových technologií tvořící platformu pro prostředí webu a OS Windows.

[2] Součinnost programovacího jazyka, vývojového prostředí, virtuálního stroje a knihoven.



Obr. 5.1: Vzhled GUI aplikace při načtení AIXM dat

Neméně důležitou částí GUI je i prohlížeč aktuálně zpracovávaných dat, neboli panel náhledu (označen jako 2). Ten může zobrazovat výpis XML kódu a stromovou strukturu vytvořenou parsováním. Obě možnosti slouží k co možná nejlepšímu popsání aktuálně zpracovávaných dat. K lepší uživatelské orientaci je možné v části `INPUT` a `OUTPUT` využít možnosti označení libovolného prvku stromové struktury, přičemž tato operace označí zároveň i přidružený řádek kódu ve výpisu XML dat.

Poslední z hlavních součástí uživatelského rozhraní je informační panel (označen jako 3). Ten zobrazuje mnohdy užitečné informace v každém okamžiku běhu programu. Textový výstup je ještě barevně odlišen v závislosti na tom, jaký typ sdělení je uživateli předkládán. Textové zprávy s azurovým odstínem jsou pouze informativní a uživatel na ně nemusí brát ohled, většinou se jedná o potvrzení o korektně provedeném úkonu. Opak tvoří zprávy se žlutým zbarvením. Takovýto text obvykle informuje o nějaké chybě, či nedostatku, který je ke správnému pokračování třeba napravit. Příkladem může být případ, kdy uživatel nevybere žádný element v sekci `SELECTION`. Obecně lze dělit chybová hlášení na dva druhy. Ty, které aplikaci nijak neomezují a jsou zobrazeny pouze v informačním panelu, a ty, které již informují o kritických problémech, které je potřeba pro správnou funkčnost vyřešit. V takovýchto případech navíc aplikace upozorní uživatele prostřednictvím oznamovacího okna s chybovou hláškou.

Samotný program sestává ze tří hlavních programových částí. Jejich názvy představují tlačítka v levé části ovládacího panelu. Lze mezi nimi tímto způsobem přepínat, přičemž základní postup v aplikaci je myšlen směrem zleva doprava ve směru pomocných šipek. Podrobnější popis je uveden v patřičných podkapitolách. V těch jsou také odkazy na ukázky kódu v příloze C. Kompletní kód lze najít na médiu přiloženém k výtisku této práce, případně vyhledat v příloze k práci na školních webových stránkách v sekci: *Databáze závěrečných prací*.

5.1.1 Vstupní programová část

Tato část je aktivní hned po spuštění aplikace. Je také možné se do ní kdykoliv vrátit přes tlačítko `INPUT`.

První věc, kterou program dělá, je ověření, zda se v jeho adresáři nachází všechny potřebné soubory. Je vhodné podotknout, že aplikace může fungovat i bez těchto souborů. Je třeba ale mít na paměti, že její funkce budou omezeny. Sdělení v informačním panelu při úspěšném spuštění je vidět na obrázku 5.2.

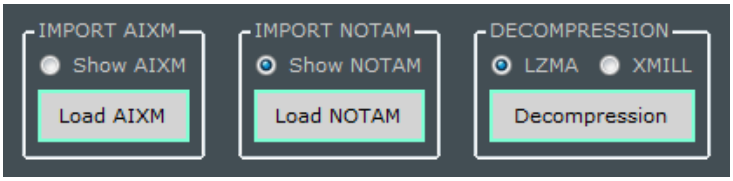
Obr. 5.2: Informace o úspěšné inicializaci programových komponent

Následují možnosti načtení AIXM dat a NOTAM, případně dekomprimovat dříve zkomprimovaný výstup. Ve všech třech případech se vstup po provedení potřebných úkonů načítá do panelu náhledu tvořeného dvěma okny, resp. levým prvkem `listBox` (vždy obsahuje kód) a pravým `treeView` (vždy obsahuje stromovou strukturu). Načítání dat do `listBoxu` usnadňuje funkce `StreamReader()` třídy `IO` zobrazena v programu C.1. K plnění `treeView` byla vytvořena vlastní funkce, která parsuje kód na

elementy, atributy a text, přičemž ke zpracování zdrojových dat je využito funkce `XmlTextReader()` třídy `Xml` zobrazené v programu C.2.

Pro práci s prvky `treeView` se nejen v této části využívá vytváření kolekcí uzlů stromových struktur `TreeNodeCollection` třídy `Forms`. Díky tomu, a také díky použití datových typů `TreeNode`, lze lépe zpracovávat údaje a ověřovat potřebné podmínky.

Při kliku na tlačítko `Load AIXM` nebo `Load NOTAM` se spouští `OpenFileDialog()` s filtrem `*.xml` a následně plní dané prvky daty. Jak je vidět na obrázku 5.3, může uživatel přepínat mezi prvky, které budou viditelné, pomocí `Show AIXM` nebo `Show NOTAM`. Dále je v této části (přes tlačítko `Decompression`) možnost obdobně načítat komprimované soubory `*.lzma` nebo `*.xmi`, které se taktéž po dekomprimaci za pomocí funkce `Process()` třídy `Diagnostics` načítají do prvků panelu náhledu. Funkce `Process()` spouští, v závislosti na zvolené možnost nad tlačítkem dekomprese, jeden z `*.exe` souborů, určených k dekompresi, se specifickými parametry pro daný způsob dekomprese. Výsledný `*.xml` soubor vždy přejímá název z komprimovaného vstupního souboru. Ukázka kódu pro možnost LZMA je zobrazena v C.3.



Obr. 5.3: Možnosti načtení a dekomprese dat

5.1.2 Výběrová programová část

Je dostupná po kliku na tlačítko `SELECTION`, přičemž v první fázi se ověřuje, zda byla v předešlém kroku nějaká data vůbec načtena. Prvky této části se zobrazí pouze v případě, kdy tomu tak je. Přičemž není podmínkou načítat AIXM data a zároveň NOTAM. Program akceptuje i načtení pouze jedné z možností, případně načtení dat prostřednictvím tlačítka `Decompression`, jelikož po úspěšné dekompresi se výsledek vkládá, jako by uživatel vybral možnost `Load AIXM`.

Další fáze již navazuje přímo na předešlou programovou část, jelikož plnění prvku `treeView` v této sekci čerpá z `treeView` sekce předešlé. Pomocí úprav prvku byly v této programové části přidány do `treeView checkBoxy` k možnosti označení a vybrání nalezených elementů z načtených AIXM dat. Díky tomu aplikace ví, ze kterých elementů chce uživatel v poslední programové části vytvořit výstup.

Algoritmus rozpozná 56 druhů elementů, resp. jejich názvů. Jelikož se obvykle jednotlivé elementy v AIXM datech opakují, bylo generování stromové struktury upraveno tak, že se pro elementy každého rozpoznatelného názvu vytváří nadřazený

uzel se stejným názvem. Ten pak reprezentuje skupinu všech nalezených elementů s daným názvem. Je pak pro uživatele přehlednější vybírat z redukovaného seznamu než mnohdy z několikanásobně většího.

Výše zmíněné úkony popisuje C.4. Příklad tvorby podřízeného uzlu a jednoho jeho atributu je uveden v C.5, resp. C.6.

Uživatel má možnost dvou druhů výběru. Při označení nadřazeného uzlu program automaticky pracuje se všemi elementy v daném uzlu a nemusí se již označovat každý zvlášť. V případě, že si uživatel chce volit individuálně, je tu možnost nadřazený uzel ignorovat, rozbalit kompletní výpis nalezených elementů daného názvu a označit pouze jednotlivé prvky. Příkladem může být obrázek 5.5.

Při hlubším rozbalení konkrétních elementů lze vidět jejich obsah, který není kompletní reprezentací fyzických zdrojových dat, ale má pouze informativní charakter. Pro jednoznačnou identifikaci každého nalezeného elementu AIXM dat obsahuje každá položka stromového výpisu atributy `AIXM ID` a `AIXM Name` označené před svým názvem znakem `->`. Pro lepší identifikaci elementů se hledají k položkám specifické atributy popisující konkrétní nalezené elementy. Nižší důležitost je značena znakem `>` a pořadí je vždy za dvěma výše zmíněnými atributy.

Pro ještě větší uživatelské přizpůsobení je možno redukovat vykreslovaný výběr v `treeView` pouze na skupiny elementů, které si uživatel zvolí a budou v danou chvíli viditelné. Obrázek 5.6 názorně ukazuje zatržení některých z 21 `checkBoxů`, které byly voleny tak, aby odpovídaly důležitým a často se vyskytujícím elementům. Tato možnost slouží převážně v situacích, kdy uživatel ví, na které konkrétní elementy se chce zaměřit a nemá zájem procházet obvykle dlouhý výpis ostatních, v danou chvíli pro něj neužitečných, seznamů elementů.

Jak je také patrné z obou výše zmíněných obrázků, je v této sekci také jisté barevné zvýraznění elementů `Events (NOTAM)`, ať už ve stromové struktuře nebo v možnostech aktualizace jejího zobrazení. Jedná se o jednoznačnou identifikaci k rozpoznání načtených NOTAM zpráv od ostatního AIXM obsahu. Aktualizační NOTAM informace jsou obsaženy v elementech typu `Event`, proto zvýraznění právě těchto prvků.

5.1.3 Výstupní programová část

Podobně jako v předešlé programové části, i tato v prvé řadě ověřuje, zda na její vstup přichází relevantní data a až poté spouští funkce pro jejich zpracování. Vytváření náhledu výstupu pak probíhá, mimo jiné, i za pomoci C.7.

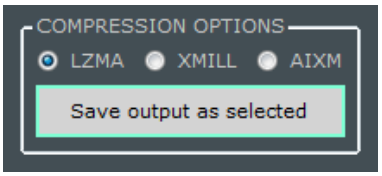
Dále tato část pracuje i se vstupními daty z první programové části, konkrétně s prvky typu `listBox`, z nichž vybírá přesné úseky dat reprezentující označené elementy ze sekce `SELECTION`. Typová funkce pro výběr relevantních úseků dat je

zobrazena v C.8. Následně se vytváří stromová struktura náhledu, pro jejíž správně vyobrazení je nutnost mít výše zmíněné právo zápisu do spouštěcího adresáře, jelikož se pracuje s dočasným úložištěm ve formě `AIXMcompress/temp.xml` souboru.

Aby byla data well-formed, program přidává ještě deklaraci (kap. 1.2.6) ve tvaru `<?xml version="1.0" encoding="UTF-8"?>`. Aplikace používá UTF-8 také pro interní kódování. Navíc je využito i XML namespaces (kap. 1.2.4). Soupis všech podporovaných je následující:

aixm	event	gml	gts	ns1	scXML	xlk
aixm5	gco	gsr	iso19115	message	smXML	xsd
amxm	gmd	gss	ns0	msg	xlink	xsi

Zatímco v průběhu výběru nalezených elementů se aplikovala pouze ztrátová komprese, zde již přichází na řadu i komprese bezeztrátová. Jak je vidět na obrázku 5.4, aplikace umožňuje uložit výstup do podoby `*.lzma` a `*.xmi`, přičemž poslední možností je `AIXM`, kdy program uloží pouze ztrátově komprimovaný soubor ve formátu `*.xml` a žádnou bezeztrátovou metodu nevyužije, viz funkce `StreamWriter()` třídy `IO` v C.9.



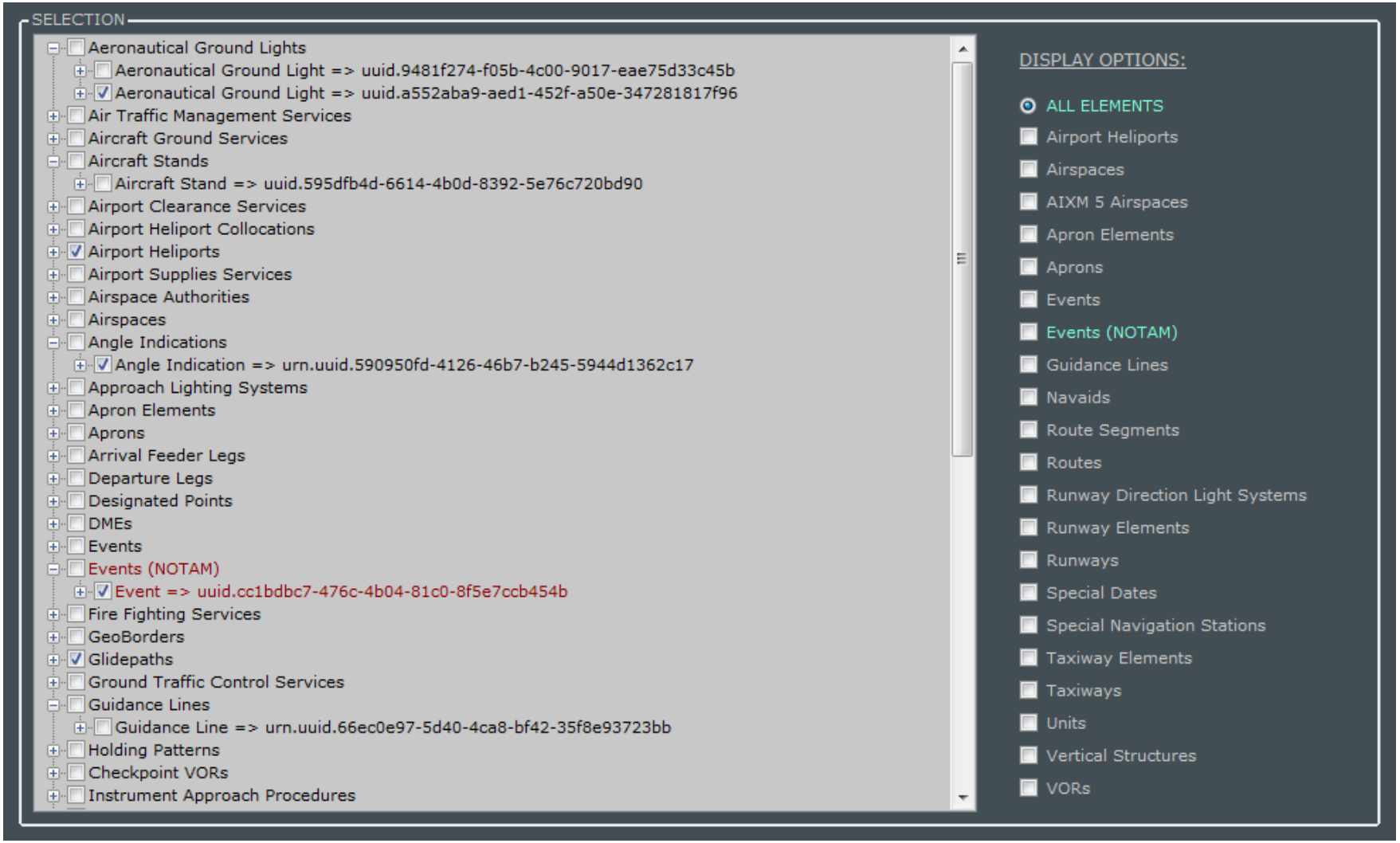
Obr. 5.4: Výběr kompresní metody

Samotná komprese probíhá obdobně jako dekomprese v C.3, pouze se ve funkci `Process()` pozmění argumenty, se kterými se spouští kompresní mechanizmus. To vše ještě s využitím kódu z C.9.

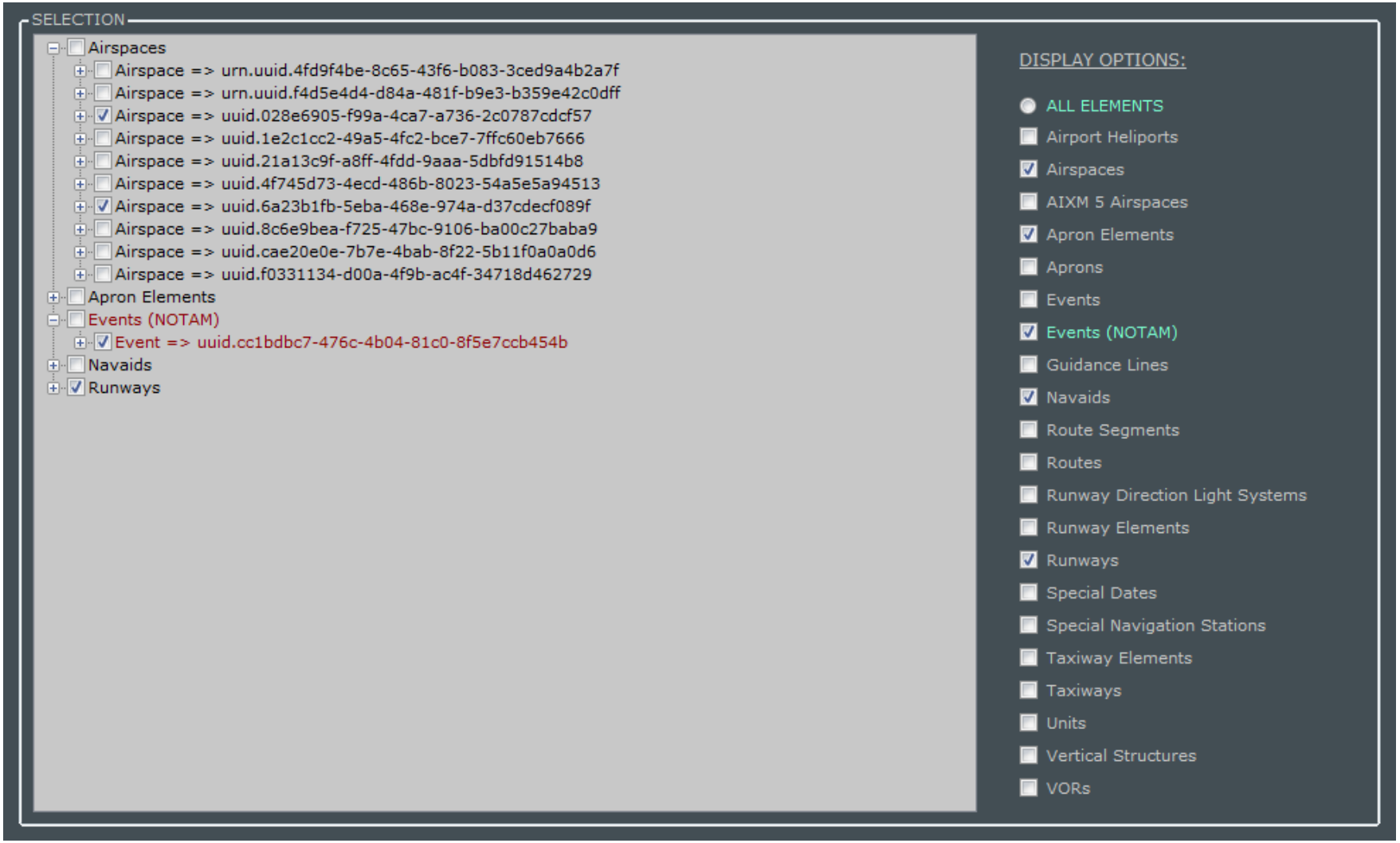
Při ukládání může nastat situace, kdy se přepisuje již existující komprimovaný soubor. V takovém případě bude muset uživatel potvrdit klasický dotaz na přepsání. Při využití metody XMILL je ale třeba, díky jejímu integrovanému dotazu na přepsání, toto potvrdit ještě jednou.

Kompresní poměr obou mechanizmů pro bezeztrátovou kompresi se pohybuje při různorodém obsahu kolem 5 %, při větším opakování stejných struktur se zvyšuje na hodnotu kolem 10 %. Bylo vycházeno z testovacích vzorků v podobě volně dostupných souborů.

Obecně lze říci, že bezeztrátová komprese ušetří přibližně 90 % z celkové velikosti vstupního souboru. Při kombinaci se ztrátovou kompresí se předpokládá dosažení ještě lepších výsledků. Samozřejmě s přihlédnutím k uživatelskému výběru.



Obr. 5.5: Možnosti výběru nalezených elementů



Obr. 5.6: Možnosti aktualizace zobrazení nalezených elementů

6 ZÁVĚR

Tato práce se zabývala kompresí XML dat. Její teoretická část popisuje charakteristiku a důležité pojmy jazyka XML, věnuje se jeho syntaxi, kódování a parsování. Je uvedeno i několik jeho výhod a nevýhod.

Pokračuje se rozšiřující kapitolou věnující se modelu AIXM, která upřesňuje zaměření práce. I zde bylo vše náležitě popsáno, jelikož se zpracované informace později promítají do praktické části práce.

Dále se práce soustřeďuje na samotnou kompresi, přičemž byla popsána její charakteristika a důležité body komprese, jako jsou druhy dělení a jednotlivé kompresní mechanizmy, které byly zařazeny do patřičných skupin. Nahlíželo se na vnitřní strukturu XML dokumentu a její vliv na celkovou kompresi. V neposlední řadě se řešila i otázka tzv. dotazování nad komprimovanými daty, resp. přístupu k nim.

V jedné ze součástí praktické části bylo úkolem testovat kompresi XML dat různými kompresními technikami. Zvoleny byly GZIP, BZIP2, PPM, LZMA a XMILL. Následně byly jejich variace použity na deset XML souborů různorodých svou vnitřní strukturou a velikostí. Výsledky těchto testování jsou k dispozici ve dvou přílohách v podobě tabulek a jejich grafického vyhodnocení. Z nastavených parametrů kompresních mechanizmů lze vyvodit, že při nastavení větší velikosti slovníku u metod, které toto umožňují, je komprese pomalejší, zato ale efektivnější. Obdobně je to i v případě nastavení větší velikost kódového slova.

Obecně se dá z daných výsledků vyjádřit o PPM jako o nejefektivnější kompresní metodě v rámci testovaných souborů. GZIP je pak v tomto srovnání nejméně efektivní. Podrobnější popis je v kapitole 4.2, ve které lze nalézt i detailnější výsledky provedených testování.

Mimo jiné byl testován i vliv vnitřní struktury na kompresi XML dat. Testování bylo docíleno zmenšením tří strukturálně rozličných souborů na stejnou velikost a následným použitím vždy všech pěti kompresních metod. Výstupem byly rozdílné hodnoty všech kompresních poměrů, tudíž mohl být vysloven závěr, že na struktuře záleží. Dané struktury byly ještě navíc i dále popsány.

V celé práci se nepřihlíží k časovému aspektu komprese, jelikož na testovacích vzorcích bylo zjištěno, že se s výkonnějšími zdroji zpracovávajícími XML data nemění výsledky, pouze čas potřebný ke zpracování byl kratší. Ten byl pozorován pouze subjektivně pro potřeby tohoto závěrečného zhodnocení. Výsledkem toho je zjištění, že z testovaných kompresních metod jsou GZIP a BZIP2 nejpomalejší, PPM a LZMA rychlejší a XMILL bezkonkurenčně nejrychlejší. Toho bylo zřejmě dosaženo díky tomu, že si byl jako jediný z testovaných kompresních algoritmů vědom specifické XML struktury, dokázal s ní pracovat a využít ji.

Hlavní součástí praktické části bylo vytvoření aplikace, která využívá ztrátovou

i bezeztrátovou kompresi XML, resp. AIXM dat. Toho bylo docíleno, mimo jiné, díky informacím zpracovaných v této práci. To vše nejenom za účelem snížení velikosti a redundance, ale také kvůli možnosti výběru dat obsažených ve výstupním souboru. Program dokáže rozpoznat desítky AIXM objektů, které si lze pomocí uživatelského rozhraní s různorodými možnostmi zobrazení označit a následně uložit pomocí kombinace ztrátové komprese a bezeztrátové komprese LZMA nebo XMILL. Případně pouze za pomoci ztrátové komprese. S takto komprimovanými daty umí aplikace následně také pracovat, jelikož dovoluje i zpětnou dekompresi a s ní spojenou případnou další kompresi dle možností výběru nalezených elementů.

LITERATURA

[1] ADAPTIC. *XML*. [online]. 2005 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: adaptic.cz/znalosti/slovnicek/xml

[2] ADIEGO, J., NAVARRO, G., FUENTE, P. *Using structural contexts to compress semistructured text collections*. Elsevier, Information processing and management, vol. 43, no. 3, pp. 769 – 790, 2007.

[3] BENZ, B., DURANT, J. R. *XML programming bible*. Wiley publishing, 2003, 987 s. ISBN 0-7645-3829-2.

[4] CLEARY, J. G., WITTEN, I. H. *Data compression using adaptive coding and partial string matching*. IEEE transactions on communications, vol. 32, no. 4, pp. 396 – 402, 1984. ISSN 0090-6778.

[5] DEUTSCH, P. *DEFLATE Compressed data format specification version 1.3*. [online]. 1996, [citováno 8. 5. 2014]. Dostupné z WWW: ietf.org/rfc/rfc1951.txt

[6] ERICSSON, M. *The effects of XML compression on SOAP performance*. Springer US, pp. 279 – 307, 2007. ISSN 1573-1413.

[7] EUROCONTROL. *Aeronautical Information Exchange*. [online]. 15. 12. 2006, [citováno 8. 5. 2014]. Dostupné z WWW: aixm.aero/public/standard_page/introduction.html

[8] EUROCONTROL. *Aeronautical Information Exchange Model*. [online]. 20. 3. 2006, [citováno 8. 5. 2014]. Dostupné z WWW: eurocontrol.int/services/aeronautical-information-exchange-model-phase-3-p-09

[9] EUROCONTROL. *Digital NOTAM*. [online]. 2001 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: eurocontrol.int/articles/digital-notam-phase-3-p-21

[10] EUROCONTROL. *Why use AIXM 5*. [online]. 28. 11. 2006, [citováno 8. 5. 2014]. Dostupné z WWW: aixm.aero/public/standard_page/introduction_benefits.html

[11] EUROCONTROL and Federal Aviation Administration. *AIXM Temporality Model*. [online]. 1. 2. 2010, [citováno 8. 5. 2014]. Dostupné z WWW: aixm.aero/public/standard_page/download.html

[12] EUROCONTROL and Federal Aviation Administration. *AIXM UML to XML Schema Mapping*. [online]. 1. 2. 2010, [citováno 8. 5. 2014]. Dostupné z WWW: aixm.aero/public/standard_page/download.html

[13] FERRARA, D. *All about XML parsers*. [online]. [citováno 8. 5. 2014]. Dostupné z WWW: webdesign.about.com/od/parsers/a/all-about-xml-parsers.htm

[14] HORDĚJČUK, V. *Komprese dat*. [online]. [citováno 8. 5. 2014]. Dostupné z WWW: voho.cz/wiki/informatika/kodovani/komprese/

[15] CHENEY, J. *Compressing XML with multiplexed hierarchical PPM models*. DCC - Data compression conference, pp. 163 – 172, 2001. ISSN 1068-0314. ISBN 0-7695-1031-0.

[16] CHENG, J., NG, W. *XQzip: querying compressed XML using structural indexing*. Springer Berlin, Advances in database technology, 9th international conference on extending database technology, pp. 219 – 236, 2004. ISSN 0302-9743. ISBN 978-3-540-24741-8.

[17] KOSEK, J. *XML pro každého*. Grada Publishing, 2000, 164 s. ISBN 80-7169-860-1. Dostupné z WWW: kosek.cz/xml/xmlprokazdeho.pdf

[18] KOSEK, J. *XML pro každého*. [online]. 2000 – 2004, [citováno 8. 5. 2014]. Dostupné z WWW: kosek.cz/xml

[19] KUBA, M. *XML snadno a rychle*. Zpravodaj ÚVT MU, vol. 8, no. 3, pp. 4 – 9, 2003. ISSN 1212-0901. [online]. 14. 11. 2011, [citováno 8. 5. 2014]. Dostupné z WWW: ics.muni.cz/bulletin/articles/268.html

[20] LEVENE, M., WOOD, P. *XML Structure Compression*. International workshop on web dynamics conference, 2002.

[21] MIN, J., PARK, M., CHUNG, CH. *XPRESS: a queriable compression for XML data*. ACM Sigmod international conference on management of data, pp. 122 – 133, 2003.

[22] PAVLOV, I. *LZMA software development kit*. [online]. 2014, [citováno 8. 5. 2014]. Dostupné z WWW: 7-zip.org/sdk.html

[23] POROSNICU, E. *AIXM 5.1 Business Rules*. [online]. 26. 7. 2013, [citováno 8. 5. 2014]. Dostupné z WWW: https://extranet.eurocontrol.int/ http://webprisme.cfmu.eurocontrol.int/aixmwiki_public/bin/view/Main/ AIXM_Business_Rules

[24] RAY, E. T. *Learning XML*. 2. vydání. O'Reilly & associates, 2003, 416 s. ISBN 0-596-00420-6.

[25] SAKR, S. *Investigate state-of-the-art XML compression techniques.* [online]. 19. 7. 2011, [citováno 8. 5. 2014]. Dostupné z WWW: ibm.com/developerworks/xml/library/x-datacompression

[26] SARK, S. *XML compression techniques: A survey and comparison.* Elsevier, Journal of computer and system sciences, vol. 75, no. 5, pp. 303 – 322, 2009.

[27] SEWARD, J. *BZip2 and libbzip2, version 1.0.5.* [online]. 10. 12. 2007, [citováno 8. 5. 2014]. Dostupné z WWW: bzip.org/1.0.5/bzip2-manual-1.0.5.html

[28] SOL, S. *Advantages and disadvantages of XML.* [online]. 2003 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: theukwebdesigncompany.com/articles/xml-advantages-disadvantages.php

[29] STARGEN. *DTD.* [online]. 2000 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: stargen.cz/slovnik/DTD

[30] STARGEN. *XSD.* [online]. 2000 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: stargen.cz/slovnik/XSD

[31] STAUDEK, J. *Komprese dat.* Brno: Masarykova univerzita, Fakulta informatiky, 2013. [online]. [citováno 8. 5. 2014]. Dostupné z WWW: fi.muni.cz/usr/staudek/vyuka/filesys/02_komprese_dat.pdf

[32] SUCIU, D., LIEFKE, H. *XMill: an efficient compressor for XML data.* ACM Sigmod international conference on management of data, vol. 29, no. 2, pp. 153 – 164, 2000. ISBN 1-58113-217-4.

[33] TOLANI, P. M., HARITSA, J. R. *XGrind: a query-friendly XML compressor.* 18th international conference on data engineering, pp. 225 – 234, 2002. ISSN 1063-6382. ISBN 0-7695-1531-2.

[34] W3C. *Extensible markup language (XML) 1.0 (fifth edition).* [online]. 26. 11. 2008, [citováno 8. 5. 2014]. Dostupné z WWW: w3.org/TR/2008/REC-xml-20081126

[35] W3C. *Extensible stylesheet language (XSL) version 1.1.* [online]. 5. 12. 2006, [citováno 8. 5. 2014]. Dostupné z WWW: w3.org/TR/xsl

[36] W3Schools. *DTD tutorial.* [online]. 1999 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: w3schools.com/dtd/default.asp

[37] W3Schools. *XML tutorial.* [online]. 1999 – 2014, [citováno 8. 5. 2014]. Dostupné z WWW: w3schools.com/xml

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AIM — Aeronautical Information Management

AIXM — Aeronautical Information eXchange Model

API — Application Programming Interface

ASCII — American Standard Code for Information Interchange

BWT — Burrows–Wheeler Transform

CIA — Central Intelligence Agency

CSS — Cascading Style Sheets

DBLP — Digital Bibliography and Library Project

DOM — Document Object Model

DTD — Data Type Definition

EPA — Environmental Protection Agency

ESAX — Encoded SAX

FAA — Federal Aviation Administration

GML — Geography Markup Language

GUI — Graphical User Interface

HTML — HyperText Markup Language

ICAO — International Civil Aviation Organization

ISO — International Organization for Standardization

JAXP — Java API for XML Processing

LZ77 — Lempel-Ziv 1977

LZMA — Lempel-Ziv-Markov-Chain Algorithm

MHMPPM — Multiplexed Hierarchical Modeling based on PPM

MSXML — Microsoft XML Core Services

MTF — Move-To-Front transform

NASA — National Aeronautics and Space Administration

NOTAM — NOtice To AirMen

PPM — Prediction by Partial Matching

RLE — Run-Length Encoding

SAX — Simple API for XML

SBVR — Semantics of Business Vocabulary and Business Rules

SCMPPM — Structural Contexts Model based on PPM

SGML — Standard Generalized Markup Language

SIT — Structure Index Tree

StAX — Streaming API for XML

UCS — Universal Character Set

UML — Unified Modeling Language

URL — Uniform Resource Locator

UTF — UCS Transformation Format

VTD-XML — Virtual Token Descriptor for XML

W3C — World Wide Web Consortium

XML — eXtensible Markup Language

XML-DEV — XML-Development

XOR — eXclusive OR

XPath — XML Path Language

XSD — XML Schema Definition

XSL — eXtensible Stylesheet Language

XSL-FO — XSL - Formatting Objects

XSLT — XSL Transformations

SEZNAM PŘÍLOH

A TABULKY

Tab. A.1: Výsledky komprese souboru **aixm.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	5 693 931	977 635	0,172
GZIP – 1 – 32KB – 258		909 328	0,160
GZIP – 9 – 32KB – 8		822 148	0,144
GZIP – 9 – 32KB – 258		692 183	0,122
BZIP2 – 1 – 100KB – ×		768 021	0,135
BZIP2 – 1 – 900KB – ×		743 519	0,131
BZIP2 – 9 – 100KB – ×		767 220	0,135
BZIP2 – 9 – 900KB – ×		742 963	0,130
PPM – 1 – 1MB – 2		822 359	0,144
PPM – 1 – 1MB – 16		822 359	0,144
PPM – 1 – 256MB – 2		822 359	0,144
PPM – 1 – 256MB – 16		822 359	0,144
PPM – 9 – 1MB – 2		753 834	0,132
PPM – 9 – 1MB – 16		753 834	0,132
PPM – 9 – 256MB – 2		753 834	0,132
PPM – 9 – 256MB – 16		753 834	0,132
LZMA – 1 – 64KB – 8		843 759	0,148
LZMA – 1 – 64KB – 273		780 904	0,137
LZMA – 1 – 64MB – 8		843 236	0,148
LZMA – 1 – 64MB – 273		777 622	0,137
LZMA – 9 – 64KB – 8		764 620	0,134
LZMA – 9 – 64KB – 273		616 233	0,108
LZMA – 9 – 64MB – 8		764 924	0,134
LZMA – 9 – 64MB – 273		**604 878**	**0,106**
XMILL – 1 – × – ×		849 306	0,149
XMILL – 9 – × – ×		770 643	0,135

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *6,6 %*

Tab. A.2: Výsledky komprese souboru **dblp.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	133 862 735	27 309 305	0,204
GZIP – 1 – 32KB – 258		25 415 833	0,190
GZIP – 9 – 32KB – 8		26 153 790	0,195
GZIP – 9 – 32KB – 258		23 086 021	0,172
BZIP2 – 1 – 100KB – ×		20 072 712	0,150
BZIP2 – 1 – 900KB – ×		15 980 652	0,119
BZIP2 – 9 – 100KB – ×		20 066 408	0,150
BZIP2 – 9 – 900KB – ×		15 979 568	0,119
PPM – 1 – 1MB – 2		17 532 560	0,131
PPM – 1 – 1MB – 16		17 532 560	0,131
PPM – 1 – 256MB – 2		17 532 560	0,131
PPM – 1 – 256MB – 16		17 532 560	0,131
PPM – 9 – 1MB – 2		10 959 349	0,082
PPM – 9 – 1MB – 16		10 959 349	0,082
PPM – 9 – 256MB – 2		10 959 349	0,082
PPM – 9 – 256MB – 16		**10 959 349**	**0,082**
LZMA – 1 – 64KB – 8		24 506 617	0,183
LZMA – 1 – 64KB – 273		21 968 162	0,164
LZMA – 1 – 64MB – 8		21 548 818	0,161
LZMA – 1 – 64MB – 273		18 055 022	0,135
LZMA – 9 – 64KB – 8		23 519 508	0,176
LZMA – 9 – 64KB – 273		20 088 795	0,150
LZMA – 9 – 64MB – 8		20 010 493	0,149
LZMA – 9 – 64MB – 273		14 281 868	0,107
XMILL – 1 – × – ×		24 852 127	0,186
XMILL – 9 – × – ×		20 445 337	0,153

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *12,2 %*

Tab. A.3: Výsledky komprese souboru **epa.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	233 268 372	19 409 820	0,083
GZIP – 1 – 32KB – 258		13 073 060	0,056
GZIP – 9 – 32KB – 8		17 052 080	0,073
GZIP – 9 – 32KB – 258		11 857 688	0,051
BZIP2 – 1 – 100KB – ×		14 524 926	0,062
BZIP2 – 1 – 900KB – ×		7 258 472	0,031
BZIP2 – 9 – 100KB – ×		14 495 775	0,062
BZIP2 – 9 – 900KB – ×		7 242 777	0,031
PPM – 1 – 1MB – 2		12 731 925	0,055
PPM – 1 – 1MB – 16		12 731 925	0,055
PPM – 1 – 256MB – 2		12 731 925	0,055
PPM – 1 – 256MB – 16		12 731 925	0,055
PPM – 9 – 1MB – 2		6 478 710	0,028
PPM – 9 – 1MB – 16		6 478 710	0,028
PPM – 9 – 256MB – 2		6 478 710	0,028
PPM – 9 – 256MB – 16		**6 478 710**	**0,028**
LZMA – 1 – 64KB – 8		12 951 237	0,056
LZMA – 1 – 64KB – 273		9 328 090	0,040
LZMA – 1 – 64MB – 8		11 184 656	0,048
LZMA – 1 – 64MB – 273		7 779 729	0,033
LZMA – 9 – 64KB – 8		11 982 824	0,051
LZMA – 9 – 64KB – 273		8 292 108	0,036
LZMA – 9 – 64MB – 8		10 825 009	0,046
LZMA – 9 – 64MB – 273		6 718 598	0,029
XMILL – 1 – × – ×		9 985 652	0,043
XMILL – 9 – × – ×		7 483 790	0,032

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *5,5 %*

Tab. A.4: Výsledky komprese souboru **mondial.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	1 906 765	291 417	0,153
GZIP – 1 – 32KB – 258		246 017	0,129
GZIP – 9 – 32KB – 8		260 959	0,137
GZIP – 9 – 32KB – 258		209 852	0,110
BZIP2 – 1 – 100KB – ×		176 023	0,092
BZIP2 – 1 – 900KB – ×		153 656	0,081
BZIP2 – 9 – 100KB – ×		175 963	0,092
BZIP2 – 9 – 900KB – ×		153 453	0,081
PPM – 1 – 1MB – 2		191 375	0,100
PPM – 1 – 1MB – 16		191 375	0,100
PPM – 1 – 256MB – 2		191 375	0,100
PPM – 1 – 256MB – 16		191 375	0,100
PPM – 9 – 1MB – 2		139 808	0,073
PPM – 9 – 1MB – 16		139 808	0,073
PPM – 9 – 256MB – 2		139 808	0,073
PPM – 9 – 256MB – 16		**139 808**	**0,073**
LZMA – 1 – 64KB – 8		248 309	0,130
LZMA – 1 – 64KB – 273		197 274	0,103
LZMA – 1 – 64MB – 8		246 183	0,129
LZMA – 1 – 64MB – 273		193 350	0,101
LZMA – 9 – 64KB – 8		234 677	0,123
LZMA – 9 – 64KB – 273		173 465	0,091
LZMA – 9 – 64MB – 8		236 566	0,124
LZMA – 9 – 64MB – 273		164 032	0,086
XMILL – 1 – × – ×		196 100	0,103
XMILL – 9 – × – ×		165 615	0,087

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *8 %*

Tab. A.5: Výsledky komprese souboru **nasa.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	25 050 431	4 299 159	0,172
GZIP – 1 – 32KB – 258		3 940 216	0,157
GZIP – 9 – 32KB – 8		3 991 673	0,159
GZIP – 9 – 32KB – 258		3 605 821	0,144
BZIP2 – 1 – 100KB – ×		3 671 333	0,147
BZIP2 – 1 – 900KB – ×		2 751 421	0,110
BZIP2 – 9 – 100KB – ×		3 663 082	0,146
BZIP2 – 9 – 900KB – ×		2 750 505	0,110
PPM – 1 – 1MB – 2		3 573 995	0,143
PPM – 1 – 1MB – 16		3 573 995	0,143
PPM – 1 – 256MB – 2		3 573 995	0,143
PPM – 1 – 256MB – 16		3 573 995	0,143
PPM – 9 – 1MB – 2		2 032 024	0,081
PPM – 9 – 1MB – 16		2 032 024	0,081
PPM – 9 – 256MB – 2		2 032 024	0,081
PPM – 9 – 256MB – 16		**2 032 024**	**0,081**
LZMA – 1 – 64KB – 8		3 753 489	0,150
LZMA – 1 – 64KB – 273		3 317 211	0,132
LZMA – 1 – 64MB – 8		3 361 152	0,134
LZMA – 1 – 64MB – 273		2 705 737	0,108
LZMA – 9 – 64KB – 8		3 538 428	0,141
LZMA – 9 – 64KB – 273		3 094 229	0,124
LZMA – 9 – 64MB – 8		3 109 822	0,124
LZMA – 9 – 64MB – 273		2 285 810	0,091
XMILL – 1 – × – ×		3 660 495	0,146
XMILL – 9 – × – ×		3 064 990	0,122

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *9,1 %*

Tab. A.6: Výsledky komprese souboru **psd.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	716 853 016	126 515 658	0,175
GZIP – 1 – 32KB – 258		114 071 970	0,159
GZIP – 9 – 32KB – 8		113 887 679	0,159
GZIP – 9 – 32KB – 258		101 117 794	0,141
BZIP2 – 1 – 100KB – ×		101 321 825	0,141
BZIP2 – 1 – 900KB – ×		76 758 069	0,107
BZIP2 – 9 – 100KB – ×		101 175 764	0,141
BZIP2 – 9 – 900KB – ×		76 720 796	0,107
PPM – 1 – 1MB – 2		99 124 513	0,138
PPM – 1 – 1MB – 16		99 124 513	0,138
PPM – 1 – 256MB – 2		99 124 513	0,138
PPM – 1 – 256MB – 16		99 124 513	0,138
PPM – 9 – 1MB – 2		65 774 082	0,092
PPM – 9 – 1MB – 16		65 774 082	0,092
PPM – 9 – 256MB – 2		65 774 082	0,092
PPM – 9 – 256MB – 16		65 774 082	0,092
LZMA – 1 – 64KB – 8		99 654 200	0,139
LZMA – 1 – 64KB – 273		91 179 588	0,127
LZMA – 1 – 64MB – 8		82 528 174	0,115
LZMA – 1 – 64MB – 273		72 686 061	0,101
LZMA – 9 – 64KB – 8		88 605 783	0,124
LZMA – 9 – 64KB – 273		78 487 992	0,109
LZMA – 9 – 64MB – 8		71 042 337	0,099
LZMA – 9 – 64MB – 273		**57 680 713**	**0,080**
XMILL – 1 – × – ×		84 914 493	0,118
XMILL – 9 – × – ×		73 472 930	0,102

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *9,5 %*

Tab. A.7: Výsledky komprese souboru **shakespeare.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	7 897 596	2 404 449	0,304
GZIP – 1 – 32KB – 258		2 288 553	0,290
GZIP – 9 – 32KB – 8		2 252 984	0,285
GZIP – 9 – 32KB – 258		2 067 636	0,262
BZIP2 – 1 – 100KB – ×		1 748 098	0,221
BZIP2 – 1 – 900KB – ×		1 605 632	0,203
BZIP2 – 9 – 100KB – ×		1 746 185	0,221
BZIP2 – 9 – 900KB – ×		1 604 251	0,203
PPM – 1 – 1MB – 2		1 564 339	0,198
PPM – 1 – 1MB – 16		1 564 339	0,198
PPM – 1 – 256MB – 2		1 564 339	0,198
PPM – 1 – 256MB – 16		1 564 339	0,198
PPM – 9 – 1MB – 2		1 453 735	0,184
PPM – 9 – 1MB – 16		1 453 735	0,184
PPM – 9 – 256MB – 2		1 453 735	0,184
PPM – 9 – 256MB – 16		**1 453 735**	**0,184**
LZMA – 1 – 64KB – 8		2 263 753	0,287
LZMA – 1 – 64KB – 273		2 103 741	0,266
LZMA – 1 – 64MB – 8		2 232 590	0,283
LZMA – 1 – 64MB – 273		2 061 802	0,261
LZMA – 9 – 64KB – 8		2 164 019	0,274
LZMA – 9 – 64KB – 273		1 967 060	0,249
LZMA – 9 – 64MB – 8		2 128 473	0,270
LZMA – 9 – 64MB – 273		1 914 417	0,242
XMILL – 1 – × – ×		2 365 648	0,300
XMILL – 9 – × – ×		2 058 620	0,261

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *12 %*

Tab. A.8: Výsledky komprese souboru **sigmod.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	478 416	97 642	0,204
GZIP – 1 – 32KB – 258		85 246	0,178
GZIP – 9 – 32KB – 8		87 728	0,183
GZIP – 9 – 32KB – 258		77 220	0,161
BZIP2 – 1 – 100KB – ×		63 571	0,133
BZIP2 – 1 – 900KB – ×		48 792	0,102
BZIP2 – 9 – 100KB – ×		63 571	0,133
BZIP2 – 9 – 900KB – ×		48 792	0,102
PPM – 1 – 1MB – 2		57 153	0,119
PPM – 1 – 1MB – 16		57 153	0,119
PPM – 1 – 256MB – 2		57 153	0,119
PPM – 1 – 256MB – 16		57 153	0,119
PPM – 9 – 1MB – 2		42 574	0,089
PPM – 9 – 1MB – 16		42 574	0,089
PPM – 9 – 256MB – 2		42 574	0,089
PPM – 9 – 256MB – 16		**42 574**	**0,089**
LZMA – 1 – 64KB – 8		80 856	0,169
LZMA – 1 – 64KB – 273		71 484	0,149
LZMA – 1 – 64MB – 8		75 014	0,157
LZMA – 1 – 64MB – 273		64 920	0,136
LZMA – 9 – 64KB – 8		76 606	0,160
LZMA – 9 – 64KB – 273		65 270	0,136
LZMA – 9 – 64MB – 8		71 217	0,149
LZMA – 9 – 64MB – 273		58 340	0,122
XMILL – 1 – × – ×		69 208	0,145
XMILL – 9 – × – ×		57 551	0,120

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *11,5 %*

Tab. A.9: Výsledky komprese souboru **swissprot.xml**

Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	114 820 211	16 756 999	0,146
GZIP – 1 – 32KB – 258		15 019 985	0,131
GZIP – 9 – 32KB – 8		15 679 952	0,137
GZIP – 9 – 32KB – 258		13 305 216	0,116
BZIP2 – 1 – 100KB – ×		13 119 685	0,114
BZIP2 – 1 – 900KB – ×		8 724 523	0,076
BZIP2 – 9 – 100KB – ×		13 104 015	0,114
BZIP2 – 9 – 900KB – ×		8 718 968	0,076
PPM – 1 – 1MB – 2		11 727 795	0,102
PPM – 1 – 1MB – 16		11 727 795	0,102
PPM – 1 – 256MB – 2		11 727 795	0,102
PPM – 1 – 256MB – 16		11 727 795	0,102
PPM – 9 – 1MB – 2		6 180 732	0,054
PPM – 9 – 1MB – 16		6 180 732	0,054
PPM – 9 – 256MB – 2		6 180 732	0,054
PPM – 9 – 256MB – 16		6 180 732	0,054
LZMA – 1 – 64KB – 8		13 483 174	0,117
LZMA – 1 – 64KB – 273		11 343 279	0,099
LZMA – 1 – 64MB – 8		10 239 699	0,089
LZMA – 1 – 64MB – 273		7 780 356	0,068
LZMA – 9 – 64KB – 8		12 706 333	0,111
LZMA – 9 – 64KB – 273		10 082 934	0,088
LZMA – 9 – 64MB – 8		9 742 381	0,085
LZMA – 9 – 64MB – 273		**6 055 654**	**0,053**
XMILL – 1 – × – ×		10 576 948	0,092
XMILL – 9 – × – ×		8 253 277	0,072

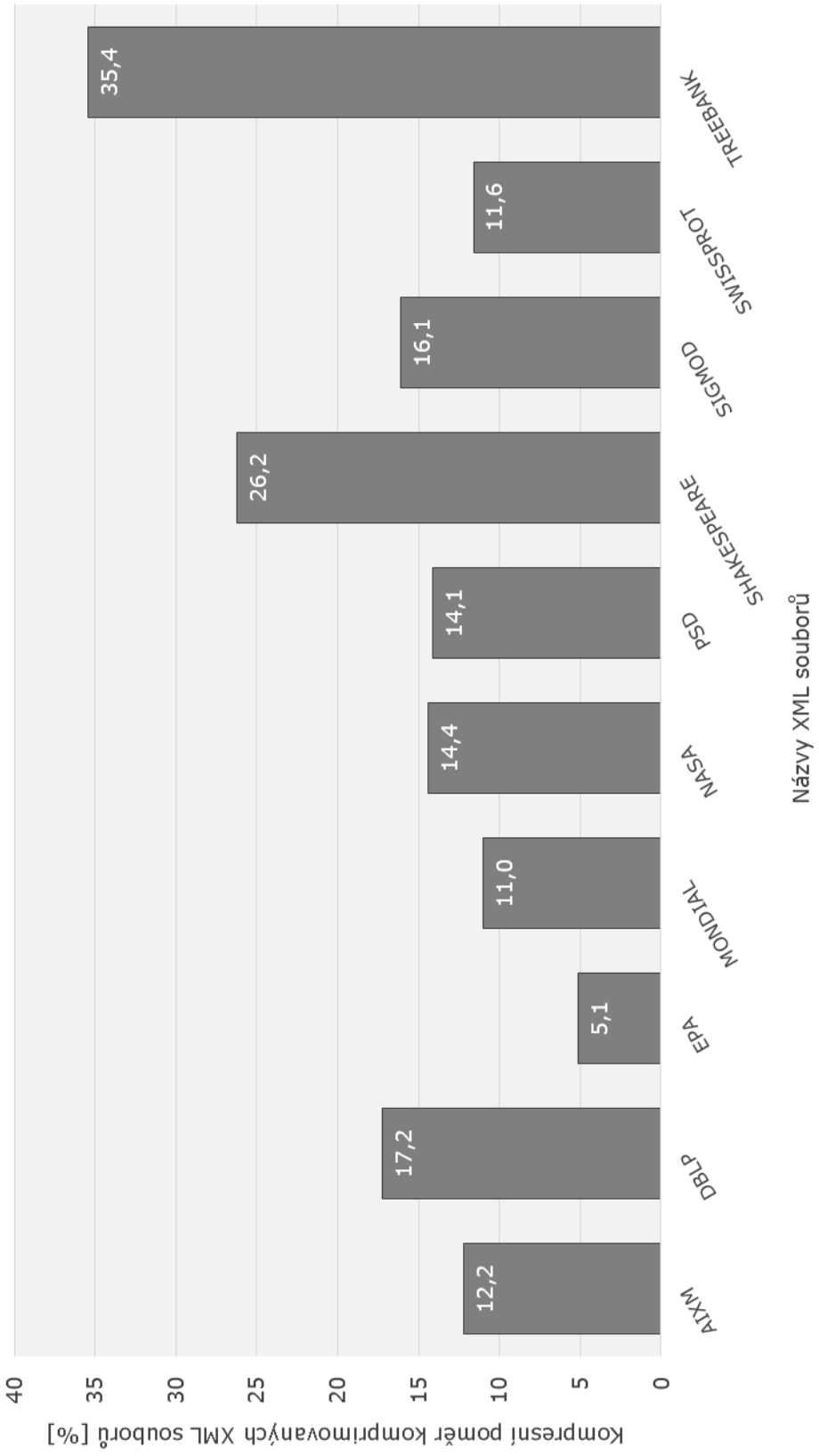
Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *9,3 %*

Tab. A.10: Výsledky komprese souboru **treebank.xml**

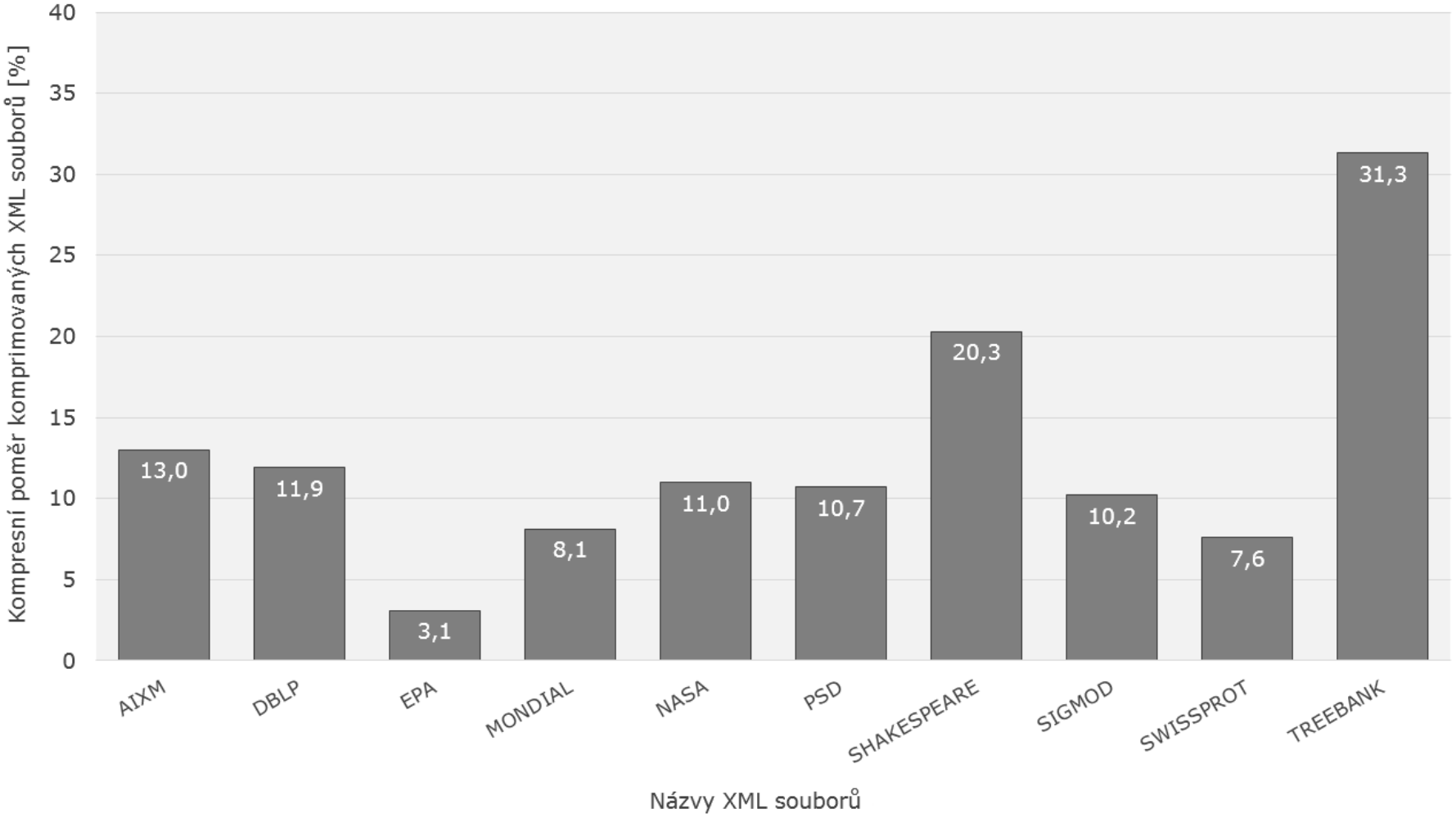
Kódové označení kompresní metody	**Původní velikost [B]**	**Komprimovaná velikost [B]**	**Kompresní poměr [–]**
GZIP – 1 – 32KB – 8	86 082 517	37 017 858	0,430
GZIP – 1 – 32KB – 258		34 602 229	0,402
GZIP – 9 – 32KB – 8		34 093 391	0,396
GZIP – 9 – 32KB – 258		30 463 568	0,354
BZIP2 – 1 – 100KB – ×		28 266 922	0,328
BZIP2 – 1 – 900KB – ×		26 957 956	0,313
BZIP2 – 9 – 100KB – ×		28 244 755	0,328
BZIP2 – 9 – 900KB – ×		26 947 394	0,313
PPM – 1 – 1MB – 2		28 987 933	0,337
PPM – 1 – 1MB – 16		28 987 933	0,337
PPM – 1 – 256MB – 2		28 987 933	0,337
PPM – 1 – 256MB – 16		28 987 933	0,337
PPM – 9 – 1MB – 2		27 319 445	0,317
PPM – 9 – 1MB – 16		27 319 445	0,317
PPM – 9 – 256MB – 2		27 319 445	0,317
PPM – 9 – 256MB – 16		27 319 445	0,317
LZMA – 1 – 64KB – 8		34 921 839	0,406
LZMA – 1 – 64KB – 273		31 866 709	0,370
LZMA – 1 – 64MB – 8		37 768 481	0,439
LZMA – 1 – 64MB – 273		33 991 440	0,395
LZMA – 9 – 64KB – 8		33 040 928	0,384
LZMA – 9 – 64KB – 273		28 587 447	0,332
LZMA – 9 – 64MB – 8		33 341 963	0,387
LZMA – 9 – 64MB – 273		27 695 471	0,322
XMILL – 1 – × – ×		27 218 710	0,316
XMILL – 9 – × – ×		**25 874 617**	**0,301**

Rozptyl mezi nejlepším a nejhorším kompresním poměrem: *13,8 %*

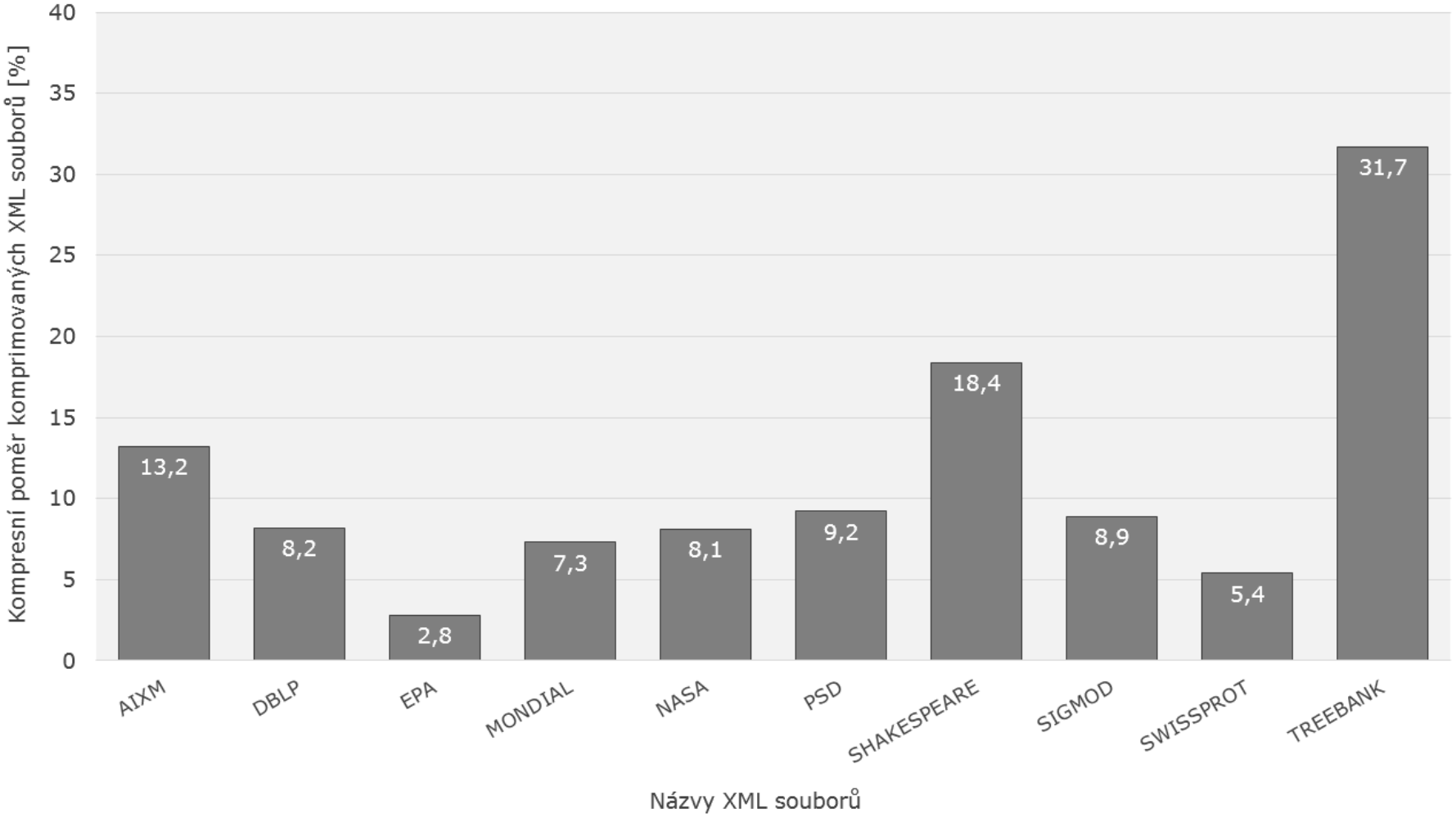
B GRAFY



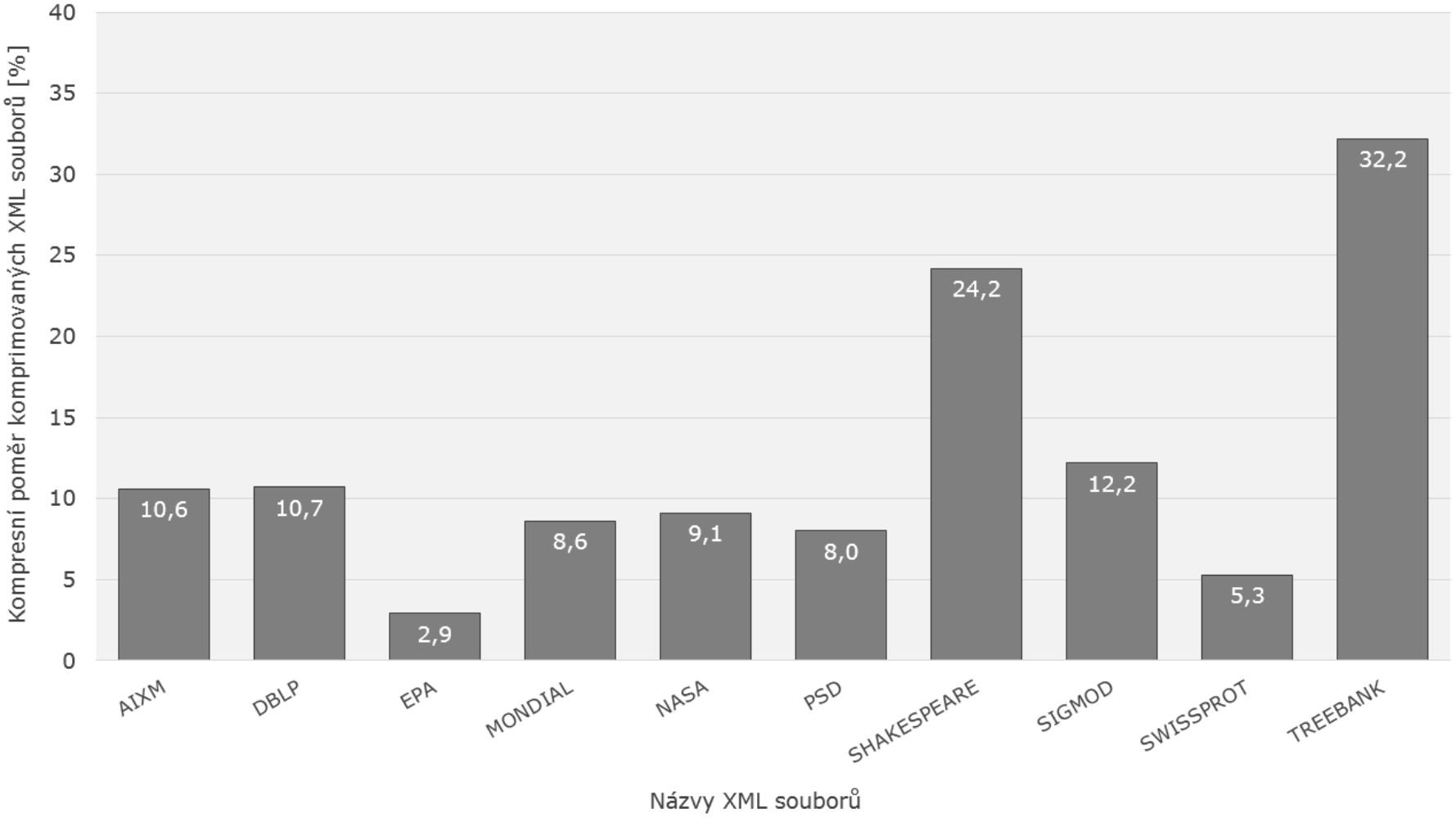
Obr. B.1: Kompresní poměry souborů při použité metodě GZIP – 9 – 32KB – 258



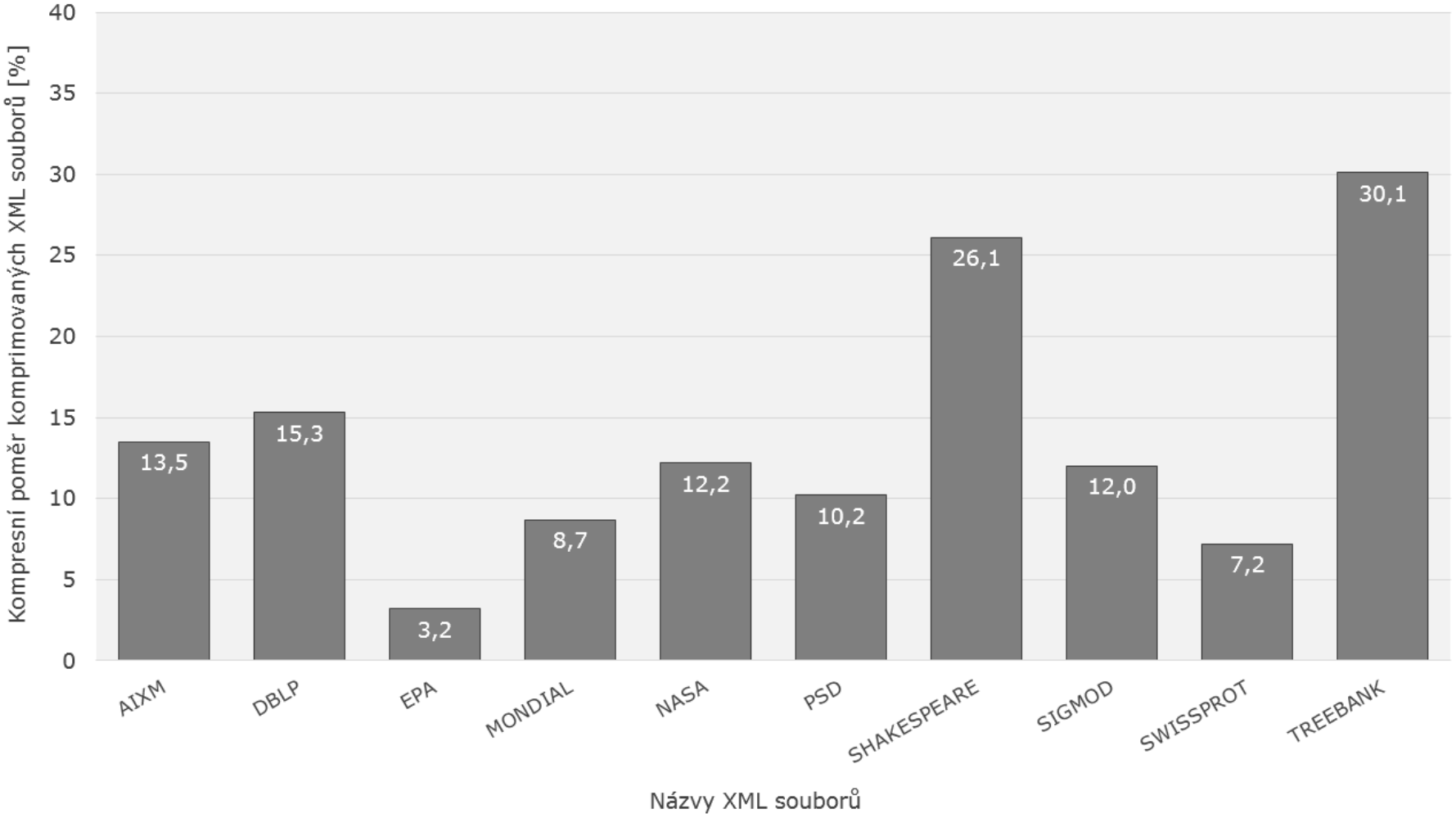
Obr. B.2: Kompresní poměry souborů při použité metodě BZIP2 – 9 – 900KB – ×



Obr. B.3: Kompresní poměry souborů při použité metodě PPM – 9 – 256MB – 16



Obr. B.4: Kompresní poměry souborů při použité metodě LZMA – 9 – 64MB – 273



Obr. B.5: Kompresní poměry souborů při použité metodě XMILL−9−×−×

C DŮLEŽITÉ ČÁSTI KÓDU

Program C.1: Načítání vstupních dat do prvku `listBox`

```
StreamReader sr = new StreamReader(cestaXml, Encoding.UTF8);
sr.BaseStream.Seek(0, SeekOrigin.Begin);
while (sr.Peek() > -1)
{
 string str = sr.ReadLine();
 listBox1.Items.Add(str);
}
sr.Close();
listBox1.SetSelected(1, true);
```

Program C.2: Parsování vstupních dat a plnění prvku `treeView`

```
XmlTextReader reader = new XmlTextReader(cestaXml);
reader.WhitespaceHandling = WhitespaceHandling.None;
string readerName = "";
bool start_node = false;
int depth = 0;
TreeNode actual_node = null;
TreeNode root = null;
TreeNode attribute = null;
TreeNode new_node = null;
bool empty = false;
while (reader.Read())
{
 switch (reader.NodeType)
 {
  case XmlNodeType.Element:
   {
    readerName = reader.Name;
    empty = reader.IsEmptyElement;

    if (!start_node)
    {
     start_node = true;
     root = this.treeView1.Nodes.Add(readerName);
     Associate(root, reader.LineNumber);
     root.SelectedImageIndex = 0;
```

```
26          root.ImageIndex = 0;
27          continue;
28        }
29        depth = reader.Depth;
30
31        if (reader.IsStartElement() && depth == 1)
32        {
33          actual_node = root.Nodes.Add(reader.Name);
34          Associate(actual_node, reader.LineNumber);
35        }
36        else
37        {
38          TreeNode parent = actual_node;
39          actual_node = parent.Nodes.Add(reader.Name);
40          Associate(actual_node, reader.LineNumber);
41        }
42
43        actual_node.SelectedImageIndex = 1;
44        actual_node.ImageIndex = 1;
45
46        for (int i = 0; i < reader.AttributeCount; i++)
47        {
48          reader.MoveToAttribute(i);
49          string rValue = reader.Value.Replace("\r\n", " ");
50          attribute = actual_node.Nodes.Add(reader.Name);
51          Associate(attribute, reader.LineNumber);
52
53          attribute.SelectedImageIndex = 1;
54          attribute.ImageIndex = 1;
55          TreeNode tmp = attribute.Nodes.Add(rValue);
56          tmp.SelectedImageIndex = 2;
57          tmp.ImageIndex = 2;
58          Associate(tmp, reader.LineNumber);
59
60          attribute.SelectedImageIndex = 2;
61          attribute.ImageIndex = 2;
62
63        }
64
65        if (empty)
66          actual_node = actual_node.Parent;
```

```
67      }
68      break;
69    case XmlNodeType.Text:
70      {
71       string rValue = reader.Value.Replace("\r\n", " ");
72       new_node = actual_node.Nodes.Add(rValue);
73       Associate(new_node, reader.LineNumber);
74       new_node.SelectedImageIndex = 3;
75       new_node.ImageIndex = 3;
76      }
77      break;
78    case XmlNodeType.EndElement:
79      actual_node = actual_node.Parent;
80      break;
81   }
82 }
83 reader.Close();
84 root.Expand();
```

Program C.3: Příprava k procesu dekomprese pomocí LZMA

```
1  string[] fragmentslzma = nactenilzma.FileName.Split('.');
2  fragmentslzma = fragmentslzma.Where(path => path != ↩
       fragmentslzma[fragmentslzma.Length - 1]).ToArray();
3  string mezivysledeklzma = string.Join(".", fragmentslzma);
4  vysledeklzma = mezivysledeklzma + ".xml";
5
6  Process launch = new Process();
7  launch.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
8  launch.StartInfo.FileName = @"AIXMcompress\lzma.exe";
9  launch.StartInfo.Arguments = "d " + nactenilzma.FileName + "↩
       " + vysledeklzma;
10 launch.Start();
11 launch.WaitForExit();
```

Program C.4: Vytváření soupisu nalezených elementů

```
1  if (xmlLoaded == true || notamLoaded == true)
2  {
3   NotVisible();
4
5   if (xmlLoaded == true)
6   {
7    for (int x = 0; x <= listBox1.Items.Count - 1; x++)
8    {
9     listBox4.Items.Add(listBox1.Items[x]);
10   }
11
12   treeView5.BeginUpdate();
13
14   events = new TreeNode("Events");
15   treeView2.Nodes.Add(events);
16   runways = new TreeNode("Runways");
17   treeView2.Nodes.Add(runways);
18   taxiwayElements = new TreeNode("Taxiway Elements");
19   treeView2.Nodes.Add(taxiwayElements);
20   runwayElements = new TreeNode("Runway Elements");
21   treeView2.Nodes.Add(runwayElements);
22   apronElements = new TreeNode("Apron Elements");
23   treeView2.Nodes.Add(apronElements);
24   verticalStructures = new TreeNode("Vertical Structures");
25   treeView2.Nodes.Add(verticalStructures);
26   AIXM5Airspaces = new TreeNode("AIXM 5 Airspaces");
27   treeView2.Nodes.Add(AIXM5Airspaces);
28   airspaces = new TreeNode("Airspaces");
29   treeView2.Nodes.Add(airspaces);
30   runwayDirectionLightSystems = new TreeNode("Runway ↩
        Direction Light Systems");
31   treeView2.Nodes.Add(runwayDirectionLightSystems);
32   airportHeliports = new TreeNode("Airport Heliports");
33   treeView2.Nodes.Add(airportHeliports);
34   taxiways = new TreeNode("Taxiways");
35   treeView2.Nodes.Add(taxiways);
36   aprons = new TreeNode("Aprons");
37   treeView2.Nodes.Add(aprons);
38   navaids = new TreeNode("Navaids");
```

```
39  treeView2.Nodes.Add(navaids);
40  geoBorders = new TreeNode("GeoBorders");
41  treeView2.Nodes.Add(geoBorders);
42  specialDates = new TreeNode("Special Dates");
43  treeView2.Nodes.Add(specialDates);
44  arrivalFeederLegs = new TreeNode("Arrival Feeder Legs");
45  treeView2.Nodes.Add(arrivalFeederLegs);
46  instrumentApproachProcedures = new TreeNode("Instrument ↩
       Approach Procedures");
47  treeView2.Nodes.Add(instrumentApproachProcedures);
48  departureLegs = new TreeNode("Departure Legs");
49  treeView2.Nodes.Add(departureLegs);
50  standardInstrumentDepartures = new TreeNode("Standard ↩
       Instrument Departures");
51  treeView2.Nodes.Add(standardInstrumentDepartures);
52  runwayDirections = new TreeNode("Runway Directions");
53  treeView2.Nodes.Add(runwayDirections);
54  routes = new TreeNode("Routes");
55  treeView2.Nodes.Add(routes);
56  airspaceAuthorities = new TreeNode("Airspace Authorities")↩
       ;
57  treeView2.Nodes.Add(airspaceAuthorities);
58  airportHeliportCollocations = new TreeNode("Airport ↩
       Heliport Collocations");
59  treeView2.Nodes.Add(airportHeliportCollocations);
60  airportClearanceServices = new TreeNode("Airport Clearance↩
        Services");
61  treeView2.Nodes.Add(airportClearanceServices);
62  airportSuppliesServices = new TreeNode("Airport Supplies ↩
       Services");
63  treeView2.Nodes.Add(airportSuppliesServices);
64  passengerServices = new TreeNode("Passenger Services");
65  treeView2.Nodes.Add(passengerServices);
66  fireFightingServices = new TreeNode("Fire Fighting ↩
       Services");
67  treeView2.Nodes.Add(fireFightingServices);
68  units = new TreeNode("Units");
69  treeView2.Nodes.Add(units);
70  searchRescueServices = new TreeNode("Search Rescue ↩
       Services");
71  treeView2.Nodes.Add(searchRescueServices);
```

```
72    groundTrafficControlServices = new TreeNode("Ground ↩
          Traffic Control Services");
73    treeView2.Nodes.Add(groundTrafficControlServices);
74    airTrafficManagementServices = new TreeNode("Air Traffic ↩
          Management Services");
75    treeView2.Nodes.Add(airTrafficManagementServices);
76    angleIndications = new TreeNode("Angle Indications");
77    treeView2.Nodes.Add(angleIndications);
78    approachLightingSystems = new TreeNode("Approach Lighting ↩
          Systems");
79    treeView2.Nodes.Add(approachLightingSystems);
80    touchDownLiftOffLightSystems = new TreeNode("Touch Down ↩
          Lift Off Light Systems");
81    treeView2.Nodes.Add(touchDownLiftOffLightSystems);
82    taxiwayLightSystems = new TreeNode("Taxiway Light Systems"↩
          );
83    treeView2.Nodes.Add(taxiwayLightSystems);
84    dmes = new TreeNode("DMEs");
85    treeView2.Nodes.Add(dmes);
86    vors = new TreeNode("VORs");
87    treeView2.Nodes.Add(vors);
88    localizers = new TreeNode("Localizers");
89    treeView2.Nodes.Add(localizers);
90    tacans = new TreeNode("TACANs");
91    treeView2.Nodes.Add(tacans);
92    markerBeacons = new TreeNode("Marker Beacons");
93    treeView2.Nodes.Add(markerBeacons);
94    routeSegments = new TreeNode("Route Segments");
95    treeView2.Nodes.Add(routeSegments);
96    designatedPoints = new TreeNode("Designated Points");
97    treeView2.Nodes.Add(designatedPoints);
98    runwayCentrelinePoints = new TreeNode("Runway Centreline ↩
          Points");
99    treeView2.Nodes.Add(runwayCentrelinePoints);
100   glidepaths = new TreeNode("Glidepaths");
101   treeView2.Nodes.Add(glidepaths);
102   holdingPatterns = new TreeNode("Holding Patterns");
103   treeView2.Nodes.Add(holdingPatterns);
104   touchDownLiftOffSafeAreas = new TreeNode("Touch Down Lift ↩
          Off Safe Areas");
105   treeView2.Nodes.Add(touchDownLiftOffSafeAreas);
```

```
106   guidanceLines = new TreeNode("Guidance Lines");
107   treeView2.Nodes.Add(guidanceLines);
108   runwayProtectAreas = new TreeNode("Runway Protect Areas");
109   treeView2.Nodes.Add(runwayProtectAreas);
110   checkpointVORs = new TreeNode("Checkpoint VORs");
111   treeView2.Nodes.Add(checkpointVORs);
112   aircraftStands = new TreeNode("Aircraft Stands");
113   treeView2.Nodes.Add(aircraftStands);
114   aircraftGroundServices = new TreeNode("Aircraft Ground ↩
         Services");
115   treeView2.Nodes.Add(aircraftGroundServices);
116   specialNavigationStations = new TreeNode("Special ↩
         Navigation Stations");
117   treeView2.Nodes.Add(specialNavigationStations);
118   radioFrequencyAreas = new TreeNode("Radio Frequency Areas"↩
         );
119   treeView2.Nodes.Add(radioFrequencyAreas);
120   aeronauticalGroundLights = new TreeNode("Aeronautical ↩
         Ground Lights");
121   treeView2.Nodes.Add(aeronauticalGroundLights);
122   standardLevelTables = new TreeNode("Standard Level Tables"↩
         );
123   treeView2.Nodes.Add(standardLevelTables);
124   standardLevelColumns = new TreeNode("Standard Level ↩
         Columns");
125   treeView2.Nodes.Add(standardLevelColumns);
126
127   CopyToTreeView1();
128
129   TreeNodeCollection aixmNodes = treeView2.Nodes;
130   foreach (TreeNode an in aixmNodes)
131   {
132    if (an.Nodes.Count != 0)
133    {
134     treeView5.Nodes.Add((TreeNode)an.Clone());
135    }
136   }
137
138   treeView5.Sort();
139   treeView5.EndUpdate();
140  }
```

```
141
142  if (notamLoaded == true)
143  {
144
145   int konec_notam = -1;
146   string slovo_notam = "event:Event gml:id";
147   string koncove_slovo_notam = "</event:Event>";
148   int pocatek = 0;
149
150   for (int x = 0; x <= listBox3.Items.Count; x++)
151   {
152    int pozice_notam = StartIndex(listBox3, slovo_notam, ↩
        pocatek);
153    if (pozice_notam != -1)
154    {
155     konec_notam = EndIndex(listBox3, koncove_slovo_notam, ↩
         pozice_notam);
156    }
157    else
158    {
159     konec_notam = -1;
160    }
161    if (konec_notam != -1)
162    {
163     for (int i = pozice_notam; i <= konec_notam; i++)
164     {
165      listBox4.Items.Add(listBox3.Items[i]);
166     }
167     pocatek += konec_notam;
168    }
169   }
170
171   treeView5.BeginUpdate();
172
173   eventsNotam = new TreeNode("Events (NOTAM)");
174   eventsNotam.ForeColor = Color.DarkRed;
175   treeView5.Nodes.Add(eventsNotam);
176
177   CopyToTreeView3();
178
179   treeView5.Sort();
```

```
180   treeView5.EndUpdate();
181  }
182
183  Step2();
184  radioButton8.Checked = true;
185 }
```

Program C.5: Vytváření jednoho z podřízených uzlů

```
1  if (findFunction_tn.Text == "aixm:Airspace")
2  {
3   if (findFunction_tn.FirstNode.Text == "gml:id")
4   {
5    string ident = findFunction_tn.FirstNode.LastNode.Text.↩
        ToString();
6    TreeNode tmp = (TreeNode)findFunction_tn.Clone();
7    TreeNode rootNode = new TreeNode("Airspace => " + ident);
8    airspaces.Nodes.Add(rootNode);
9    status2 = 1;
10   FindInfo(rootNode, tmp);
11  }
12 }
```

Program C.6: Vytváření jednoho z atributů podřízeného uzlu

```
1  if (findInfo_tn.Text == "gco:DateTime")
2  {
3   findInfo_parent.Nodes.Add("> gco:DateTime");
4   findInfo_parent.LastNode.Nodes.Add(findInfo_tn.LastNode.↩
      Text.ToString());
5  }
```

Program C.7: Vytváření jedné z částí náhledu výstupních dat

```
1  string s1 = oz.FirstNode.NextNode.LastNode.Text.ToString();
2  string s2 = " gml:id=\"";
3  string s3 = oz.FirstNode.LastNode.Text.ToString();
4  string slovo = s1 + s2 + s3;
5  string koncove_slovo = "</" + oz.FirstNode.NextNode.LastNode↩
     .Text.ToString() + ">";
```

```
int pozice = StartIndex(listBox4, slovo, 0);
if (pozice != -1)
{
 konec = EndIndex(listBox4, koncove_slovo, pozice);
}
else
{
 konec = -1;
}
if (konec != -1)
{
 for (int i = pozice; i <= konec; i++)
 {
  listBox2.Items.Add(listBox4.Items[i]);
 }
 status = 1;
}
```

Program C.8: Typová funkce pro výběr úseků dat

```
private int StartIndex(ListBox lb, string searchString, int ↩
   startIndex)
{
 for (int i = startIndex; i < lb.Items.Count; ++i)
 {
  string lbString = lb.Items[i].ToString();
  if (lbString.Contains(searchString))
   return i;
 }
 return -1;
}
```

Program C.9: Uložení XML výstupu před bezeztrátovou kompresí

```
using (var sw = new StreamWriter(ulozeni.FileName, false, ↩
   Encoding.UTF8))
 foreach (var item in listBox2.Items)
 {
  sw.Write(item.ToString() + Environment.NewLine);
 }
```