Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Experimentální medicínský informační systém – zpracování semistrukturovaných dat (XML)

Plzeň, 2013 Petr Žák

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. 5. 2013

Petr Žák

Abstrakt

Tématem této práce jsou semistrukturovaná data a využití značkovacího jazyka XML pro jejich zpracování a ukládání ve zdravotnických informačních systémech. Práce obsahuje dvě hlavní části – teoretickou a praktickou. Cílem teoretické části bylo vytvořit komplexní úvod do medicínské informatiky, informačních systémů a dat. Praktická část byla zaměřena na design a implementaci softwaru, který bude schopen zpracovávat data uložená v XML souborech. Přesněji řečeno, úkolem bylo implementovat několik základních funkcí pro zpracování těchto dat.

Abstract

Experimental medical information system – processing of semistructured data (XML):

Topic of this thesis are semistructured data and usage of XML markup language for their processing and storage in medical information systems. This work contains two main sections – theoretical part and practical part. The goal of the theoretical part was to create complex introduction into medical informatics, information systems and data. Practical part was focused on design and implementation of software, which can be used to process medical data stored in XML files. More specifically, the goal was to implement several basic functions for processing of those data.

Poděkování

Rád bych tímto poděkoval doc. Dr. Ing Janě Klečkové za její čas, řadu konzultací a mnoho odborných rad, kterými mi byla velice nápomocna při tvorbě této práce. Zároveň bych rád poděkoval svojí rodině za materiální i duševní podporu, zvláště během mnoha vypjatých okamžiků.

Obsah

1. Úvod

Informatika je v současné době jedním z nejrychleji se rozvíjejících oborů lidské činnosti. Poznatky a řešení, které nabízí, jsou využívány v mnoha dalších oborech od ekonomických, přes technické a strojírenské až po humanitní a společenskovědní. Žádná větší organizace by v dnešní době prakticky nemohla fungovat bez kvalitního informačního řešení, ať již se jedná o soukromou firmu, státní úřad či nemocnici. A právě posledně jmenovaným případem, tedy využitím informatiky ve zdravotnictví, konkrétně zdravotnickými informačními systémy a daty jimi zpracovávanými se bude zabývat tato práce.

Hlavním tématem práce jsou semistrukturovaná data a využití značkovacího jazyka XML při jejich zpracování a ukládání v prostředí zdravotnických informačních systémů. Cílem teoretické části práce je vytvořit co nejkomplexnější teoretický úvod do prostředí medicínské informatiky, medicínských informačních systémů a zdravotnických dat.

Cílem praktické části bylo vytvoření softwaru pro částečné zpracování textových zdravotnických zpráv, uložených pomocí jazyka XML dle standardu RDF. Konkrétním úkolem bylo navrhnout a zrealizovat několik základních funkcí, na nichž bude v budoucnu možné postavit pokročilejší analytické funkce, jako je například automatická kategorizace zpráv dle společných znaků. Ve finální verzi byly implementovány tři základní funkce – předzpracování, frekvenční analýza a vyhledávání.

2. Data, Informace, Znalosti

Na začátku je vhodné uvést definici tří základních pojmů, které budu hojně využívat a které jsou sice všeobecně známé, nicméně rozdíl mezi nimi nemusí být zcela zřejmý a pro účely této práce je přesné rozlišení nezbytné. Ani v jednom z případů sice neexistuje jediná všeobecně přijímaná definice, většina autorů se však více či méně shoduje a význam jednotlivých pojmů je zřejmý.

2.1 Data

Asi nejsrozumitelnější, nejjednodušší a přitom stále přesná definice pojmu data je, že se jedná o posloupnost znaků. Zpravidla samozřejmě písmen, číslic a případných dalších znamének, nicméně toto není podmínkou, prakticky neexistují žádná omezení pro obsah. Je důležité zmínit, že data jako taková s sebou nemusejí nést údaje o jejich významu či způsob, jakým mají být interpretována. [1] Jedná se čistě o popis, lépe řečeno reprezentaci reálné skutečnosti, která může být přenášena, uchována či zpracována. [2] Příklad dat z medicínského oboru může vypadat třeba takto:

22, 190, 75, A RH POS

Je zřejmé, že tato data nám poskytují popis nějakého reálného objektu v takové formě, ve které je možné jej např. uložit do databáze či zaslat elektronickou poštou. Nicméně sama o sobě nám nejsou příliš užitečná, protože se můžeme maximálně dohadovat, co znamenají a jakým způsobem je interpretovat.

2.2 Informace

Jak lze odvodit z předchozího odstavce, informace jsou data doplněná o jejich význam, jinak řečeno *data, která mají smysl*. [2] Laicky řečeno, jsou-li nám data k něčemu užitečná, známe-li jejich význam a dovedeme z nich tedy získat nějaké poznatky či údaje, jsou pro nás tato data nositelem informace či informací. Je tedy zřejmé, že data, která jsou pro někoho (či něco) informací, mohou být pro někoho (či něco) jiného obyčenými daty. [1]

Například hash řetězec ec457d0a974c48d5685a7efa03d137dc8bbde7e3 je pro člověka obyčejnými daty, které mu neposkytují žádnou informaci, a to i v případě, že ví, že se jedná o výsledek hashovací funkce. Oproti tomu pro počítač se jedná o plnohotnotnou informaci. Naopak příkladem dat, jež při interpretaci člověkem nesou informaci, je toto rozšíření předchozího příkladu:

Věk 22 let, výška 190 cm, váha 75 kg, krevní skupina A RH pozitivní

Zde je jasné, že jde o základní údaje o nějakém člověku, z těchto dat se bez problémů dozvíme nějaké nové údaje, které nám mohou být užitečné – jsou tedy pro nás nositelem informace.

2.3 Znalosti

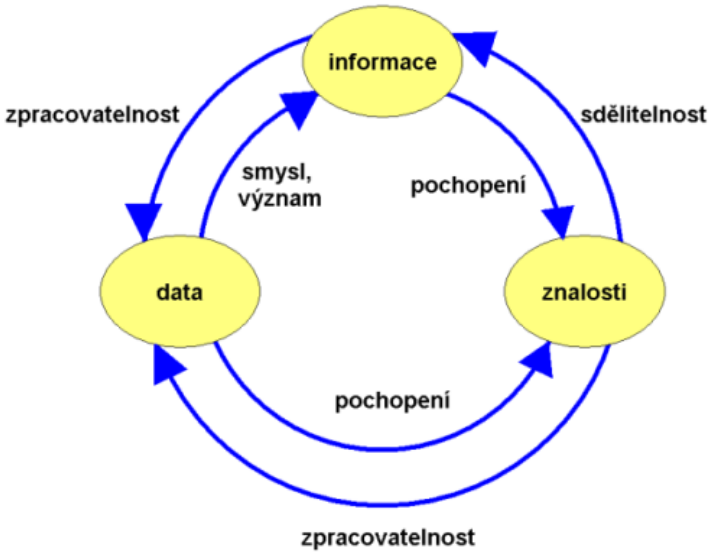
Znalost je informace, rozšířená o schopnost ji aktivně využít k rozhodování či řešení problémů. Tedy *to, co jednotlivec vlastní (ví) po osvojení dat a informací a po jejich začlenění do souvislostí.* [1] [2] Různí autoři se na této definici víceméně shodují:

- *Znalost je organizovaná informace využitelná k řešení problémů. (Woolf, 1990)*
- *Znalost je informace, která prošla uspořádáním a analýzou, aby se stala srozumitelnou a použitelnou k řešení problémů nebo k rozhodování. (Turban, 1992)*
- *Znalost spočívá v uvažování o informacích a datech za účelem umožnění aktivního jednání, řešení problémů, rozhodování, učení se a vyučování. (Beckman, 1997)*

Rozšíříme-li tedy opět náš předchozí příklad, znalostí je třeba to, že dle BMI je 75 kg u 190 cm vysokého člověka normální hmotnost, nebo že při transfuzi může dostat i krev skupiny A negativní.

2.4 Vztah mezi daty, informacemi a znalostmi

Vztah mezi těmito třemi formami je dobře znatelný z obrázku 2.1.



Obr. 2.1 – vztah mezi daty, informacemi a znalostmi [2]

Zároveň je z obrázku 2.1 zřejmé, že údaje lze mezi formami obousměrně „přesouvat“. Z předchozího textu by se mohlo zdát, že čím vyšší formu údajů máme, tím lépe a nemá smysl uchovávat a používat např. data ve chvíli, kdy máme znalosti. To však není pravda, protože pokud chceme údaje uchovat například v databázi, potřebujeme je mít ve formě dat – znalosti do databáze uložit prakticky nelze. Záleží tedy na konkrétní situaci a potřebách, která z forem je nejlepší.

3. Informatika a informační systémy

Lze předpokládat, že tyto dva pojmy bude drtivá většina čtenářů dobře znát a bude vědět, co si pod nimi představit, přesto jsou pro úplnost v rychlosti uvedeny a definovány.

3.1 Informatika

Obecně známá a přijatá definice informatiky zní, že se jedná o vědu či lidskou činnost, zabývající se získáváním, zpracováním a uchováváním informací za využití počítače. Dle jiné definice je informatika *obor lidské činnosti, který se zabývá vytvářením obrazů o reálném světě.* [3] Dle Jiřího Cejpka je počítač a jeho technické a programové vybavení *předmětem a východiskem informatiky*. [4] Existuje mnoho podoborů informatiky, jako je například obor informačních technologií, studující fungování počítačů po technické stránce, geomatika, zabývající se vývojem softwaru pro geografické potřeby a konečně také lékařská informatika, kterou se tato práce bude zabývat především a proto jí bude věnována přesnější a obsáhlejší definice později.

3.2 Informační systémy

Informační systém, zkráceně IS, lze obecně definovat jako systém, jehož účelem je sběr, ukládání, zpracování, přenos a prezentace dat, potřebných (či užitečných) pro rozhodování a řízení. [5] Jiná, nicméně velmi podobná definice zní: *Informační systém je systém, umožňující účelné uspořádání sběru, uchování, zpracování a poskytování informací.* [6]

Pravděpodobně nejznámější kategorií IS jsou systémy určené pro soukromou sféru, tedy podnikové informační systémy typu ERP, SCM, CRM či BI – například různé varianty systému SAP, Oracle Business Suite či systémy od společnosti INFOR Global Solutions. [7] Tato práce se však bude zabývat informačními systémy používanými ve zdravotnictví, kterým je rovněž věnováno více prostoru dále v textu.

4. Lékařská informatika

Lékařská informatika je obor, který v sobě kombinuje dva jiné vědní obory – informatiku a lékařství (medicínu). Volný a zkrácený překlad relativně rozsáhlé definice dle University of Illinois zní takto:

„Zdravotní informatika je věda, která v sobě spojuje zdravotní péči a informační technologie a integruje veškeré části péče o pacientovo zdraví, včetně klinické péče, ošetřovatelství, lékárenství a veřejného zdraví. Zaměřuje se nejen na implementaci a optimalizaci informačních technologií pro podporu lékařské praxe, ale zároveň též vytváří infrastrukturu, která umožňuje tok kritických informací mezi všemi články péče o pacienta.“ [8]

Zahrnuje v sobě tedy veškeré prostředky, určené pro podporu práce s informacemi ve zdravotnickém sektoru, od datových úložišť po systémy pro podporu rozhodování. Uplatňuje se prakticky ve všech oborech medicíny, od intenzivní péče, přes lékárenství či zubní praxi až po výzkum a vývoj.

4.1 Dělení lékařské informatiky

Lékařskou informatiku lze rozdělit na tři základní oblasti: [9]

4.1.1 Obecná lékařská informatika

Jedná se o část, která je z velké části shodná s klasickou informatikou, respektive která obsahuje především nespecializované komponenty. Tedy například balíky kancelářských aplikací (textové editory, tabulkové procesory…), webové prezentace a různé on-line systémy (informace o lékařích a nemocnicích, registrační a rezervační aplikace…), sítě a komunikační systémy (e-mail, IM, kolaborační nástroje…) či databázové systémy. [9]

Zařazení veškerých těchto nástrojů, systémů a aplikací do lékařské informatiky se může zdát lehce nesmyslné, nicméně je naprosto v pořádku, protože všechny tyto části jsou samozřejmě v medicíně hojně využívané.

4.1.2 Informační zdroje

Tato oblast se dále dělí na dvě hlavní podoblasti, z nichž do první spadají komponenty týkající se vědeckých zdravotnických informací a využití informačních technologií k jejich získávání, zpracování a ukládání. Druhá pak obsahuje klasifikační systémy, což jsou zjednodušeně řečeno aplikace určené pro podporu standardizace lékařské terminologie a následné zjednodušení a zefektivnění komunikace. [9]

Mezi části pro **podporu informací** je možné zařadit například i Internet, jeho vyhledávače a obrovské množství informací, které je na něm volně k dispozici. Hlavní tíhu však v lékařství samozřejmě nesou specializované systémy, především databáze informací z akademických periodik a prací, jako je Medline či Current Content. [9] Pro představu o jejich rozsáhlosti, například Medline v současné době indexuje přes 5600 periodik z oblasti medicíny. [10]

Klasifikační systémy pak jsou, jak už bylo zmíněno, určeny pro zjednodušení a především sjednocení formy psaní lékařských zpráv a záznamů. Jak říká jeden z článků na toto téma, *zapisování lékařských zpráv formou volného textu je velice nehomogenní a není standardizováno. Standardizovaná terminologie by přinesla výhody lékařům, pacientům, administrátorům, softwarovým vývojářů a plátcům. Pomohla by poskytovatelům zdravotnické péče tím, že by poskytovala kompletní a snadno dostupné informace, které náleží k procesu zdravotnictví a to by vyústilo v lepší péči a pacienty. Použití mezinárodních klasifikačních systémů je nezbytným prvním krokem, který umožní interoperabilitu heterogenních elektronických zdravotních záznamů.* [11]

Z tohoto důvodu vznikají kódovací klasifikační systémy jako je MKN (Mezinárodní Klasifikace Nemocí, v originále ICD – International Classification of Diseases) či SNOMED, při jejichž aplikaci je nemožné použít jiné, než schválené termíny (často se namísto termínů používají pouze přiřazené kódy), což omezuje variabilitu vyjadřování a prakticky znemožňuje problémové nazývání stejné věci mnoha různými synonymy. [11]

4.1.3 Zpracování informací a podpora diagnostiky

Poslední a zároveň velmi rozsáhlá kategorie, do které lze zařadit mnoho různých oblastí. Spadají sem: [9]

- Hardware a software pro zpracování informací získaných pomocí lékařských přístrojů a měření. To mohou být biologické signály v podobě EKG a EEG, obraz z RTG nebo CT, sonografie apod.
- Informační systémy různého rozsahu a zaměření, od těch největších určených pro nemocnice či velké sítě lékáren, až po nejmenší pro praktické lékaře či soukromé lékárny.
- Využití počítačových simulací a umělé inteligence.
- Expertní systémy a systémy pro podporu diagnostiky a rozhodování v medicíně, což jsou systémy, které porovnávají dodaná fakta s integrovanou znalostní bází a tímto způsobem vytvářejí expertízu. Přinášejí mnoho výhod, jako je detailní vysvětlení způsobu, jakým expertíza vznikla, rychlost, neměnnost výsledků či schopnost řešit dilemata a konfliktní situace. Přitom mohou být provozovány prakticky na jakémkoli hardwaru a jejich cena je relativně nízká. [9]
- Další oblasti, jako je využití informačních technologií pro vzdělávání lékařského personálu či vývoj nových léčiv.

4.2 Konkrétní využití IT ve zdravotnictví

Nyní se podíváme na popis některých konkrétních praktických oblastí, ve kterých jsou informační technologie aktivně využívány pro zdravotnické účely. [9]

4.2.1 Ukládání a archivace dat a jejich opětovné získávání či vyhledávání

Poměrně rozsáhlá a zároveň velice důležitá oblast. Prakticky celé zdravotnictví, od ambulantního ošetření po lékařský výzkum, je závislé na dostupnosti správných a úplných dat. O případném pacientovi je nutné mít co nejvíce informací, od těch základních (jako je např. krevní skupina) po ty nejdetailnější (kompletní anamnéza atd.) Vývoj či výzkum zase vyžaduje mít co nejvíce vzorků, informace o co největším počtu případů apod., na kterých pak může zakládat svoje dedukce.

Znamená to tedy, že zdravotnictví vyžaduje ukládání poměrně velkého množství dat po dlouhou dobu (relevantní mohou být i desítky let stará data), zachování možnosti

mezi nimi co nejlépe a nejrychleji vyhledávat a to vše samozřejmě s co nejnižšími náklady.

Využívají se jak klasické, zpravidla samozřejmě relační databázové systémy pro ukládání dat v jednotlivých nemocnicích, odděleních či ordinacích, tak centrální, povětšinou mezinárodní databáze (např. již výše zmíněné MEDLINE, MKN, či SNOMED) či databáze obrazových dat PACS (Picture Archiving and Communications Systém – *technologie zdravotnického zobrazování, která poskytuje ekonomické uložení a pohodlný přístup k obrazům z mnoha různých zdrojových zařízení* [12]). Pro opětovné získávání dat jsou pak použity procesy, známé i z jiných oblastí, jako je fulltextové vyhledávání či data mining (*Netriviální proces získávání validních, nových, pochopitelných a potenciálně užitečných informací z dat* [Fayad, 1996]). [9]

4.2.2 Zpracování dat a automatizace operací

Do této oblasti spadá využití IT pro zjednodušení a zpřesnění člověkem definovaných úkonů, které je zpravidla potřeba provádět opakovaně. Jedná se například o analýzy vzorků krve či moči (člověk nastaví parametry, analýza proběhne plně automaticky a výsledky jsou v přehledné formě předány opět člověku) nebo zpracování biologických signálů (tep, krevní tlak apod.; lékař určí maximální či minimální úroveň a přístroj automaticky upozorní na překročení limitů).

Poloautomatické přístroje pro zobrazování v medicíně, jako jsou zařízení pro magnetickou rezonanci (MRI) či počítačovou tomografii (CT), patří do této oblasti rovněž. Snímkování probíhá automaticky, avšak dle lékařem nastavených parametrů, stejně tak interpretace výsledků je úkolem lékaře. [9]

4.2.3 Podpora diagnostiky a rozhodování

Diagnostika a rozhodování jsou samozřejmě především lidskou doménou. Vyžadují schopnost správně zkombinovat velké množství informací s mnoha znalostmi a zkušenostmi a na základě toho dedukovat diagnostické možnosti či rozhodnout o další činnosti.

Informační tehnologie však mohou být nápomocny schopností rychle projít veliké množství dat a nalézt v nich určitý vzorec („pattern recognition“ – automatické přiřazení konkrétních dat do určité třídy na základě podobnosti s ostatními členy této třídy či definicí třídy) či řešit problémy heuristickou analýzou („pokus a omyl“ – počítač nasimuluje co nejvíce možných řešení daného problému a z nich vybere na základě daných kritérií to nejlepší [13]).

Mezi konkrétní příklady se řadí například automatické zpracování a klasifikace biologických systémů či výše zmíněné expertní systémy. [9]

4.2.4 Řízení léčby

V oblasti distribuce léků a obecně provádění léčebných procedur je mnoho prostoru pro lidskou chybu, snaha o její eliminování automatizací alespoň některých činností je tedy zcela logická. Týká se to například automatického dávkování látek v transfuzích, kontroly případných interakcí léčiv před jejich vydáním či nastavení a ovládání radioterapeutických přístrojů. [9]

5. Zdravotnické informační systémy

Jak již bylo uvedeno v úvodu, informační systém je obecně vzato systém, jehož účelem je sběr, ukládání, zpracování, přenos a prezentace dat, potřebných (či užitečných) pro rozhodování a řízení. [5] Tato definice samozřejmě platí i ve zdravotnickém prostředí i přesto, že se jedná o prostředí relativně komplikované a specifické. [9] Stejně tak v systémech samotných můžeme pozorovat řadu podobností se standardními komerčními IS. Zdravotnické informační systémy lze rozdělit do dvou hlavních kategorií.

5.1 Systémy pro zdravotnická zařízení

Do této kategorie lze zařadit velkou část medicínských IS, od těch nejmenších, určených pro pokrytí potřeb např. jedné soukromé lékařské praxe či lékárny, až po ty, které pokrývají potřeby celé sítě zdravotnických zařízení. Jedná se o následující systémy: [5]

- **Nemocniční informační systémy** – pokrývají potřeby celých nemocnic, od přijímací kanceláře, přes lůžková oddělení, JIP, ARO či operační sály až po ekonomickou agendu či styk se zdravotními pojišťovnami. Často v sobě zahrnují i některé (či všechny) další zmíněné systémy. Těmito systémy se bude tato práce podrobněji zabývat dále v textu.
- **Ambulantní informační systémy** – systémy pro samostatné či poliklinické ambulance, např. sada systémů od společnosti TILL CONSULT a. s. (DENTA, PRIVAT, PEDIATR, OPTIK)
- **Laboratorní informační systémy** – správa požadavků (zpravidla umožňující offline i online příjem), vzorků a materiálů, automatická kontrola výsledků, po jejich ověření lékařem pak i generování a odesílání výsledných zpráv, archiv atd. [19] Příkladem je systém INFOLAB. [14]
- **Lékárenské informační systémy** – příjem, skladování a expedice lékárenského zboží. Velice podobné systémům v jiných prodejnách či skladech materiálu, největším rozdílem je důraz na odborné informace o „zboží“. [19] Častá je přímá návaznost na systémy nemocnic, distributorů a pojišťoven. Typickým příklad lékárenského IS je FaRMIS. [15]
- **Manažerské a ekonomické informační systémy** – samozřejmě se zaměřením na provoz zdravotnických zařízení, např. MIS Hospital, ale i klasické systémy, např. účetnictví Účto 2005.
- **Informační systémy pro stravovací provoz** – umožňuje plánování jídelníčků s ohledem na stravovací plány, ordinované diety atd. a automatizuje odesílání požadavků do skladů a kuchyní. Jedná se například o systém GASTRIS. [19]
- **Systémy PACS** – popsány výše v textu, např. MediPictures.

5.2 Systémy ve veřejném zdravotnictví (národní a nadnárodní úroveň)

5.2.1 Nadnárodní informační systémy

Např. již zmíněné klasifikační systémy ICD či SNOMED, nebo databáze periodik Medline či Current Content. Dále mezinárodní systémy obsahující informace o léčivech, jako je ATC či British National Formulary. [16]

5.2.2 Národní systémy

- **Národní zdravotnický informační systém (NZIS)**, který je provozován Ústavem zdravotnických informací a statistiky. Zdrojem informací v tomto systému jsou statistická zjišťování prováděná Ministerstvem Zdravotnictví, resortní zdravotnické registry, výběrová šetření nebo data z Českého statistického úřadu. Výstupem pak jsou ročenky, speciální zprávy a především DPS databáze [16] (***Data Presentation System***, *interaktivní databáze, vyvinutá pod patronací Světové zdravotnické organizace (WHO) v rámci programu EU/WHO Copernicus Care Support Project* [17]).
- **Národní zdravotní registry**, dělené do skupin A (registry hospitalizací či registry týkající se porodnictví), B (onkologický registr, registr týkající se kardiochirurgie, cévní chirurgie, registr kloubních náhrad apod.), a C (registr lékařů a zdravotnických zařízení). [16]
- **Informační systém hygienické služby**, obsahující program EPIDAT, určený *k zajištění povinného hlášení, evidence a analýzy výskytu infekčních nemocí v České republice*. Tento program zároveň spadá i do NZIS. [18] Dále do něj patří další registry a subsystémy, jako je PAN (registr pandemií), PIVO (evidence zdrojů pitné vody) či RPN (registr pohlavních nemocí). [16]
- **Další nezařazené systémy a registry**, jako je transplantační registr, registr osob nesouhlasících s posmrtným odběrem tkání či orgánů či národní systém informací o léčivech. [16]

6. Nemocniční informační systémy

Definice NIS je samozřejmě velmi podobná obecné definici IS a může znít například takto: *NIS je systém informačních technologií pro sběr, zpracování a využití informací ve zdravotnickém zařízení.* Jejich účelem je, stejně jako v případě klasických IS, především *zkvalitnění řídící činnosti, zjednodušení administrativy, urychlení přenosu informace a přenosu a zpracování dat.* [9]

6.1 Součásti NIS

6.1.1 Klinická část

Klinická část obsahuje moduly pro práci s odbornými daty, týkajícími se diagnostiky a léčebného procesu. Je možné ji dále rozdělit na menší části s úzce definovanou funkčností: [5]

- **Lůžková oddělení**, obsahující veškerou standardní agendu týkající se hospitalizovaných pacientů. Tedy jejich základní údaje, anamnézu, výsledky veškerých testů a vyšetření, lékařské a laboratorní zprávy, informace o současném stavu, probíhajících léčbách apod. Modul je zároveň schopen automaticky sestavovat výkazy pro zdravotní pojišťovny či např. propouštěcí zprávy. [19]
- **JIP**, tedy Jednotka intenzivní péče. Tato část doplňuje funkčnost standardního lůžkového oddělení o veškerá data získávaná z monitorovacích zařízení JIP, tedy údaje o krevním tlaku, tepu, dýchací frekvenci, výsledky EKG a EEG měření a další. [19]
- **Odborná ambulance** uchovává zprávy o ambulantním ošetření, další její funkce jsou např. objednávací systém či správa došlé pošty (výsledků z laboratoří atd.) Zároveň umožňuje předávat data modulům lůžkového oddělení či JIP, pokud je pacient po ambulantním ošetření hospitalizován. [19]
- **Nelůžková oddělení** – míněna jsou především pracoviště, vytvářející obrazový výstup, jako jsou RTG, CT, MRI. Zde nacházejí uplatnění výše zmíněné systémy PACS. Problémem v této oblasti je konflikt dvou požadavků, kdy z odborného hlediska je vyžadována maximální možná kvalita obrazových dat, zatímco z hlediska IT je nutno udržet objem dat na co nejrozumnější míře. Řešením je vhodná bezztrátová komprese. [19] I přesto je však nezbytné dostatečně dimenzované síťové připojení a veliké datové úložiště (zpravidla min. stonásobek úložiště běžného IS [5]).

6.1.2 Administrativní část

Administrativní část se mnohem více blíží standardním IS, používaným v komerční sféře. Vzhledem k tomu, že administrativa zdravotnického zařízení se až na určitá specifika značně podobá administrativě jakéhokoli jiného podniku, je to logické. Její moduly jsou: [5]

- **Management** je modul, který, jak je z jeho názvu zřejmé, slouží nejvyššímu vedení nemocnice. Jeho účelem je poskytovat vedení relevantní, přesné a přitom dostatečně jednoduché informace pro podporu strategických, medicínských a organizačních rozhodnutí. Zpravidla je dodává ve formě tabulek, grafů, indexů apod. a umožňuje jejich hierarchické třídění – v první chvíli zobrazí co nejjednodušší a nejzákladnější informace a detailnější, konkrétnější údaje si uživatel může vyžádat, pokud je potřebuje. [19]

- **Výkazy pro zdravotní pojišťovny** mají za cíl shromáždit údaje ze všech ostatních částí systému tak, aby vznikly úplné a správné sestavy ve formátu přijímaném zdravotní pojišťovnou, které jsou následně odeslány k proplacení. [19]
- **Hospodářská správa, evidence majetku, účetnictví a zásobování** se prakticky neliší od modulů ostatních, nemedicínských IS. Jsou určeny ke sledování hmotného, nehmotného a finančního majetku a jejich přírůstků a úbytků. [19]
- **Personální agenda** rovněž stejně jako v ostatních IS obsahuje databázi zaměstnanců a informací o nich, umožňuje výpočet jejich platů, sociálních odvodů atd. [19]

6.2 Výhody NIS

Použití kvalitního a správně implementovaného NIS má samozřejmě pro zdravotnické zařízení řadu přínosů. Prvním z nich je obecné zefektivnění chodu a zlepšení výkonnosti celého zařízení. Další výhodou je aktuální centrální registr a možnost získat okamžitý přehled o pacientech, volných lůžkách či personálu. Pokud je součástí IS i klasifikační systém, značným přínosem je sjednocení způsobu vedení lékařské dokumentace a následné zlepšení její přehlednosti, čitelnosti a jasnosti. Stejně tak IS zjednoduší a zlepší zpracování výkazů pro zdravotní pojišťovny, NZIS a další orgány. V neposlední řadě pak samozřejmě implementace NIS díky jasnému přehledu o nákladech a správě zlepší i manažerské a nákladové řízení nemocnice, mimo jiné např. hospodaření s léky. [5]

7. Data ve zdravotnictví

Dostatek kvalitních dat je ve zdravotnictví naprosto zásadním faktorem. Lékař či zdravotní sestra potřebují co nejkompletnější a nejpřesnější informace o pacientovi, výsledcích probíhající léčby a obecně všech faktorech, které mohou mít vliv na danou pacientovu situaci. Pouze na kvalitních informacích mohou založit svoje následné rozhodnutí a případnou činnost, od sběru dalších dat a získání více informací, přes pokračování v léčbě pro propuštění pacienta.

7.1 Obecná problematika zdravotnických dat

Za nejmenší jednotku dat se ve zdravotnické informatice považuje tzv. elementární datum, což je zpravidla výsledek jednoho pozorování či měření, jako je tělesná teplota, krevní tlak nebo tep. Každé takovéto datum je pak kromě samotné hodnoty – výsledu měření či pozorování charakterizováno třemi parametry, což jsou idetifikace pacienta (jméno, rodné číslo…), pozorovaný parametr (teplota, krevní tlak…) a čas/kalendářní datum provedeného pozorování či měření.

Data přitom mohou být v různém formátu, od prostých textových, přes záznamy zvuků a elektrických signálů až po obrazy a fotografie. Zároveň je nutné uvažovat i tzv. status informací, což je určitý parametr dále charakterizující zjištěný údaj – ten tak může být trvalý (alergie…) či přechodný (horečka…) nebo urgentní (tep pacienta na JIP) či neurgentní (kalendářní datum pravidelné kontroly).

7.2 Požadované vlastnosti zdravotnických dat

Medicínská data lze posuzovat jak z hlediska obecných měřítek kvality dat, [20] tak z hlediska specifického prostředí zdravotnictví. [9]

7.2.1 Relevance

Míra, nakolik data splňují účel, pro který jsou používána. [20] Jinak řečeno informace o tom, zda jsou nám v dané situaci daná data užitečná a pokud ano, tak jak moc. Například v případě pacienta, který má zjevně horečku, je rozhodně relevantním datem přesná tělesná teplota. O něco méně relevantní jsou pak údaje jako krevní tlak nebo věk. Zcela irelevantní jsou pak například data o tom, jakou vystudoval vysokou školu.

7.2.2 Přesnost

Přesnost v případě dat neznamená pouze matematickou přesnost, ale technicky vzato i obecnou správnost dat. Použijeme-li příklad s tělesnou teplotou z předchozího příkladu, tak za předpokladu, že skutečná tělesná teplota pacienta je 39°C, tak údaj „38,97°C“ má bezesporu dostatečnou matematickou přesnost a je zároveň správný, takže podmínku splňuje. Oproti tomu údaj „38,41°C“ ne, protože i přes matematickou přesnost je očividně nesprávný. Stejně tak „kolem devětatřiceti“ jsou chybná data, protože ačkoli jsou technicky vzato správná, rozhodně nejsou dostatečně přesná. [9] [20]

Ověřování přesnosti, resp. správnosti je především lidským úkolem, nicméně kvalitní systém může značně pomoci hned několika způsoby automatického ověření, počínaje upozorněním na očividně nesmyslné hodnoty („50,19°C“ = nemožná hodnota), přes kontrolu přesnosti („39°C“ = uvedeno málo desetinných míst) až po porovnávání s dříve naměřenými hodnotami a detekci nečekaných změn. [9]

7.2.3 Včasnost a dostupnost

Včasnost říká, *za jakou dobu lze data aktualizovat.* [20] Jinak řečeno nám včasnost říká, jestli data, která se k nám dostanou, jsou dostatečně nová, resp. „čerstvá.“ V principu nezáleží na tom, co způsobuje případné zpomalení (dlouhá doba měření, pomalé síťové připojení, časté poruchy…), důležitý je pouze rozdíl mezi momentem, kdy data potřebujeme (resp. kdy o ně požádáme) a momentem, kdy je dostaneme. Dostupnost pak řeší krajní situaci, ve které data nedostaneme nikdy. [20]

Pokud je opět použit příklad s tělesnou teplotou, informace o pacientově tělesné teplotě reflektující skutečný stav před deseti sekundami podmínku včasnosti bezesporu splňuje, informace o jeho tělesné teplotě před hodinou ji splňovat může a nemusí (záleží na konkrétní situaci), informace stará dva dny je pak již zcela zbytečná, stejně jako informace, která sice byla změřena, ale kvůli chybě v systému lékař nemá vůbec možnost ji získat.

7.2.4 Porovnatelnost a koherence

Porovnatelnost byla jednou z hlavních motivací pro vytvoření mezinárodní klasifikačních systémů, které byly zmíněny výše v této práci. Data splňují požadavek na porovnatelnost, pokud je možné dva různé údaje stejného typu objektivně srovnávat, ale také pokud je možné slučovat data stejného typu z více zdrojů. Podmínku porovnatelnosti mohou data porušit kvůli nejednotnému formátu či metodě měření. Koherence je pak víceméně zpřesnění této metriky, které vyžaduje nejen srovnatelný formát výsledných dat, ale i srovnatelný způsob jejich získání. [20]

V našem příkladě jsou hodnoty „38,54°C“ a „37,91°C“ porovnatelné, zatímco „38,54°C“ a „311K“ nikoli. Nejenže je v druhém případě nelze přímo srovnat, problémy by pravděpodobně způsobil například i pokus o jejich uložení do stejné databázové tabulky (podmínka slučování dat). V tomto případě by samozřejmě bylo velice jednoduché vytvořit software, který by porovnatelnost zajistil, nicméně u jiných typů dat, jako jsou textové zprávy či grafika může být tento problém jen velmi těžce překonatelný.

Zajímá-li nás koherence dat, pak příklad nesplňující porovnatelnost nesplňuje ani podmínku koherence (data byla očividně získána jiným způsobem měření), nicméně první případ, který je dostatečně porovnatelný, může a nemusí být koherentní. Pokud byly teploty změřeny stejným teploměrem na stejném místě, jsou data dostatečně koherentní. Pokud byl např. v jednom případě použit teploměr digitální a ve druhém analogový, data jsou sice porovnatelná, ale nejsou koherentní.

7.2.5 Úplnost

Udává, *jaká část potenciálních dat byla zachycena a uložena*. [20] Ve většině situací je možno získat mnohem více dat, než kolik jich ve skutečnosti nakonec změříme či jinak zjistíme a následně uložíme. Prakticky vždy je možné úplnost ještě zlepšovat (teplotu je možné měřit každý den, každou hodinu, každou minutu…), nicméně téměř ve všech případech existuje hranice, na které už nám získaná data postačují a zvyšování úplnosti tedy již nemá význam a je obyčejným plýtváním zdroji.

Data mohou být však neúplná i přes požadavek na jejich úplnost, např. z důvodu momentální nedostupnosti měřícího zařízení. V takovém případě je samozřejmě zpravidla požadováno zvýšení úplnosti v co nejkratším čase. [9]

7.2.6 Úspornost

Úspornost je požadavek, který je protiváhou úplnosti a přesnosti a prakticky vytváří výše zmíněnou hranici, za kterou je zvyšování úplnosti již plýtvání zdroji. *Rostoucí objem informací nemusí nutně vést k lepším informacím*, proto nám požadavek úspornosti říká, že pokud zadání nových dat nevede k ušetření času, snížení pravděpodobnosti chyby či obecně zlepšení informací, nemá existence těchto dat smysl. Zároveň je v rámci úspornosti požadováno, aby data byla zadávána v co nejkratší a nejjednodušší formě, upřednostňuje text před obrázkem (pokud mají stejnou informační hodnotu), kód před textem atd. [9]

7.2.7 Jedinečnost a integrita

Požadavek na jedinečnost znamená, že data by neměla být duplikována, pokud to není nezbytně nutné. Ukládání stejných dat více než jednou je plýtvání zdroji a zároveň může způsobit ztrátu integrity, tedy situaci, kdy stejná data mají různé hodnoty, a není jasné, která z nich je správná. V situaci, kdy je nutné ukládat duplicitní data, je tedy nezbytné zavést mechanismy, která zabrání porušení jejich integrity, jako je určení autoritativních dat (=ta, která jsou vždy správná). [9]

7.3 Některé standardizované datové formáty ve zdravotnictví

Stejně jako ve většině jiných oblastí, i ve zdravotnictví samozřejmě existují standardizované datové formáty, jejichž cílem je především zajistit kompatibilitu napříč různými platformami a softwarovými řešeními a umožnit vzájemnou komunikaci jinak rozdílných softwarových řešení, především pak informačních systémů. V této práci budou zmíněny tři z nich.

7.3.1 DASTA – Datový standard Ministerstva zdravotnictví České republiky

DASTA je standard, jehož hlavním cílem je umožnění vzájemné komunikace rozdílných informačních systémů a přenosu informací jak přes počítačové sítě, tak pomocí rozličných médií. Dalším z logických cílů tohoto standardu je pak vzájemná kompatibilita při přechodu z jednoho systému na jiný. [21]

Vývoj tohoto standardu byl zahájen v roce 1992, první verze byla dokončena o dva roky později. K zásadní změně pak došlo v roce 2002, kdy byl standard převeden kompletně do formátu XML za použití jazyka DTD. V současné době je standard akceptován většinou dodavatelů zdravotnických IS, přičemž mnozí z nich se zároveň podílejí na jeho dalším vývoji. [22]

Standard definuje jednotlivé položky elektronického záznamu pacienta pomocí DTD, což umožňuje určení parametrů jednotlivých položek, jako je datový typ, délka, popis atd. [21]

7.3.2 HL7 – Health Level 7

HL7 je rovněž série datových standardů určených pro výměnu, sdílení, získávání a integraci zdravotnických informací. Tyto standardy byly vytvořeny stejnojmennou neziskovou organizací, která byla založena již v roce 1987 a později akreditována ANSI. [23] Jsou nejvyužívanějšími zdravotnickými datovými standardy na světě. [24]

Rodina HL7 obsahuje standardy pro dokumenty (HL7 CDA), aplikace (HL7 CCOW) či zprávy (HL7 3.0). Organizace kromě standardů samotných poskytuje i různé návody a rady pro jejich implementaci či výukové materiály týkající se těchto standardů

a jejich využívání. [25] Standardy definují vlastní, proprietární formát dat (jazyk, datové typy atd.), nevyužívají XML či jiný standardní formát.

7.3.3 DICOM – Digital Imaging and Communications in Medicine

DICOM je další z mezinárodních standardů, týkající se čistě obrazových dat v medicíně, tedy výsledků procedur jako je CT, MRI, RTG, sonografie apod. Definuje způsob ukládání, tisku, přenosu a obecně práce s obrazovým materiálem. Skládá se z definic souborových formátů a síťového komunikačního protokolu využívajícího sadu protokolů TCP/IP. [26] Tento standard je často využíván pro integraci mnoha různých zařízení, jako jsou počítače, servery, scannery, tiskárny a další do v předchozích kapitolách zmiňovaných systémů PACS.

Základním principem tohoto standardu je, že jeho datový objekt obsahuje nejen obrazová data, ale i další atributy, jako jsou pacientovo jméno, identifikační číslo atd. Zároveň je možné uložit více obrazových atributů do jediného objektu, což umožňuje například uložení 3D obrazu/modelu nebo podobně. Samotná obrazová data pak mohou být uložena v mnoha různých formátech, například standardním JPEG.

8. Některé pojmy, týkající se dat

Následují definice některých zkratek a pojmů, které budou používány dále v textu. Během práce na následujících částech textu bylo relativně často čerpáno ze serveru *w3schools.com.* [27]

8.1 XML

XML, což je zkratka slov eXtensible Markup Language (v překladu rozšiřitelný značkovací jazyk) je značkovací jazyk, jehož účelem je umožnit strukturování, ukládání a přenos dat. Poskytuje hierarchickou strukturu značek („tagů“) pro popis jednotlivých informací či částí informací. Nemá však žádné předdefinované značky ani strukturu, návrh obojího je čistě na uživateli či vývojáři, což zajišťuje obrovskou univerzálnost a prakticky neomezené možnosti využití tohoto jazyka.

Jazyk XML umožňuje vytvořit určitý systém v datech a zároveň je oddělit od informací o tom, jak mají být zobrazeny. Zde je jeden z hlavních rozdílů oproti jiným jazykům, jako je například HTML, který nese informace jak o struktuře dat, tak o způsobu, jakým mají být zobrazovány. Použití XML může zároveň výrazně zjednodušit přenos a výměnu dat, především díky tomu, že je softwarově i hardwarově nezávislý a tudíž stoprocentně multiplatformní, což umožňuje sdílení dat mezi nekompatibilními systémy. Na základě jazyka XML bylo vytvořeno mnoho dalších jazyků (XHTML, WSDL…), protokolů (např. SOAP), formátů (RSS apod.) či standardů a specifikací (např. RDF).

Syntaxe jazyka XML je založena na hierarchickém systému prvků (elements), které tvoří strom; každý dokument má jeden kořenový prvek (root element), jenž na další úrovni má tzv. potomky (child elements), kteří mohou mít na další úrovni rovněž potomky atd. Prvek se skládá z počáteční a koncové značky (tagu), obsahu (text či další prvky) a případně atributů. Důležité je dodržení základních pravidel syntaxe, především správné ukončení všech prvků a jejich správné vnořování (nepřekrývání). Dokument, splňující veškerá pravidla, se označuje *well-formed*.

Pro definici struktury XML dokumentů se využívá schémat, jako jsou například DTD (Document Type Definition – seznam prvků, které mohou být použity) či XML Schema. Informace o zobrazení je možno ukládat do tzv. stylesheetů, jako jsou kaskádové styly CSS či Extensible Sheet Language Transformations – XLST.

8.2 DOM

DOM, neboli Document Object Model je objektová reprezentace XML dokumentů, využívaná v některých objektových programovacích jazycích. Definuje objekty pro prvky XML dokumentů a především metody pro čtení, přidávání, odebírání či změnu jednotlivých prvků.

Funguje na principu uzlů (nodes), kdy vše v XML dokumentu je považováno za uzel – samotný dokument, jeho prvky, textový obsah, atributy i komentáře. Tyto pak tvoří stromovou strukuturu, odpovídající struktuře původního dokumentu.

Prvním krokem pro použití DOM je převod XML dokumentu na DOM objekt, což je provedeno pomocí speciální funkce či sady funkcí – tzv. parseru. Vzniklý objekt obsahuje zmíněnou strukturu uzlů, přičemž každý uzel je objekt s určitými vlastnostmi (typ, název, odkazy na nadřazené či podřízené uzly…) a metodami (získej atribut, nastav atribut, získej odkazy na podřízené uzly…), které jsou používány pro práci s dokumentem.

8.3 URI

Uniform Resource Identifier, v překladu jednotný identifikátor zdroje, je řetězec znaků, který jednoznačně identifikuje zdroj informací a jeho umístění.[28] Příkladem URI je www adresa stránky či souboru – jednoznačně určuje zdroj informací (jedna www adresa vždy odpovídá jedné stránce či souboru) a zároveň cestu, kterou se k němu lze dostat.

8.4 Strukturovaná, nestrukturovaná a semistrukturovaná data

8.4.1 Strukturovaná data

Jedná se typicky o data uložená v různých databázových systémech. Data jsou organizována v určité struktuře, mají určené datové typy a jsou vždy jasně identifikovatelná. Strukturovaná data jsou srozumitelná jak pro lidského uživatele, tak pro počítačové systémy. Vyhledávání ve strukturovaných datech je relativně snadné. [29]

8.4.2 Nestrukturovaná data

Jsou data v podobě čistého textu, audio či video souboru a podobně. Tato forma obvykle nabízí více dat než forma strukturovaná, avšak zpracování těchto dat pomocí počítače je výrazně složitější a náročnější, než dat strukturovaných. Totéž platí pro vyhledávání v těchto datech. [30]

8.4.3 Semistrukturovaná – hybridní data

Tento typ dat je strukturován pouze do určité míry, resp. úrovně, přičemž struktura je obsažena přímo v datech, například ve formě určitého systému značek. Jinak řečeno, semistrukturovaná data jsou organizována dle určitého schématu, které je obsaženo přímo v těchto datech. [31] Tato forma je kompromisem mezi strukturovaným modelem, který nelze ve spoustě případů aplikovat, a modelem nestrukturovaným, který přináší řadu problémů při strojním zpracování. Typickým příkladem jsou XML dokumenty.

9. Data ke zpracování

9.1 Vlastnosti dat

Jak již bylo zmíněno, data ke zpracování jsou textové medicínské zprávy, zápisy výsledků lékařských vyšetření a případně další texty podobného rázu. Z toho samozřejmě plynou určité specifické vlastnosti.

První z nich je častá přítomnost odborných termínů a cizích slov obecně. To je zcela očekávatelné, viditelné po přečtení byť několika málo prvních zpráv a později provedená frekvenční analýza to potvrdila. Z částečných výsledků frekvenční analýzy, provedené nad vzorovou množinou záznamů, zobrazených v tabulce 9.1, je zřejmé, že po vynechání předložek a spojek lze cca polovinu z prvních dvaceti nejčastěji se vyskytujících považovat za odborné termíny.

je	507
CT	463
ACM	439
mozek	420
vlevo	349
vpravo	339
volný	297
ACI	280
tepna	274
změna	268
povodit	257
odstup	244
uzávěr	237
jsou	234
ischemie	194
levý	178
ACC	165
mm	161
sin	155
stenóza	154

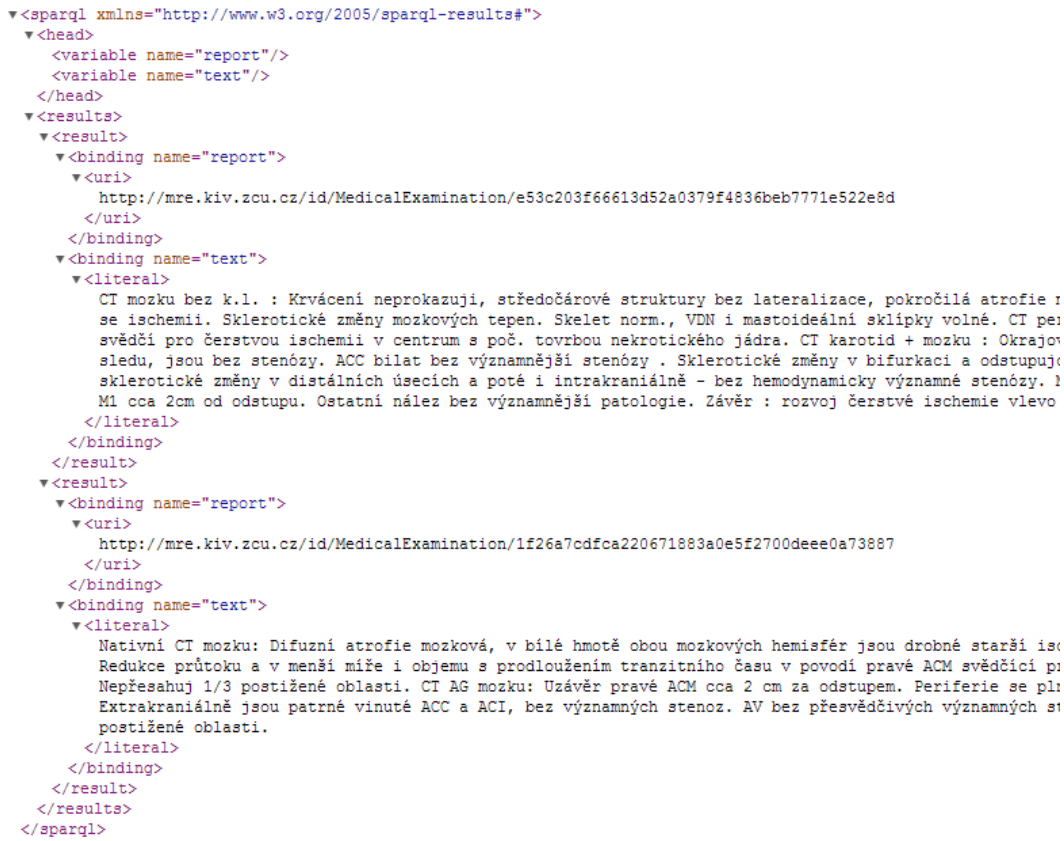
Tab. 9.1 – Částečné výsledky frekvenční analýzy vzorové množiny zpráv

Při návrhu způsobu zpracování je samozřejmě nutno s touto vlastností počítat, optimalizovat funkčnost pro práci s texty s takto úzkou tematikou a testovat funkce použitého softwaru na vhodné množině zpráv.

Druhou vlastností, resp. zvláštností tohoto druhu textů je relativně časté používání různých zkratek a rovněž poměrně veliké množství překlepů a chyb. V případě zdravotnických zpráv to vzhledem k jejich častému psaní ve spěchu není nic neobvyklého. Z tohoto důvodu je před dalším zpracováním nutná nějaká forma předzpracování, během které budou chyby, překlepy, nesmyslné znaky apod. odstraněny a zkratky ideálně „rozepsány“ na původní slova, či jim alespoň nějakým způsoben přiřazen původní význam.

9.2 Formát dat

Data jsou uložena a dodávána ve struktuře, vytvořené pomocí jazyka XML. Zprávy jsou jednoznačně identifikovány pomocí URI. Na obrázku 9.1 je možné vidět příklad struktury, v jaké jsou data dodávána.



Obr. 9.1 – Ukázka formátu zpracovávaných dat

Jazyk XML je ve své podstatě formou semistrukturovaných, jinak řečeno hybridních, dat a soubory se zprávami, kterými se zabývá tato práce, jsou toho perfektním příkladem. Jak je zřejmé, data jsou strukturována do úrovně jednotlivých zpráv, zprávy jako takové jsou následně děleny pouze na URI a obsah, samotný text zprávy už nijak strukturován není. V případě dat, jako jsou lékařské zprávy, je tento přístup logickým krokem. Pro získání striktně strukturovaných dat by bylo potřeba zcela změnit způsob, jakým jsou informace zaznamenávány a klasické zprávy nahradit, či alespoň doplnit o nějakou formu formuláře, který by vyžadoval zadání přesných zjištěných parametrů, naměřených hodnot a podobně.

Strojové zpracování je tedy jednodušší, než by bylo zpracování nestrukturovaných dat (v tomto případě pravděpodobně čistého textu), nicméně nestrukturovanost zpráv výrazně ztěžuje některé úkony při zpracování. Jedná se např. o vyhledávání konkrétních výsledků či jiných hodnot, což zhoršuje možnosti automatického zpracování textu a implementaci pokročilejších funkcí, jako je již zmiňovaná automatická kategorizace zpráv dle některých hodnot v nich obsažených. Tato skutečnost je hlavním důvodem, proč je v první řadě nutné vyřešit základní možnosti zpracování, jako je vyhledávání v textech a podobně. Na těchto základech bude následně možné pokročilou funkčnost vystavět.

10. Programová realizace zpracování – úvod

Následující text si klade za cíl především nastínit základní principy řešení popisované funkčnosti, upozornit na zajímavosti a popsat problémy, které se při implementaci vyskytly. Nejedná se kompletní programovou dokumentaci, ta je k dispozici ve standardním formátu JavaDoc na přiloženém disku. Na daném disku se zároveň nachází spustitelná verze hotového programu (ve formátu Java Archive – *.jar*) společně se sadou vzorových dat pro ověření funkčnosti (viz uživatelský manuál k programu), finální verze projektu pro IDE NetBeans a zdrojové kódy programu v textové podobě.

10.1 Funkční požadavky na software

Cílem bylo vytvořit program, který bude možné použít k základnímu zpracování výše popsaných dat, tedy lékařských zpráv, uložených v XML souborech. Program měl nabízet tři základní funkce:

- Předzpracování textu zpráv, tedy jeho kontrolu na gramatické chyby, překlepy, odstranění nepatřičných znaků apod.
- Frekvenční analýzu textů zpráv.
- Vyhledávání ve zprávách.

Dalšími požadavky byly multiplatformnost, resp. možnost spuštění na OS Windows i Linux, uživatelská přívětivost umožňující použití programu běžnými uživateli a možnost rozšíření hotového softwaru o další, pokročilejší funkce.

Pro zprovoznění této základní funkčnosti bylo samozřejmě nutno zároveň vytvořit mnoho podpůrných metod a algoritmů, např. pro načítání či export dat a podobně.

10.2 Základní design programu

Software byl realizován v programovacím jazyce Java, a to především pro jeho multiplatformnost, která umožňuje bezproblémové spuštění pod OS Windows i Linux. Funkčnost pod OS Linux byla jedním ze základních požadavků. Další výhodou použití jazyka Java je dostupnost knihoven pro jednoduchou práci s XML dokumenty. Software byl vyvíjen v IDE NetBeans verze 7.1.1 a testován pod OS Windows 7 64bit a Ubuntu 12.10, v obou případech s nainstalovaným prostředím Java Runtime Environment ve verzi 7.

Samozřejmě byla využita řada objektů a metod z knihoven z Java API (Application Programming Interface), především z *java.util*, *java.io* a *javax.swing*. Jednalo se o datové struktury (např. *ArrayList*), prostředky pro práci se soubory, grafické komponenty atd. Při implementaci některých částí programu a především řešení problémů byly pro inspiraci využity informace, řešení a části kódu ze serveru *stackoverflow.com.* [32]

Jak bylo již několikrát zmíněno, hlavním cílem bylo vytvořit metody pro základní funkčnost, které bude v budoucnu možno dále využívat, rozšiřovat a používat při implementaci pokročilých funkcí. Veškeré požadované funkce jsou proto izolovány v samostatných metodách a uživatelské rozhraní není jejich součástí. V současné době je ovládání programu vyřešeno pomocí GUI, vytvořeného na základě knihovny Swing, v závislosti na budoucích požadavcích lze však většinu funkcí adaptovat pro použití přes textové/konzolové rozhraní či využít software jakožto součást většího celku.

11. Práce s daty

11.1 Načtení XML souboru

Prvním krokem bylo zajištění načtení dat z XML souborů do vhodných datových struktur v programu a jejich opětovného uložení ve správném formátu. Toto bylo z velké části vyřešeno díky balíčku *org.w3c.dom,* jenž je součástí Java API a obsahuje veškeré potřebné metody, třídy, datové typy atd. standardu DOM pro načítání, zpracování a tvorbu XML dokumentů. Kromě toho byly samozřejmě využity standardní komponenty API pro práci se soubory.

Zpracování XML dokumentu bylo rozděleno na dvě části. Během první z nich je uživatel vyzván k výběru souboru, ten je následně načten a pomocí třídy *DocumentBuilder* zpracován na DOM reprezentaci. V druhém kroku jsou následně z této reprezentace získána textová data a je využit *ArrayList* instancí třídy *Record* k jejich uložení pro další využití. Při běhu programu jsou pak kompletní data v této formě načtena v operační paměti.

11.2 Objekt Record

Jedná se v principu o reprezentaci jedné konkrétní lékařské zprávy, cílem je zjednodušit práci se zprávami tím, že jsou jejich URI a text uloženy pohromadě v instanci třídy – záznamu. Pro uložení více zpráv je pak využito standardní datové struktury *ArrayList*, obsahující odkazy na jednotlivé záznamy, jak je zmíněno v předchozím odstavci.

Součástí programu je zároveň třída RecordIterator, která zjednodušuje procházení ArrayListu instancí *Record*. Instance této třídy může být vytvořena nad ArrayListem typu *Record* a obsahuje metody pro sekvenční i náhodný přístup k záznamům v něm obsaženým.

11.3 Export do XML

Opačná akce, tedy převedení objektů *Record* zpět na DOM objekt a jeho následný export do XML souboru, probíhá velmi podobně. Pomocí *DocumentBuilder* je vytvořena DOM reprezentace dokumentu, do ní jsou postupně přidána všechna textová data ze záznamů a tato je následně za pomoci třídy *Transformer* převedena a uložena do zvoleného XML souboru.

12. Slovník

Funkčnost celého programu je do značné míry založena na slovníku, tedy seznamu existujících/správných slov a jejich variant (tvarů, zkratek a podobně). Seznam správných slov jako takový je samozřejmě nezbytný při kontrole textu na chyby či gramatické překlepy, kdy jsou slova v textu porovnávána se slovníkem. Schopnost přiřadit k sobě rozdílné tvary či zkratky stejného slova je pak využita při frekvenční analýze, vyhledávání a bude velmi užitečná při případné implementaci pokročilejších funkcí. Kvalita slovníku tedy do určité míry určuje kvalitu výsledků dodaných programem.

12.1 Slovník v textové podobě

Nezbytné bylo získání co nejkvalitnějšího českého slovníku, protože jeho tvorba by byla extrémně časově náročná. Jako ideální se jevily slovníky pro GNU program *aspell*, určený pro kontrolu pravopisu. Český slovník pro tento program je k dispozici pod licencí GNU GPL a je možné jej vygenerovat v čistě textové podobě pomocí jednoduchého *shell* příkazu: [33]

```
aspell –d cs dump master | aspell –l cs expand > dict.txt
```

To umožnilo jeho bezproblémové použití. Jako ideální se jevil i jeho formát, ve kterém je každý řádek textového souboru vyhrazen pro jedno slovo a jeho varianty (tvary, zkratky…), oddělené mezerami, jak lze vidět na obrázku 12.1.

```
obušek obušcích obuškům obušků obušky obuškem obušku
obvaz obvazech obvazům obvazů obvazy obvazem obvaze obvazu
obvazek obvazcích obvazkům obvazků obvazky obvazkem obvazku
obvazivo obvazivy obvazivech obvazivům obvazivem obvazivu obvaziva obvaziv
obvaziště obvazištích obvazištím obvazišť obvazišti obvazištěm
```

Obr. 12.1 – Příklad textového formátu slovníku použitého v programu

Tento formát byl tedy zachován, což umožňuje případné použití libovolného jiného slovníku, získaného z programu *aspell*. Díky textovému formátu není problém s načítáním ani exportem slovníku pomocí standardních knihoven Javy. Drobné problémy nastaly z důvodu nejednotnosti používaných kódových stránek, kdy v závislosti na nastavení systému v některých případech docházelo ke špatné interpretaci českých znaků. Tento problém byl vyřešen explicitním nastavením kódové stránky na UTF-8 za pomoci tříd *InputStreamReader* a *OutputStreamWriter*, které na rozdíl od původně použitých *FileReader* a *FileWriter* umožňují explicitní nastavení použité kódové stránky.

Například v případě vytvoření BufferedWriteru pro zápis do souboru bylo namísto starého, problémového řešení:

```
BufferedWriter wrajtr = new BufferedWriter(new
FileWriter(fajl));
```

Použito toto řešení:

```
BufferedWriter wrajtr = new BufferedWriter(new
          OutputStreamWriter(new FileOutputStream(fajl), "UTF-
8"));
```

Z toho samozřejmě zároveň plyne, že textové soubory se slovníkem musí být uloženy v kódování UTF-8, mají-li být použity v programu. To samé platí pro soubory se stop slovy (viz níže).

12.2 Slovník v programu

V programu je slovník reprezentován třídou *Dictionary*, obsahující jak datové struktury pro uložení jeho obsahu, tak potřebné metody. Jedna instance třídy vždy odpovídá jednomu slovníku, přičemž může být vždy načten pouze jeden slovník. Nic však nebrání případnému budoucímu rozšíření. Vytvoření nového slovníku standardně probíhá přímo ze zadaného textového souboru pomocí k tomuto účelu vytvořeného konstruktoru, nicméně třída umožňuje i vytvoření slovníku prázdného.

12.2.1 Datové struktury pro slovník

Při běhu programu je slovník načten ve dvojrozměrném ArrayListu, který technicky vzato odpovídá formátu textového souboru. Pro každé slovo a jeho tvary je vytvořen samostatný „záznam“ – *ArrayList* a odkazy na všechny tyto záznamy jsou následně uloženy opět do jedné struktury *ArrayList*. Struktura je lépe zřejmá z ilustračního obrázku 12.2.

Obr. 12.2 – Ilustrace datových struktur pro slovník

Jednorázové načtení celého slovníku do této struktury umožňuje rychlé vyhledávání ve slovníku (operace *get* u *ArrayList* má vždy konstatní složitost), což bylo prioritou, protože vyhledávání ve slovníku je velice často používaná operace. Nevýhodou je pak relativně veliká paměťová náročnost programu a pomalejší operace vyžadující ukládání dat do slovníku (amortizovaná složitost operace *add* je sice rovněž konstantní, avšak asymptotická složitost nejhoršího možného případu je lineární), jako je načítání ze souboru či řazení.

12.2.2 Metody slovníku

První kategorií metod, které lze ve třídě Dictionary nalézt, jsou samozřejmě metody pro vyhledávání. Kromě obligátního vyhledání konkrétního slova nabízejí ještě vyhledání všech slov, které začínají daným řetězcem (využito při předzpracování, viz níže) a získání všech variant a tvarů daného slova. Metody tedy umožňují, kromě samotného ověření, zda se slovo ve slovníku vůbec nachází, i načtení všech jeho případných tvarů a variant, což je velice důležitá vlastnost. Pro příklad je na obrázku 12.3 uvedena základní metoda pro vyhledání slova:

```
/**
 * Nalezení zadaného slova/varianty slova ve slovníku.
 * @param w Slovo, které má být vyhledáno
 * @return Řádek (index v ArrayListu), na kterém se slovo nachází, -1 pokud nebylo nalezeno.
 */
public int findWord(String w)
{
   ArrayList current;

   //nacitame radek po radku
    for (int i=0; i<dict.size(); i++)
    {
        current = dict.get(i);

        //radek prohledavame slovo od slova
        for (int j=0; j<current.size(); j++)
        {
            //pokud je nalezeno, vratime cislo prave prohledavaneho radku
            if (current.get(j).equals(w))
            {
                return i;
            }
        }
    }
    return -1;   //nebylo nalezeno
}
```

Obr. 12.3 – Příklad metody pro vyhledání slova ve slovníku

Do druhé kategorie spadají metody pro přidání nového slova a přidání tvaru již existujícího slova. Obě před přidáním provádějí základní kontrolu na duplicitu, nelze tedy přidat slovo, které se ve slovníku již nachází. Při přidání nového slova je jednoduše vytvořen nový „záznam“ slova v podobě ArrayListu, který je přidán do slovníkového ArrayListu. Při přidání varianty je dle parametru vybrán záznam, do kterého nový tvar patří a do nějž je tvar tedy následně přidán. Obě metody vracejí informaci o tom, zda se přidání zdařilo, či nikoli. Na obrázku 12.4 lze vidět, jak vypadá metoda pro přidání varianty slova.

```
/**
 * Přidání tvaru/varianty ke slovu, které ve slovníku již je. Kontroluje případnou duplicitu.
 * @param word Slovo, ke kterému bude přidána varianta.
 * @param variant Varianta slova, která bude přidána
 * @return Pokud ve slovníku není ani původní slovo -1, pokud se přidání podařilo 0, pokud přidávaná varianta
 * ve slovníku již je, vrátí číslo řádku (index v ArrayListu), na kterém se nachází.
 */
public int addVariant(String word, String variant)
{
    //spustime metodu pro vyhledani pridavane varianty ve slovniku a ulozime vysledek
    int locvariant = findWord(variant);

    if(locvariant!=-1)
    {
        return locvariant;   //varianta uz ve slovniku je, nebude pridana, vraci kde je
    }
    else
    {
       int locword = findWord(word); //najdeme korenove slovo
       if (locword==-1)
       {
           return -1;    //slovo, ke kteremu chceme pridat tvar, neni ve slovniku
       }
       else
       {
           ArrayList locsave = dict.get(locword);    //ziskame zaznam pro korenove slovo
           locsave.add(variant);     //pridame k nemu novy tvar
           return 0;
       }
    }
}
```

Obr. 12.4 – Příklad metody pro přidání varianty slova do slovníku

Dále tato třída samozřejmě obsahuje metody pro převod slovníku zpět na text a jeho uložení do textového souboru. Rovněž v ní lze nalézt metodu, která seřadí slovník dle zadaných výsledků frekvenční analýzy, což je funkce nezbytná pro urychlení některých operací. Toto téma je podrobněji rozebráno dále v tomto textu, v části, věnující se frekvenční analýze.

Při implementaci slovníku se objevila řada problémů vyplývajících z velikosti slovníku (např. zmiňovaný český slovník, který je v programu v současné době používán, má přes 300000 řádků a v textové podobě je velikost jeho souboru cca 60MB), kdy v případě použití nevhodných řešení a špatné optimalizace algoritmů byly časy provedení některých funkcí enormně dlouhé. Jednalo se například o příliš složitý algoritmus pro kontrolu duplicity při přidávání slov či použití *String* namísto *StringBuilder* při převodu slovníku zpět na text. Postupně se podařilo většinu těchto problémů vyřešit a zkrátit časy provedení většiny metod na max. několik málo sekund, oproti původní situaci, kdy teoretické časy dokončení byly v řádech desítek minut. Relativně časově náročné zůstává první řazení slovníku, během kterého je nutno provést veliké množství operací *get* a *add*.

12.2.3 Problém s velikostí haldy

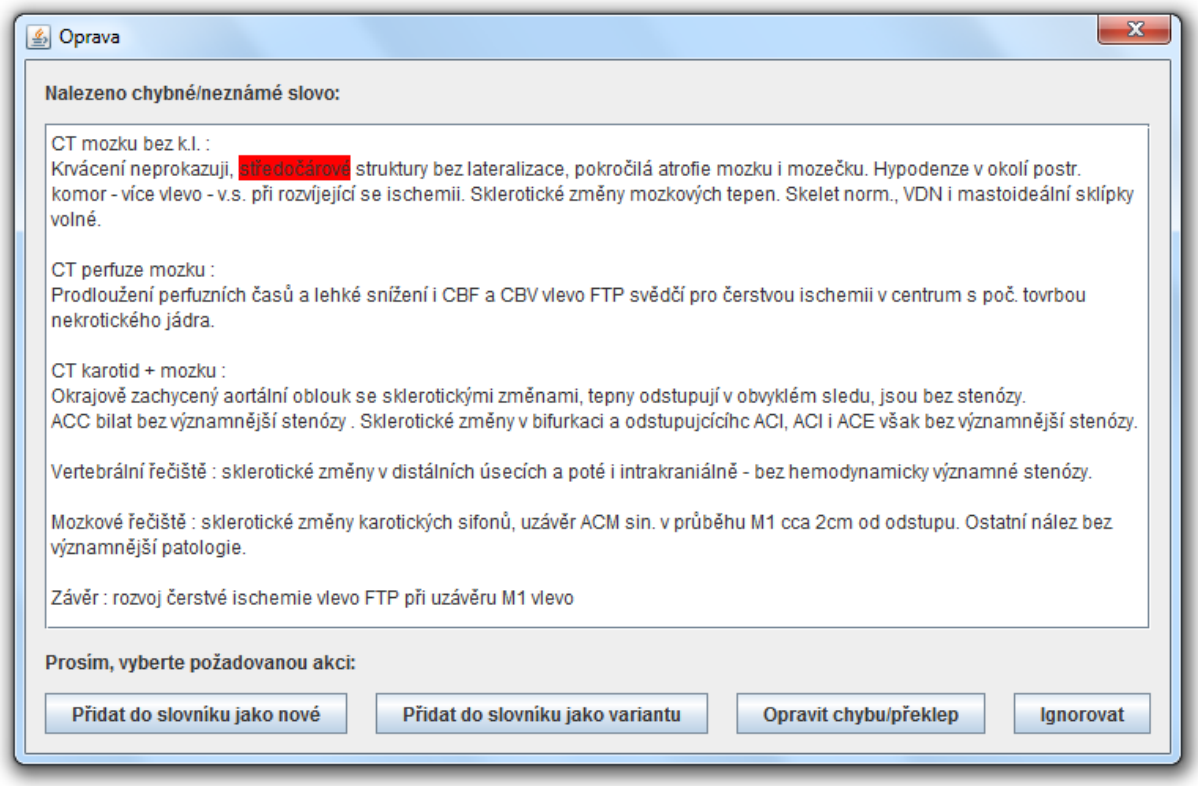
Kvůli velikosti slovníku, který program po celou dobu svého běhu udržuje v paměti, se na některých počítačích vyskytl rovněž problém s nedostatečnou maximální velikostí haldy (tzv. *heap size*) pro Java Vitual Machine. Následkem tohoto systémového nastavení byla nemožnost načíst slovník a pád programu, pokud se o to uživatel pokusil. Jediným řešením, které bylo funkční multiplatformně a nevyžadovalo po uživateli žádné složité úkony, bylo při spuštění programu zkontrolovat nastavenou velikost haldy a v případě, že byla nedostatečná, spustit nový JVM s nastavenou větší (konkrétně 1024MB) *heap size*, v něm znovu spustit program a původní JVM ukončit.

13. Předzpracování – kontrola textů

Předzpracování textů zpráv, které umožňuje opravu chyb a překlepů a zároveň doplnění slovníku o nová slova, tvary, zkratky, odborné termíny apod., je velmi důležitou součástí výsledného řešení, zvlášť vzhledem k povaze dat, zmíněné dříve v tomto textu. Prvním následkem provedení předzpracování je zlepšení kvality zpráv, které je logickým následkem opravy chyb. Doplnění slovníku pak zpřesňuje a zlepšuje výsledky programu v budoucnu, a to jak samotného předzpracování, tak i frekvenční analýzy či vyhledávání. Kvalitní slovník, doplněný především o odborné termíny a zkratky, bude rovněž s největší pravděpodobností potřeba pro další plánovanou funkčnost.

13.1 Princip předzpracování

Základní princip této funkce je procházení množiny zpráv (*ArrayList* typu *Record*, viz výše) slovo po slovu a porovnání každého slova se slovníkem, resp. jeho vyhledání ve slovníku. Části textu, které nelze považovat za slova (interpunkce, číselné údaje apod.) jsou ignorovány. Nachází-li se slovo ve slovníku, není potřeba další akce. Pokud nikoli, je uživateli zobrazeno dialogové okno s textem zpracovávané zprávy, zvýrazněným slovem a žádostí o volbu požadované akce. Toto okno lze vidět na obrázku 13.1.



Obr. 13.1 – Dialogové okno předzpracování textu

Ve zkratce je tedy možné slovo označit za správné a přidat do slovníku, slovo označit za špatné a opravit jej či pro tuto chvíli ignorovat. Po provedení některé z akcí program automaticky pokračuje na další slovo a tímto způsobem jsou postupně zkontrolovány všechny načtené zprávy. Případné změny se samozřejmě okamžitě projeví v datových strukturách programu a opravené zprávy i rozšířený slovník je možné po předzpracování exportovat.

13.2 Akce během předzpracování

Programová logika funkce předzpracování je relativně jednoduchá a vyjma dialogového okna je obsažena prakticky v jedné jediné veřejné (*public*) statické metodě. Přesto se v krátkosti podívejme na jednotlivé možnosti, kterými může uživatel řešit upozornění na neznámé slovo.

13.2.1 Doplnění slovníku

Především během prvních použití programu se ve zprávách samozřejmě objeví množství slov, jako jsou odborné termíny či zkratky, které slovník neobsahuje, ačkoli jsou naprosto v pořádku. Je tedy možné takové slovo do slovníku přidat a tím dosáhnout jak toho, že program je při příštím nalezení bude již považovat za validní, tak zlepšení kvality slovníku jako takového, který se tímto „učí“ další slova či tvary a varianty slov.

Při přidání zcela nového slova je jednoduše zavolána metoda ze třídy *Dictionary*, která toto zařídí. Při přidávání varianty je v první řadě potřeba zjistit, k jakému slovu ve slovníku přísluší. V případě zkratek je program schopen dohledat ve slovníku slova, která by mohla být základním tvarem a uživateli je nabídnout. Tato funkčnost je řešena pomocí výše zmiňované metody slovníku, která vyhledá a vrátí všechna slova, začínající daným řetězcem. To velmi zjednodušuje doplňování slovníku a zároveň snižuje množství chyb, jichž se uživatel při této činnosti dopustí.

Implementovat stejnou funkci pro různé tvary slov se zatím bohužel vzhledem ke složitosti české gramatiky nepodařilo. Původně uvažovaná řešení s využitím stemmingu či lemmatizace se neukázala jako průchodná z důvodů nedostupnosti vhodného hotového nástroje/knihovny a veliké náročnosti vývoje vlastního.

Uživatel má samozřejmě rovněž možnost napsat základní tvar slova sám. V takovém případě je i tento vyhledán ve slovníku, a pokud není nalezen, má uživatel možnost přidat do slovníku obě slova či změnit svoji volbu. Samotné přidání do slovníku je pak vyřešeno opět příslušnou metodou z třídy slovníku. Pro ukázku se na obrázku 13.2 podívejme na část algoritmu, která obsluhuje přidání varianty slova.

```
case "variant":
{
    String[] basics = dictionary.findSubstring(currentword);    //nalezeni moznych zakladnich slov
    //zobrazeni dialogu pro vyber ci zadani zakladniho slova
    SelectVariantDialog svd = new SelectVariantDialog(null, true);
    svd.setOptions(basics);
    svd.setVisible(true);
    String basic = svd.getSelected();   //nacteni slova z dialogu

    if(basic!=null) //pokud uzivatel nejake slovo vybral ci zadal
    {
        //zakladni tvar ve slovniku je, pridame k nemu variantu a pokracujeme dal
        if(dictionary.findWord(basic)!=-1)
        {
            dictionary.addVariant(basic, currentword);
            done = true;
        }
        //neni - uzivatel muze pridat obe, nebo zvolit jinou akci
        else
        {
            int dialogresult = JOptionPane.showConfirmDialog(null, "Základní tvar není ve slovníku.
            if(dialogresult==0) //vybral pridat obe
            {
                dictionary.addWord(basic);  //pridame zakladni
                dictionary.addVariant(basic, currentword);  //pridame variantu
                done=true;
            }
        }
    }
    break;
}
```

Obr 13.2 – Část metody předzpracování pro přidání nové varianty slova do slovníku

13.2.2 Oprava pravopisné chyby či překlepu

Problém, zmiňovaný na konci předešlého odstavce, zároveň způsobil, že opravování chyb a překlepů při kontrole textu je závislé čistě na uživateli a funkční na jednoduchém principu, kdy je slovo vyhledáno ve slovníku a není-li nalezeno, je uživatel požádán o jeho opravu. Je zobrazeno dialogové okno, umožňující úpravu či přepsání daného slova. Po jeho potvrzení je slovo opět ověřeno oproti slovníku, pokud je v pořádku, pokračuje se dále, pokud není, je uživatel vyzván k jiné opravě či přidání opraveného slova do slovníku.

I přesto, že bylo věnováno relativně mnoho času pokusům o využití některého z automatických nástrojů pro kontrolu pravopisu, nepodařilo se nalézt žádný, který by byl dostupný a zároveň bezproblémově funkční pro Český jazyk. Nejlépe se jevilo řešení pomocí *Jazzy*, ale i to ztroskotalo na problémech s Češtinou. Samozřejmě by bylo vhodné tento problém vyřešit a automatické opravy doplnit. Kromě vývoje zcela nového softwaru pro kontrolu pravopisu se nabízí úprava *Jazzy* tak, aby podporoval Český jazyk, nebo využití některého komerčního řešení. Všechny možnosti však přesahují svojí náročností rozsah této práce, nicméně při dalším vývoji o nich bude rozhodně uvažováno.

14. Frekvenční analýza

Cílem frekvenční analýzy je ve své podstatě pouze spočítat výskyty jednotlivých slov v daném textu. V tomto programu však byla frekvenční analýza rozšířena o další možnosti a to především z toho důvodu, že slouží a v budoucnu i bude sloužit jako základ pro další, pokročilejší funkce. Mezi tyto možnosti se řadí zejména záznam URI zpráv, ve kterých se slovo vyskytlo a možnost seskupování odlišných variant stejného slova za pomoci slovníku.

14.1 Objekt výsledku frekvenční analýzy

Pro pochopení funkčnosti frekvenční analýzy v tomto softwaru je v první řadě nutné definovat a vysvětlit, jakým způsobem jsou ukládány její výsledky. Přístup je podobný, jako v případě zpráv – záznamů. Pro uložení výsledků byla vytvořena speciální třída s názvem *FreqAnalysisResult*, jejíž každá instance reprezentuje jedno slovo či skupinu slov (viz dále v textu), nalezené v množině zpráv. Při prvním nalezení slova je pro něj vytvořena a do množiny výsledků analýzy (tvořené opět strukturou *ArrayList*) přidána instance této třídy, do které jsou následně ukládány informace o výskytu daného slova či slov. Kromě slova či slov samotných a samozřejmého počítadla výskytů obsahuje i seznam URI zpráv, ve kterých se slovo vyskytlo.

Součástí této třídy jsou samozřejmě i potřebné metody, které umožňují zjistit, kterému slovu či slovům náleží daná instance, přidat do ní nový výskyt či převést v ní uložené výsledky na text. Třída rovněž implementuje rozhraní Comparable, především kvůli umožnění jednoduchého řazení pomocí metod z balíčku java.util.Collections.

14.2 Princip frekvenční analýzy

Funkčnost je stejně, jako v případě předzpracování, obsažena v jedné veřejné statické metodě a několika *private* podpůrných metodách. Základním principem je opět procházení všech načtených zpráv slovo po slovu, přičemž číselné údaje, interpunkce apod. jsou vynechávány. Každé slovo je nejprve vyhledáno v dosavadních výsledcích analýzy, pokud je mezi nimi nalezeno, je do jeho instance přidán výskyt (přidání URI právě analyzované zprávy a zvýšení počítadla výskytů) a pokračuje se dalším slovem. Pokud nalezeno není, je pro něj vytvořena nová výsledková instance a tato je přidána k dosavadním výsledkům. Takto program pokračuje, dokud neprojde všechny zprávy. Následně jsou uživateli poskytnuty výsledky prostřednictvím dialogového okna, umožňujícího dva typy zobrazení (viz následující odstavce) a export do textového souboru.

14.3 Možnosti frekvenční analýzy

14.3.1 Základní analýza bez slovníku

Toto je nejjednodušší verze frekvenční analýzy, jejíž princip se prakticky neliší od základu, popsaného v předchozím odstavci. Každé slovo je počítáno zvlášť a má vlastní výsledkovou instanci, obsahující počet a místa jeho výskytů, zařazenou do seznamu výsledků. Postupně jsou vyhledány výskyty slov ve všech zprávách, následně je zobrazeno dialogové okno s výsledky.

14.3.2 Analýza se slovníkem

Při využití této varianty analýzy jsou dle slovníku seskupovány různé varianty stejného slova a tyto jsou následně počítány společně. Pro pochopení nejlépe poslouží zjednodušený příklad z uživatelského manuálu:

Ve zpracovávaných zprávách se nachází 2x slovo *mozek*, 1x slovo *mozku* a 3x slovo *mozkem*. Výsledky frekvenční analýzy pak budou následující:

Pokud je provedena základní analýza bez slovníku:

- mozek: 2 výskyty
- mozku: 1 výskyt
- mozkem: 3 výskyty

Pokud je provedena analýza se slovníkem:

- mozek, mozku, mozkem: 6 výskytů

Funkce je založena na vcelku jednoduchém principu, kdy každá instance výsledkové třídy odpovídá nikoli jednomu slovu, ale množině slov, resp. slovu a všem jeho tvarům. V momentě, kdy je některé ze slov nalezeno během analýzy poprvé a je tedy nutné pro něj vytvořit výsledkový objekt, dané slovo je napřed dohledáno ve slovníku a do nově vzniklého objektu jsou přiřazeny-nakopírovány i všechny jeho případné varianty. Výskyty všech těchto variant jsou pak během dalšího průběhu analýzy sdružovány do zmiňované instance, ergo počítány dohromady. Pokud slovo není nalezeno ve slovníku, je pro něj vytvořena samostatná instance, stejně jako v případě analýzy bez slovníku. Pro ilustraci je možno se podívat na část kódu, která je použita v případě, kdy je během analýzy se slovníkem některé slovo nalezeno poprvé a je tedy nutné pro něj vytvořit novou instanci výsledkové třídy a zařadit ji mezi dosavadní výsledky.

```
int dicloc = dictionary.findWord(currentword);  //najdi slovo ve slovniku
FreqAnalysisResult newresult;
if(dicloc==-1)      //pokud tam neni, vytvor novy zaznam pouze s timto slovem a pridej jej do mnoziny vysledku
{
    String [] wrd = new String[1];
    wrd[0] = currentword;
    newresult = new FreqAnalysisResult(wrd);
    newresult.addOccurence(uri);
    results.add(newresult);
}
else
{
    //novy vysledkovy zaznam obsahujici nalezene slovo a vsechny jeho tvary
    String [] dictline = dictionary.getLine(dicloc);
    newresult = new FreqAnalysisResult(dictline);
    //pridame vyskyt
    newresult.addOccurence(uri);
    //pridame jej mezi vysledky
    results.add(newresult);
}
```

Obr. 14.1 – Část metody frekvenční analýzy, která zajišťuje vznik nových instancí výsledkového objektu při analýze se slovníkem

Tato varianta frekvenční analýzy je užitečná například při vyhledávání. Vzhledem k nutnosti dohledávat každé nové slovo ve slovníku a v případě jeho nalezení „kopírovat“ jeho varianty je však relativně pomalá i na nadprůměrných počítačích. Při větší délce analyzovaného textu (100 a více zpráv) je doba jejího běhu v řádech desítek sekund. Vzhledem k tomu, že software je určen pro texty s úzkou tématikou, lze

provádění analýz zrychlit seřazením slovníku dle výsledků jedné frekvenční analýzy, provedené nad co největším a nejreprezentativnějším vzorkem zpráv.

14.3.3 Seřazení slovníku podle výsledků analýzy

Tato funkce využívá metody pro seřazení slovníku, která je obsažena ve třídě *Dictionary* (viz výše v tomto textu). Po spuštění seřadí slovník podle počtů výskytů slov, tedy slovo s největším počtem výskytů bude ve slovníku na prvním místě, slovo s druhým největším počtem výskytů na druhém místě atd. Slova, která se v analyzovaném dokumentu nevyskytují, jsou pak v abecedním pořadí na konci slovníku. Během dalších analýz je tedy většina slov nalezena hned na začátku slovníku, což zásadně zkrátí časy vyhledávání a tudíž i provádění analýzy jako takové. Jak již bylo zmíněno v popisu samotné metody pro řazení, tato operace je rovněž dosti časově náročná, avšak ve výsledku několikanásobně zrychlí následnou práci s programem, zvláště pokud je často spouštěna slovníková frekvenční analýza či vyhledávání.

14.3.4 Stop slova

Stop slova jsou slova, která uživatel považuje za irelevantní a chce je vynechat z frekvenční analýzy. Zpravidla se jedná o spojky, předložky a podobně, avšak záleží čistě na uživateli a jeho požadavcích. Seznam slov lze do programu načíst jednoduše z textového souboru, princip jejich vynechání je pak rovněž jednoduchý. Při procházení zpráv během analýzy je každé slovo předtím, než je započítáno, porovnáno se stop slovy, načtenými ze souboru do pole a pokud se s některým z nich shoduje, program ihned pokračuje na další slovo, jak ilustruje kód na obrázku 14.2.

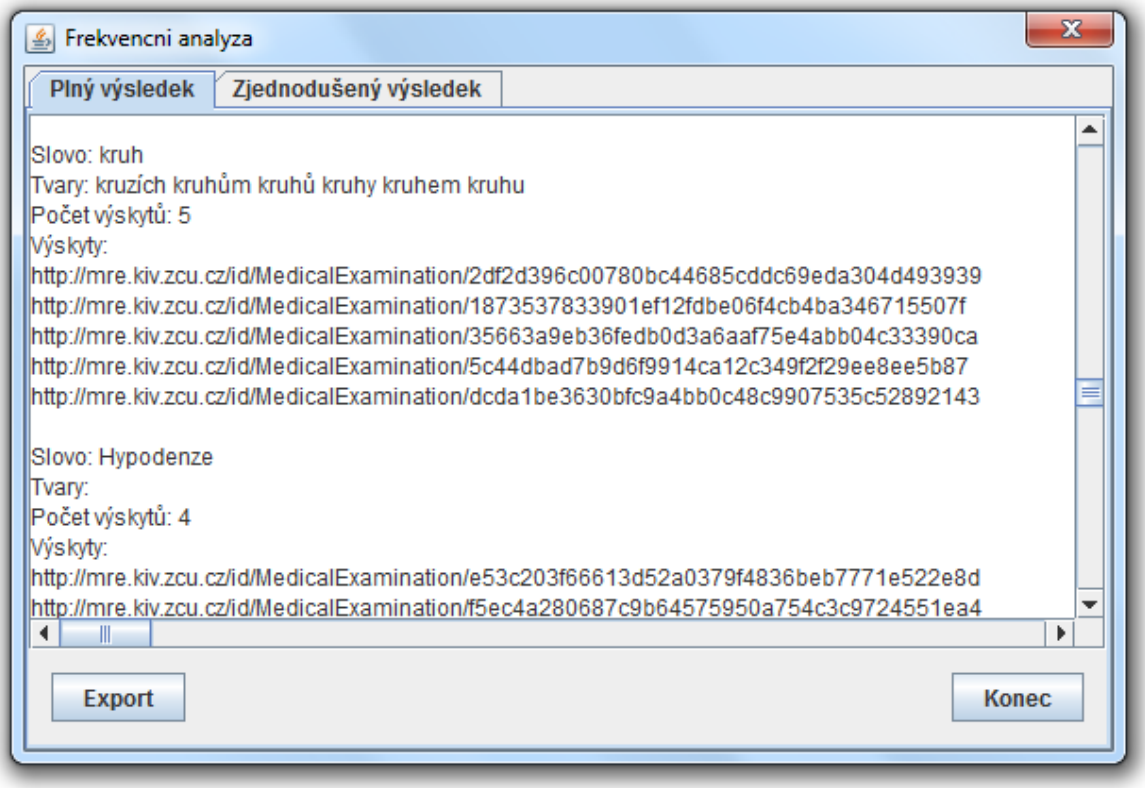
```
//pokud je pouzit seznam stopwords a slovo v nem je, preskocime
if(stopwords!=null)
{
    boolean stop = false;
    for(int j=0;j<stopwords.length;j++)
    {
        if (currentword.equals(stopwords[j]))
            stop=true;
    }

    if(stop)
        continue;
}
```

Obr. 14.2 – Ověření, zda právě zpracovávané slovo nepatří mezi stop slova

14.3.5 Možnosti zobrazení výsledků

Program umožňuje dvě různé možnosti zobrazení výsledků frekvenční analýzy – plnou a zjednodušenou. V plném zobrazení jsou kromě počtu výskytů slova dostupné veškeré další informace, včetně všech tvarů slova (v případě analýzy se slovníkem) a nalezených URI zpráv, ve kterých se slovo vyskytlo. Zjednodušené zobrazení oproti tomu nabízí přehledný seznam základních tvarů slov a počtů jejich výskytů. Na obrázku 14.3 je možné vidět plné zobrazení výsledků analýzy se slovníkem.



Obr. 14.3 – Dialogové okno výsledků frekvenční analýzy, zobrazující kompletní výsledky analýzy s použitím slovníku

Výsledky jsou v první řadě seřazeny sestupně podle počtu výskytů. Metody pro převod výsledku do obou textových podob jsou obsaženy přímo ve třídě *FreqAnalysisResult*, výpis je tedy otázkou jednoduchého projití ArrayListu výsledků, jejich spojení do řetězců a vypsání do okna.

15. Vyhledávání

Vyhledávání je poslední z hlavních funkcí softwaru. Nabízí základní vyhledávací funkce, tedy vyhledání jednoho slova, vyhledání několika slov a vyhledání přesného řetězce. Při vyhledávání více slov lze nastavit, zda mají být nalezeny záznamy obsahující všechna zadaná slova, či zda postačuje alespoň jedno ze slov. Díky použití slovníku je zároveň možné vyhledávat nejen přesně zadané slovo, ale i jeho případné tvary a varianty. Vyhledání přesného řetězce je specifická funkce, určená především pro vyhledávání číselných dat a jiných podobných údajů, které dvě prvně zmíněné funkce nejsou schopny vyhledat.

15.1 Řešení vyhledávacích funkcí

Vyhledávací funkčnost je řešena pomocí dvou veřejných statických metod – jedné pro vyhledávání slov a druhé pro vyhledávání přesného textu. Spouštění vyhledávání i zobrazení jeho výsledků je řešeno pomocí dialogového okna.

15.1.2 Vyhledávání slov

První dvě zmíněné funkce, tedy vyhledání všech zadaných slov a vyhledání alespoň jednoho ze zadaných slov, jsou založeny na frekvenční analýze, resp. na jejích výsledcích. Slova tedy nejsou vyhledávána přímo v záznamech, ale ve výsledcích dřívě proběhlé frekvenční analýzy. Důsledkem toho je relativně pomalé první spuštění vyhledávání (musí být provedena frekvenční analýza, pokud již nebyla nad danou množinou záznamů spuštěna dříve), avšak velmi rychlé vyhledávání samotné.

Princip vyhledávání samotného je zřejmý. V ArrayListu výsledků frekvenční analýzy jsou nalezeny výsledky, odpovídající vyhledávaným slovům. Z nich jsou následně získány URI, ve kterých se slova vyskytla. Další krok závisí na parametru vyhledávání:

- Mají-li být nalezeny záznamy, obsahující všechna slova, jsou množiny URI výskytů každého slova porovnány mezi sebou. Do výsledků vyhledávání jsou pak zařazeny jen ty URI, které byly nalezeny ve všech množinách zároveň.
- Pro možnost, kdy stačí alespoň jedno ze zadaných slov, jsou všechny množiny URI sjednoceny do jedné, která je následně vyčištěna od duplicit.

Tímto je vyhledávání dokončeno a výsledky jsou v podobě ArrayListu získaných URI z metody vráceny pro další zpracování, což znamená jejich zobrazení v dialogovém okně a případně export do textového souboru.

15.1.3 Vyhledávání přesného textu

Jediná funkce vyhledávání, která nevyužívá výsledků frekvenční analýzy, ale hledá přímo v záznamech, je vyhledání přesného textu. Jedná se o relativně jednoduchou funkci založenou na metodě *substring*, která vrátí pouze záznamy, v nichž se nachází skutečně přesně zadaný text včetně interpunkce apod. Tato funkce byla implementována především pro umožnění vyhledávání jiných, než textových údajů, jako jsou údaje číselné a podobně; dá se předpokládat, že tato funkčnost bude nutná při budoucím rozšiřování programu.

Spuštění této funkce a zobrazení výsledků vyhledávání je řešeno stejně, jako v předchozím případě. Pro ukázku se můžeme na obr 15.1 podívat na zdrojový kód této funkce, ve kterém lze zároveň vidět příklad použití iterátoru *RecordIterator*, zmíněného výše.

```java
/**
 * Vyhledá záznamy obsahující přesně zadaný řetězec.
 * @param rec ArrayList záznamů, ve kterých bude vyhledáváno.
 * @param search Vyhledávaný řetězec.
 * @return ArrayList URI záznamů, odpovídajících parametru vyhledávání.
 */
public static ArrayList<String> searchForText(ArrayList<Record> rec, String search)
{
    RecordIterator iterator = new RecordIterator(rec);
    Record current;

    ArrayList<String> uris = new ArrayList<>();

    while((current=iterator.getNext())!=null)    //prochazeni zaznamu
    {
        String text = current.text;
        if (text.contains(search))
            uris.add(current.uri);

    }

    return uris;

}
```

Obr. 15.1 – Metoda pro vyhledání přesně zadaného řetězce s ukázkou použití iterátoru

16. Grafické uživatelské rozhraní

Ovládací rozhraní programu je čistě grafické, založené na knihovně *Swing* a vytvořené za použití NetBeans Designer. Skládá se z hlavního okna *JFrame* a několika dialogových oken *JDialog*. Hlavní okno umožňuje spuštění všech funkcí – načítání a ukládání dat, předzpracování, frekvenční analýzy a vyhledávání. Samotné funkce se pak ovládají pomocí příslušných dialogových oken, která jsou zobrazena po jejich spuštění. Instance hlavního okna zároveň obsahuje několik globálních proměnných pro data, která musí být v paměti po celou dobu běhu programu, jako jsou zpracovávané záznamy či slovník.

Ovládací prvky v hlavním okně jsou dynamicky aktivovány či deaktivovány podle toho, zda jsou načtena data, potřebná k použití daných funkcí. To zabraňuje uživateli v pokusech o spouštění funkcí, pro které zatím nenačetl potřebná data (např. předzpracování nelze spustit bez slovníku). Stejně tak veškerá dialogová okna jsou ošetřena tak, aby uživatel žádnou svojí akcí nemohl vyvolat pád nebo neočekávané chování programu. Uživatel je rovněž dynamicky informován o činnosti programu pomocí stavového panelu v hlavním okně.

Přesný popis ovládání programu je možno nalézt v uživatelském manuálu, dostupném v příloze tohoto textu.

17. Závěr

Teoretická část práce vytváří co nejkomplexnější náhled do problematiky medicínské informatiky a obecně využití informačních technologií ve zdravotnictví. Zvláštní pozornost byla věnována dvěma oblastem zdravotnické informatiky – medicínským informačním systémům a problematice dat ve zdravotnictví. V případě zdravotnických IS byla uvedena jak obecná teorie týkající se jejich dělení, účelu či využitelnosti, tak konkrétní příklady některých takových systémů. Rovněž u datové problematiky je obsahem textu nejen obecná teorie týkající se například požadovaných vlastností dat, ale i na některé konkrétní datové formáty, využívané v medicíně.

Při zpracování praktické části práce se požadovanou funkčnost podařilo z větší části implementovat. Největší překážky v průběhu vývoje byly nedostupnost vhodného řešení pro kontrolu českého pravopisu a přílišná složitost a/nebo hardwarová náročnost některých algoritmů při zpracování větších objemů dat. Druhý zmíněný problém se podařilo ve většině případů vyřešit optimalizací algoritmů a volbou lepších datových typů/struktur či metod, určitý prostor pro optimalizaci zůstává v případě slovníkové frekvenční analýzy a řazení slovníku dle jejích výsledků. První z problémů zůstává otevřen k dalšímu řešení.

V současné době je software připraven jak pro použití koncovými uživateli (především modul pro předzpracování textu), tak pro další rozšiřování o pokročilejší funkce (moduly frekvenční analýzy a vyhledávání). Program byl důkladně otestován na několika počítačích a vykazuje plnou a bezproblémovou funkčnost jak pod OS Windows, tak pod OS GNU/Linux. Přesto lze však předpokládat, že při ostrém použití i dalším rozvoji budou odhaleny další drobnější chyby a především možnosti pro optimalizaci a další vylepšení stávajících řešení.

Literatura a zdroje informací

[1] KALOUSKOVÁ, Eva a Jiřina POLÁKOVÁ. Data, informace, znalosti - rozdíly, podrobnosti. *Knowledge Management* [online]. [cit. 2012-11-04]. Dostupné z: knowledgemanagement.ic.cz/informaceznalosti.doc
[2] KUČEROVÁ, Helena. Definice informace. Data - informace - znalosti. *Vyšší odborná škola informačních služeb: Helena Kučerová – domovská stránka* [online]. [cit. 2012-11-04]. Dostupné z: http://web.sks.cz/users/ku/ZIZ/inform1.htm
[3] NAIDR, Jan P. Úvod do zdravotnické informatiky. *Katedra managementu Fakulty managementu VŠE v Praze* [online]. [cit. 2012-11-06]. Dostupné z: http://www.fm.vse.cz/km/wp-content/uploads/2009/10/VEH598-01-uvod.pdf
[4] CEJPEK, Jiří. *Informace, komunikace a myšlení*. 1. vyd. Praha: Karolinum, 1998, 179 s. ISBN 80-718-4767-4
[5] SZABÓ, Zoltán. Informační systémy ve zdravotnictví. *České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství: ZOD* [online]. [cit. 2012-11-11]. Dostupné z: http://webzam.fbmi.cvut.cz/szabozol/ISZ/2pr_2008_09.pdf
[6] HRONEK, Jiří. Informační systémy. *Univerzita Palackého v Olomouci: Phoenix* [online]. [cit. 2012-11-12]. Dostupné z: http://phoenix.inf.upol.cz/esf/ucebni/infoSys.pdf
[7] BASL, Josef. *Podnikové informační systémy: podnik v informační společnosti*. 2., výrazně přeprac. a rozš. vyd. Praha: Grada, 2008, 283 s. Management v informační společnosti. ISBN 978-80-247-2279-5
[8] Health Informatics and Health Information Management. *University of Illinois at Chicago* [online]. [cit. 2012-11-13]. Dostupné z: http://healthinformaticsdegree.uic.edu/health-informatics/health-informatics/
[9] Informační systémy ve zdravotnictví. *Jihočeská univerzita v Českých Budějovicích, Zdravotně sociální fakulta, Katedra radiologie, toxikologie a ochrany obyvatelstva* [online]. [cit. 2012-11-13]. Dostupné z: http://www.zsf.jcu.cz/structure/departments/kra/informace-pro-studenty/ucebni_texty/ochrana-obyvatelstva-se-zamerenim-na-cbrne-aplikovana-radiobiologie-a-toxikologie-krizova-radiobiologie-a-toxikologie/informacni-systemy-ve-zdravotnictvi
[10] Number of Titles Currently Indexed for Index Medicus® and MEDLINE® on PubMed®. *National Library of Medicine - National Institutes of Health* [online]. [cit. 2012-11-27]. Dostupné z: http://www.nlm.nih.gov/bsd/num_titles.html
[11] PŘEČKOVÁ, Petra. Jazyk českých lékařských zpráv a klasifikační systémy v medicíně. *Official journal of the European Federation of Medical Informatics* [online]. [cit. 2012-11-14]. Dostupné z: http://www.ejbi.org/en/ejbi/article/53-cs-jazyk-ceskych-lekarskych-zprav-a-klasifikacni-systemy-v-medicine-.html
[12] CHOPLIN, Robert H. Picture archiving and communication systems: an overview. *Radiographics*, January 1992, no. 12, p. 127-129
[13] PEARL, Judea. *Heuristics: intelligent search strategies for computer problem solving*. Reading, Mass.: Addison-Wesley Pub. Co., c1984, xvii, 382 p. ISBN 02-010-5594-5.

[14] Laboratorní informační systém INFOLAB®. *Česká společnost zdravotnické informatiky a vědeckých informací* [online]. [cit. 2012-11-19]. Dostupné z: http://www.medinfo.cz/oblasti/informacni-systemy/is/doc/lis/LIS_MPPROGRAM_2006.doc
[15] Farmaceutický informační systém FaRMIS. *FaRMIS s.r.o.* [online]. [cit. 2012-11-19]. Dostupné z: http://www.medinfo.cz/oblasti/informacni-systemy/is/doc/lis/LIS_MPPROGRAM_2006.doc
[16] SEINER, Miroslav. Informační systémy ve veřejném zdravotnictví. *INFOMED - nezávislý server o zdravotnické informatice* [online]. [cit. 2012-11-22]. Dostupné z: http://www.infomed.cz/libfile/file_download.php?id=61
[17] SEINER, Miroslav. Nová podoba www.uzis.cz. *INFOMED - nezávislý server o zdravotnické informatice* [online]. [cit. 2012-11-22]. Dostupné z: http://www.infomed.cz/ps/article.php?arid=74
[18] Infekce v ČR - EPIDAT. *Státní zdravotní ústav* [online]. [cit. 2012-11-11]. Dostupné z: http://www.szu.cz/publikace/data/infekce-v-cr
[19] KASAL, P. *Lékařská informatika*. Vyd. 1. Praha: Karolinum, 1998, 543 s. ISBN 80-718-4594-9.
[20] KRÁL, Jaroslav a Michal ŽEMLIČKA. Kvalita dat a informací – základní omezení IT ve veřejné správě. *Risk-Management.cz* [online]. [cit. 2013-02-15]. Dostupné z: http://www.risk-management.cz/clanky/kral-zemlicka-kvalita-dat-a-informaci-zakladni-omezeni-IT-ve-verejne-sprave.pdf
[21] NAIDR, Jan P. Datový standard MZ ČR a NČLP. *Katedra managementu Fakulty managementu VŠE v Praze* [online]. [cit. 2013-02-14]. Dostupné z: http://www.fm.vse.cz/km/wp-content/uploads/2009/10/VEH598-06-datovystandard.pdf
[22] ZÁMEČNÍK, Miroslav. Datový standard MZ ČR a NČLP v praxi: současný stav a další rozvoj. *INFOMED - nezávislý server o zdravotnické informatice* [online]. [cit. 2013-02-14]. Dostupné z: www.infomed.cz/libfile/file_download.php?id=59
[23] About HL7. *Health Level Seven International* [online]. [cit. 2013-02-16]. Dostupné z: http://www.hl7.org/about/index.cfm
[24] The HL7 Evolution: Comparing HL7 Version 2 to Version 3, Including a History of Version 2. *Corepoint health* [online]. [cit. 2013-02-16]. Dostupné z: http://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf
[25] Introduction to HL7 Standards. *Health Level Seven International* [online]. [cit. 2013-02-16]. Dostupné z: http://www.hl7.org/implement/standards/index.cfm?ref=nav
[26] DICOM: Digital Imaging and Communications in Medicine. *DICOM Homepage* [online]. [cit. 2013-02-16]. Dostupné z: http://medical.nema.org/dicom/geninfo/Brochure.pdf
[27] *W3 schools* [online]. [cit. 2013-04-16]. Dostupné z: www.w3schools.com
[28] What is URI? *The Hong Kong Polytechnic University* [online]. [cit. 2013-04-17]. Dostupné z: http://www.eie.polyu.edu.hk/~entchsun/EIE423Lab/wiurl.html
[29] Structured data. *Webopedia* [online]. [cit. 2013-04-17]. Dostupné z: http://www.webopedia.com/TERM/S/structured_data.html
[30] Data v počítači. *Informatika* [online]. [cit. 2013-04-17]. Dostupné z: http://informatika.topsid.com/index.php?war=data_v_pocitaci
[31] BUNEMAN, Peter. Semistructured data. *Penn Engineering* [online]. [cit. 2013-04-17]. Dostupné z: http://www.cis.upenn.edu/~db/abstracts/semistructured.html
[32] *Stack Overflow* [online]. [cit. 2013-04-16]. Dostupné z: www.stackoverflow.com
[33] How to convert aspell dictionary to simple list of words?. *Super User* [online]. [cit. 2013-03-22]. Dostupné z: http://superuser.com/questions/137957/how-to-convert-aspell-dictionary-to-simple-list-of-words

Přílohy

Příloha A – Uživatelská příručka

Úvod

Software, na jehož použití se právě chystáte, je určen pro zpracování textových zdravotnických zpráv, uložených ve formátu XML dle specifikací RDF. Jeho základní funkce jsou předzpracování/kontrola zpráv a oprava případných chyb a překlepů, frekvenční analýza zpráv a vyhledávání ve zprávách.

Je vytvořen v jazyce Java, díky čemuž jej lze používat pod libovolnou verzí OS Windows či Linux. Podmínkou je nainstalované runtime prostředí Java verze 7 či novější. Pod staršími verzemi nebyl program testován a nelze tedy zaručit jeho funkčnost, proto je vhodné před spuštěním provést aktualizaci Java RE na nejnovější dostupnou verzi.

Pro plné využití je kromě zpracovávaných dat ve správném formátu vyžadován slovník a seznam stop slov (viz dále v tomto manuálu). Program lze použít i bez těchto součástí, jeho funkčnost bude však omezena.

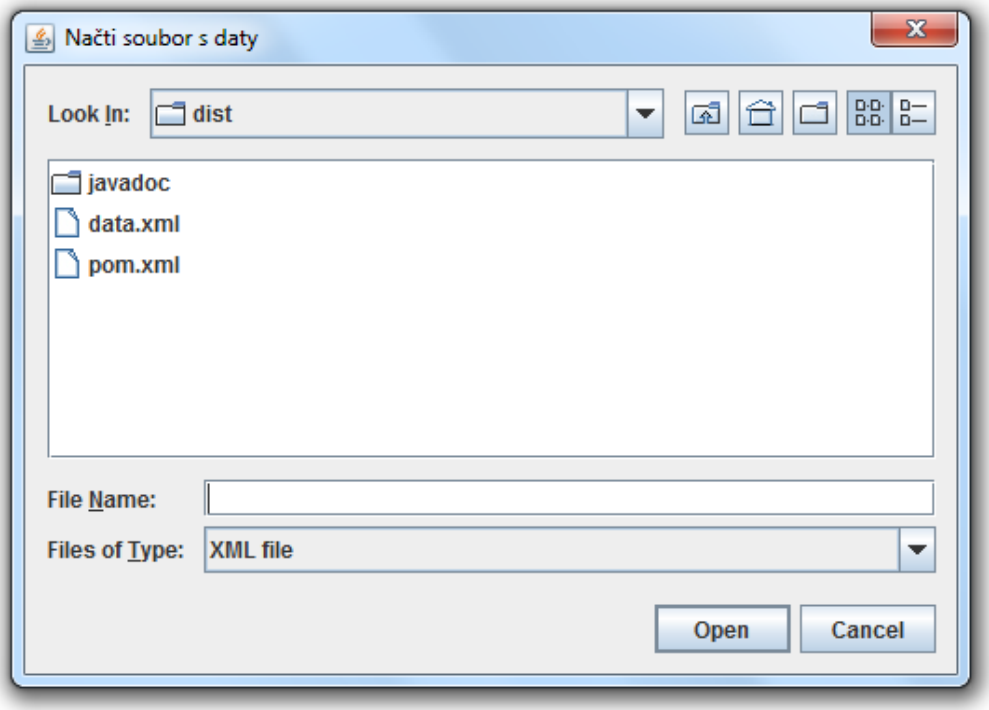
Spuštění programu a načtení dat

Spuštění programu

Program se spouští souborem *med_text_tools.jar*. Po jeho spuštění je postupně nabídnuto načtení zpracovávaných dat, slovníku a seznamu stop slov. Tyto akce je možné odložit na později zavřením příslušných dialogových oken, nezapomínejte však, že pro plnou funkčnost programu jsou vyžadovány všechny tři datové soubory.

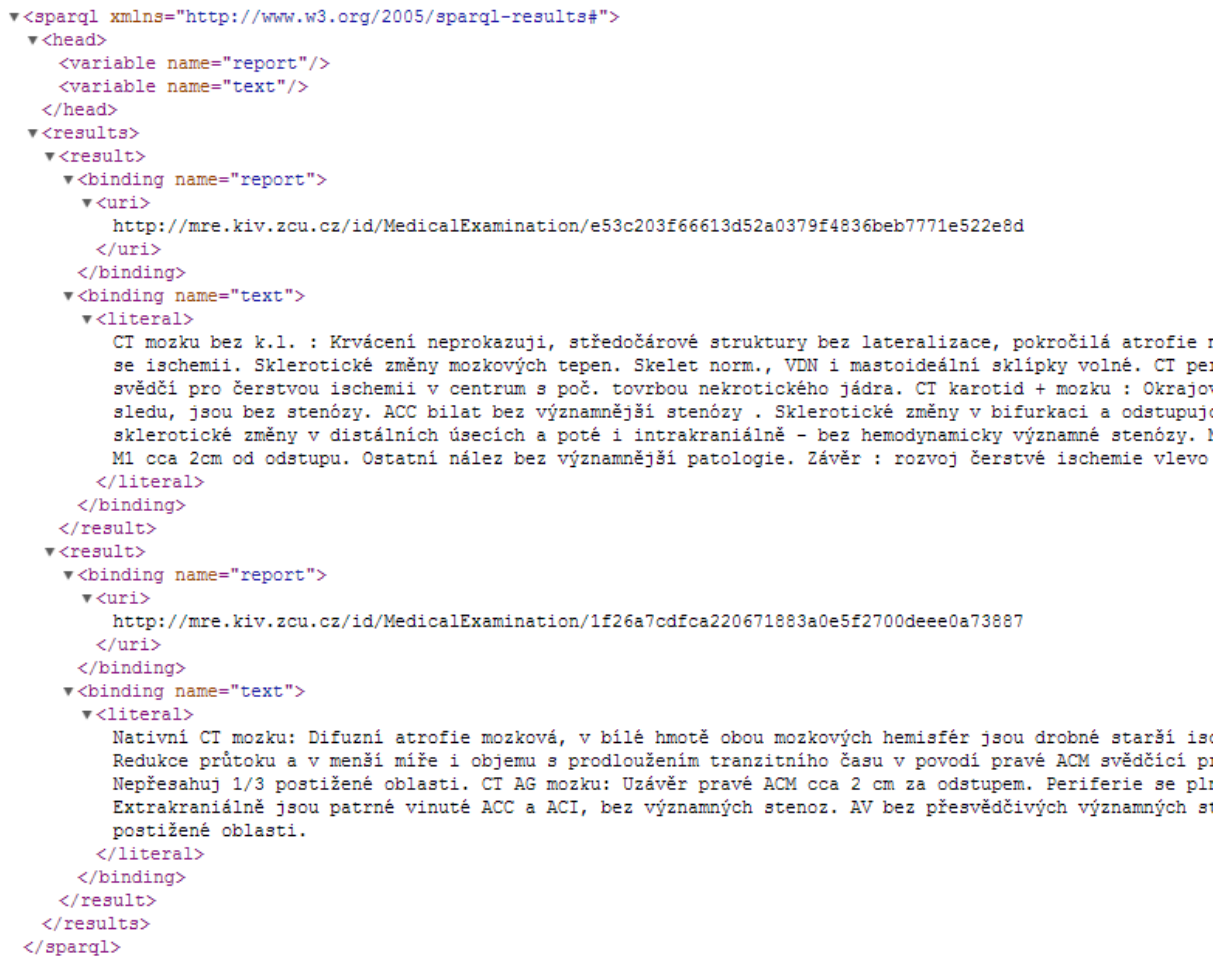
Načtení dat pro zpracování

Zvolení datového souboru obsahujícího zdravotnické záznamy určené ke zpracování je proveden prostřednictvím klasického dialogového okna:



Obr. 1 – Dialogové okno pro načtení souboru s daty pro zpracování

Soubor s daty musí být ve formátu, pro který byl software vytvořen, jinak program nebude pracovat správně. Struktura tohoto formátu vypadá takto:



Obr. 2 – příklad struktury XML dokumentu pro zpracování programem

Načtení slovníku

Slovník, obsahující přehled správných slov a jejich tvarů a variant, které se mohou v textech vyskytovat, je vyžadován pro kontrolu textu a některé funkce frekvenční analýzy a vyhledávání. Výběr slovníku se provádí stejným způsobem, jako výběr dat, tedy rovněž pomocí dialogového okna. Soubor se slovníkem musí být v pevně daném textovém formátu, kdy na každém řádku se nachází jedno slovo a všechny jeho tvary či varianty (zkratky apod.):

```
obušek obušcích obuškům obušků obušky obuškem obušku
obvaz obvazech obvazům obvazů obvazy obvazem obvaze obvazu
obvazek obvazcích obvazkům obvazků obvazky obvazkem obvazku
obvazivo obvazivy obvazivech obvazivům obvazivem obvazivu obvaziva obvaziv
obvaziště obvazištích obvazištím obvazišť obvazišti obvazištěm
```

Obr. 3 – Příklad formátu textového souboru se slovníkem

Soubor se slovníkem musí mít nastavené kódování UTF-8. Při případné ruční tvorbě či editaci slovníku dbejte na dodržení těchto pravidel, jinak může dojít k chybné funkčnosti programu. Pokud chcete otevřít textový soubor se slovníkem přímo, mějte na paměti, že vzhledem k jeho velikosti mohou mít některé programy (např. Poznámkový

blok) problémy s jeho otevřením. V takovém případě je nutné použít jiný textový editor, např. PSPad. Formát odpovídá formátu slovníku vygenerovaného ze softwaru aspell, je tedy možné přímo použít takto získaný slovník. K programu je standardně dodáván český slovník, který je k dispozici pod licencí GNU GPL.

Pokud zadaný soubor slovníku neexistuje, bude dialogové okno pro jeho výběr zobrazeno opakovaně, dokud nebude zvolen existující slovník či odmítnuta výzva k načtení.

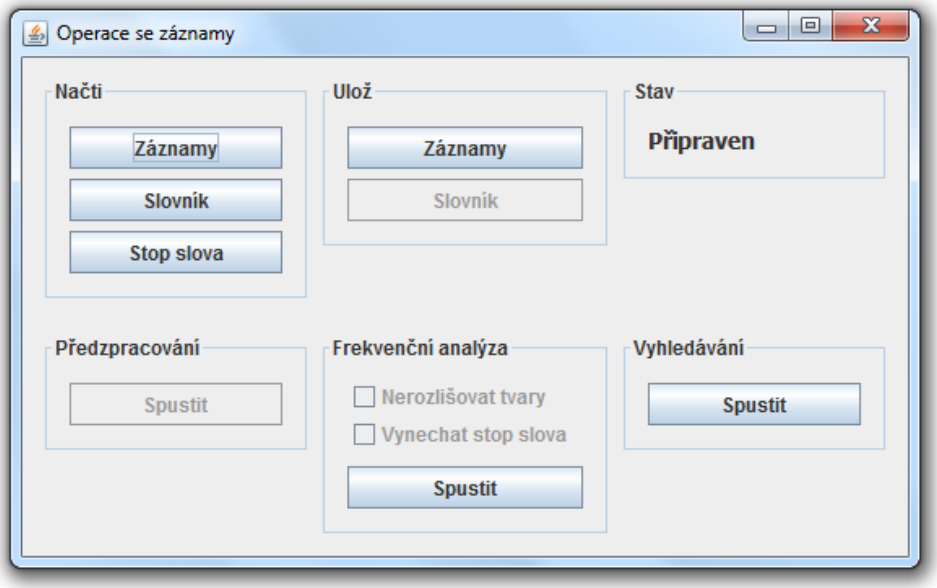
Načtení stop slov

Stop slova jsou slova, která nejsou při frekvenční analýze za žádných okolností relevantní a je tedy vhodné je vynechat pro zrychlení procesu a zpřesnění výsledků. Výběr vhodné množiny stop slov závisí na charakteru zpracovávaných zpráv, nejčastěji vyhledávaných heslech atd., zpravidla se však jedná především o předložky, spojky a podobně.

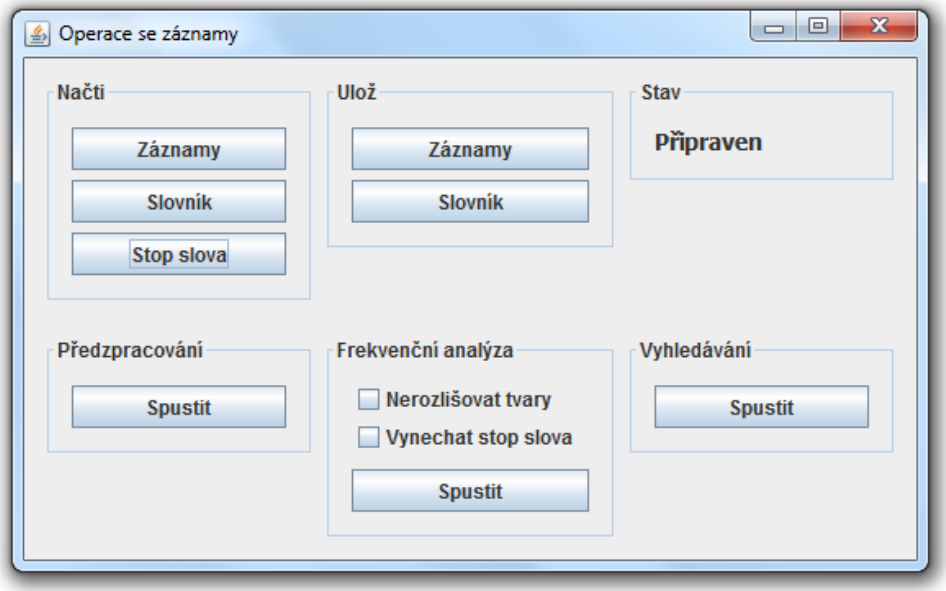
Stejně jako v případě slovníku a dat se soubor se stop slovy vybírá pomocí dialogového okna. Všechna slova v souboru se musejí nacházet na jedné řádce, oddělena mezerou. Jakýkoli text na dalších řádcích souboru je programem ignorován. Stejně jako v případě slovníku, soubor se stop slovy musí mít nastavené kódování UTF-8. S programem je dodáván vzorový soubor stop slov.

Hlavní ovládací rozhraní programu

Hlavní ovládací grafické rozhraní je zobrazeno ihned po úvodních nabídkách načtení dat a umožňuje uživateli spouštět jednotlivé funkce softwaru, stejně jako načítat či ukládat použitá data. Rozhraní je rozděleno do několika částí, jejichž přesný význam a funkčnost budou popsány dále v textu. Některé ovládací prvky mohou být neaktivní, pokud nebyla načtena data, která jsou vyžadována k jejich použití. Tyto jsou samozřejmě okamžitě zpřístupněny, pokud uživatel data dodatečně načte. Příklad je možno vidět na následujících dvou obrázcích:



Obr. 4 – Hlavní ovládací rozhraní programu, slovník ani stop slova nejsou načteny



Obr. 5 – Hlavní ovládací rozhraní programu, všechna data jsou načtena a všechny funkce tudíž dostupné

Operace s daty a stavový panel

Načti

Tlačítka v tomto poli umožňují otevřít stejná dialogová okna pro načtení dat, která jsou popsána výše v části týkající se spouštění programu.

Ulož

Otevře dialogová okna pro uložení zpracovaných dat a upraveného slovníku do externích souborů. Tato okna jsou rovněž automaticky otevřena při ukončení programu.

Záznamy

Dialogové okno umožní vybrat existující soubor xml či zadat jméno nového souboru. Existující soubor bude automaticky přepsán. Data jsou uložena ve formátu XML, přičemž jeho přesná struktura je popsána v části týkající se načítání dat.

Slovník

Uložení slovníku provedete stejným způsobem, jako uložení dat, tedy výběrem souboru v dialogovém okně.

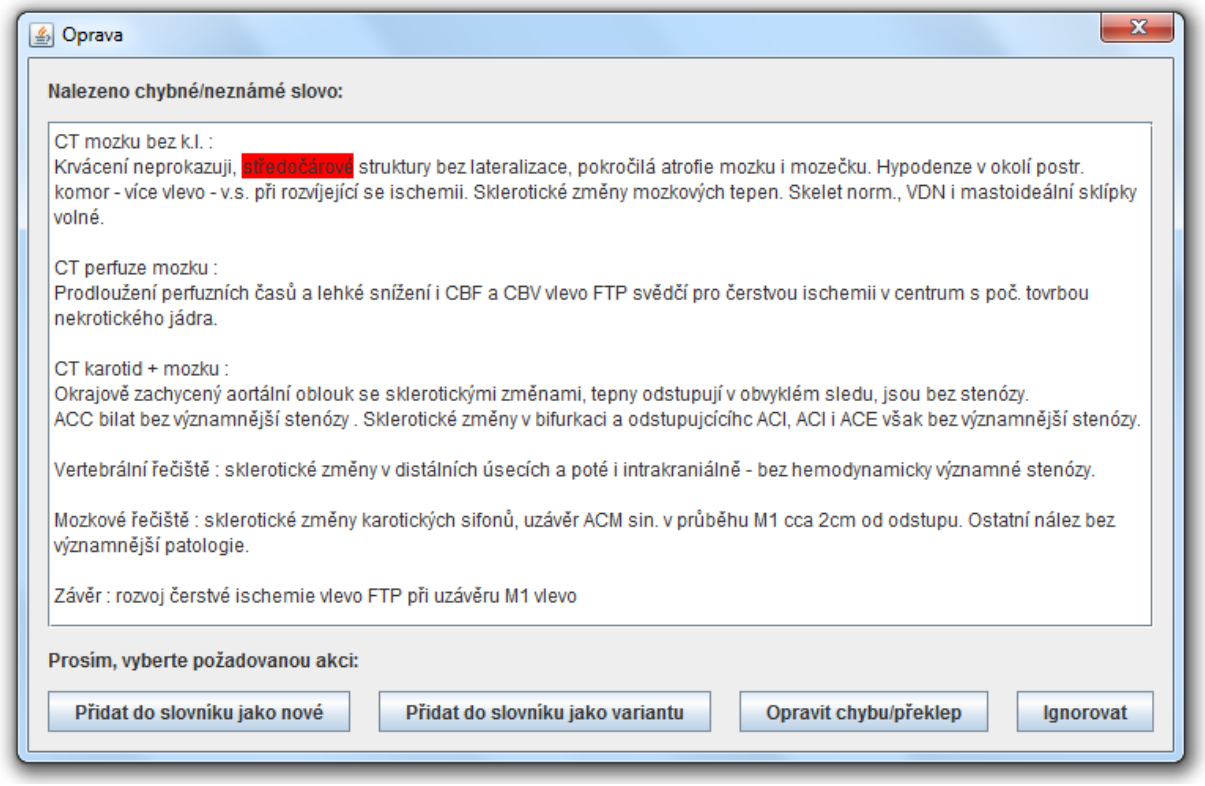
Stav

Účelem stavového panelu je informovat, zda je program v současné době zaneprázdněn, či je možné zadávat příkazy. Pokud je program zaneprázdněn a je tedy nutné počkat na dokončení stávající úlohy, text ve stavovém panelu se změní na *„Pracuji…“*. *„Připraven*“ pak naopak znamená, že je možné s programem pracovat.

Předzpracování

Tato funkce použije slovník k tomu, aby zkontrolovala správnost všech slov v textu a v případě, že najde možnou chybu, zobrazí dialogové okno s nabídkou možných řešení. Především v počátcích používání programu je tato situace relativně častá, vzhledem k tomu, že slovník mnoho slov (především odborných termínů apod.), jejich tvarů a zkratek neobsahuje a program je tedy vyhodnotí jakožto chybná. Je tedy potřeba věnovat čas doplnění slovníku o nová slova (přesný popis této funkčnosti naleznete dále v textu) a „naučit“ tak program texty lépe zpracovávat. Kvalita a úplnost slovníku do značné míry rozhoduje o kvalitě výsledků dosažených použitím programu.

Předzpracování je možné kdykoli přerušit zavřením dialogového okna, dosavadní výsledky zůstanou uloženy. **Nezapomeňte však, že po dokončení předzpracování textu je potřeba záznamy i slovník uložit, jinak budou po zavření programu veškeré úpravy a změny ztraceny!**



Obr. 6 – Dialogové okno s výběrem akce při objevení chybného či neznámého slova

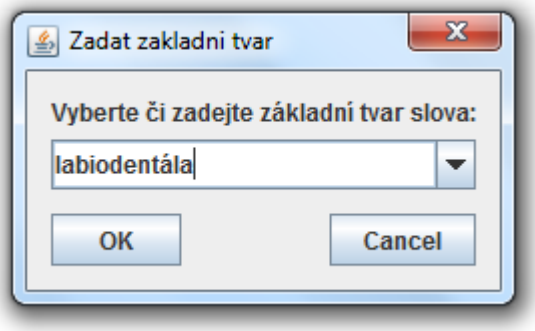
Jak můžete vidět na obrázku, program vždy při nalezení neznámého slova zobrazí text zprávy, ve které bylo nalezeno. Slovo je v něm zvýrazněno červeně. Je možné provést jednu ze čtyř akcí.

Přidat do slovníku jako nové

Tuto variantu volte v případě, že je slovo správně napsáno a není tvarem či zkratkou jiného slova. Po klepnutí na tlačítko bude přidáno do slovníku a při dalším zpracování tedy považováno za korektní slovo.

Přidat do slovníku jako variantu

Pokud je slovo zkratkou, tvarem či jinou variantou (např. stejné slovo s velkým počátečním písmenem) jiného slova, použijte tuto možnost. Program nabídne možná slova, jichž může být neznámé slovo variantou či zkratkou. Nejsou-li žádná taková nalezena či žádné z nich nevyhovuje, je možné vepsat vlastní návrh základní verze slova.



Obr. 7 – Zadání základního tvaru slova

Pokud se uživatelem vepsaný návrh na základní tvar shoduje s některým ze slov ve slovníku, je k němu varianta přidána. Pokud nikoli, je uživateli nabídnuto přidání obou slov. Toto je možné potvrdit, či odmítnout a zadat jiný návrh či vybrat zcela jinou akci.

Opravit chybu či překlep

Je-li slovo chybně napsané, pomocí tohoto tlačítka je možné vyvolat jednoduché dialogové okno, pomocí kterého jej lze přepsat. Po přepsání je nová verze opět zkontrolována oproti slovníku. Automatické návrhy oprav bohužel nejsou v této verzi programu implementovány.

Ignorovat

Tato možnost přeskočí dané slovo a pokračuje v kontrole textu bez jakékoli další akce.

Frekvenční analýza

Frekvenční analýza umožňuje zjistit počty a místa výskytů jednotlivých slov ve zpracovávané množině zpráv.

Spuštění analýzy

Analýza se spouští příslušným tlačítkem v hlavním rozhraní programu, před jejím provedením je možné nastavit dva parametry:

Nerozlišovat tvary

Je-li tento parametr zvolen, bude během frekvenční analýzy využit slovník k rozpoznání stejných tvarů či variant jednoho slova a jejich výskyty budou následně sečteny. Zjednodušený příklad:

Ve zpracovávaných zprávách se nachází 2x slovo *mozek*, 1x slovo *mozku* a 3x slovo *mozkem*. Výsledky frekvenční analýzy pak budou následující:

Pokud je možnost *Nerozlišovat tvary* vypnuta:

- mozek: 2 výskyty
- mozku: 1 výskyt
- mozkem: 3 výskyty

Pokud je zapnuta:

- mozek, mozku, mozkem: 6 výskytů

Při použití této možnosti je frekvenční analýza relativně náročná na výpočetní výkon, proto může v závislosti na velikosti slovníku a množství zpracovávaných záznamů trvat delší dobu, na slabších počítačích i v řádu desítek sekund.

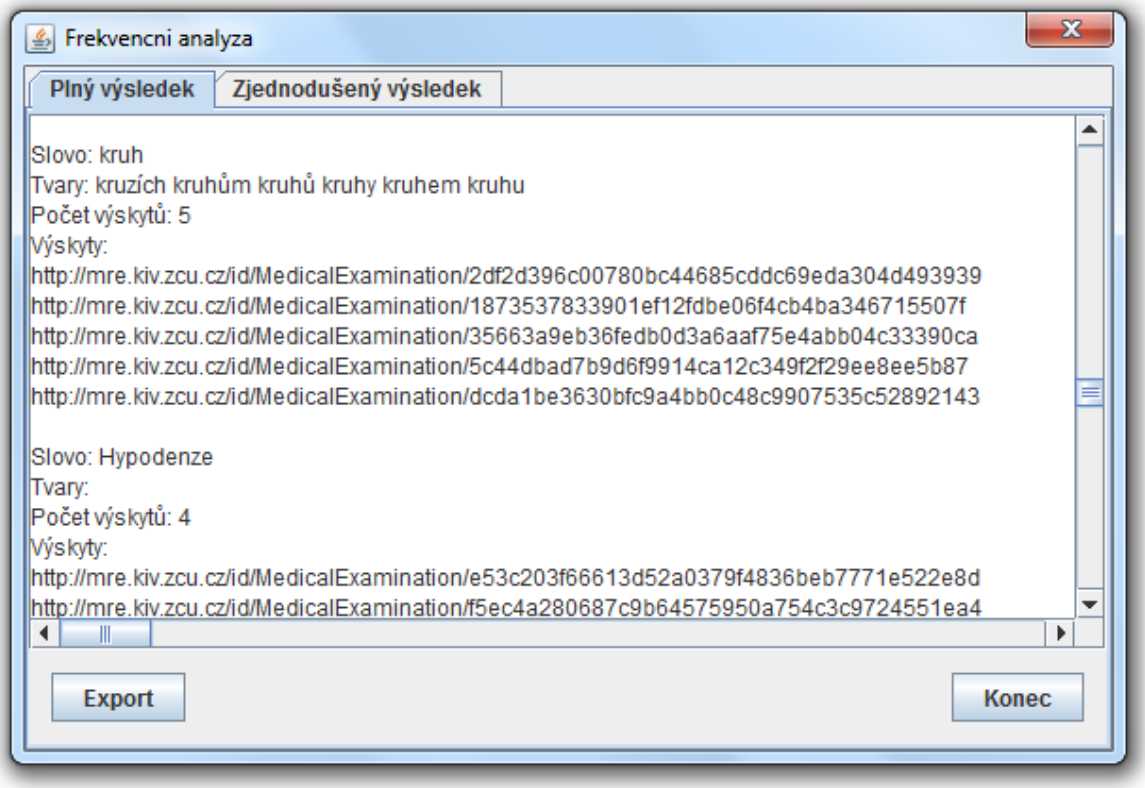
Vynechat stop slova

Je-li v programu načten seznam stop slov, je možné zvolením této možnosti aktivovat jeho použití. Slova v něm obsažená následně budou během analýzy ignorována a neobjeví se v jejích výsledcích.

Výsledky analýzy

Dialogové okno výsledků

Okno se zobrazí po proběhnutí analýzy. Je možné v něm přepínat mezi kompletním a zjednodušeným zobrazením výsledků. Kompletní zobrazení kromě počtu výskytů obsahuje i výčet všech tvarů a variant slova, které byly počítány společně (v případě analýzy s použitím slovníku) a URI záznamů, ve kterých se slovo vyskytlo. Zjednodušené zobrazení pak nabízí pouze jednoduchý přehled slov a počtů jejich výskytů. V obou případech jsou výsledky seřazeny sestupně dle počtu výskytů daného slova. Obě formy zobrazení výsledků je možné pomocí tlačtíka Export uložit do textového souboru pro archivaci či další využití.



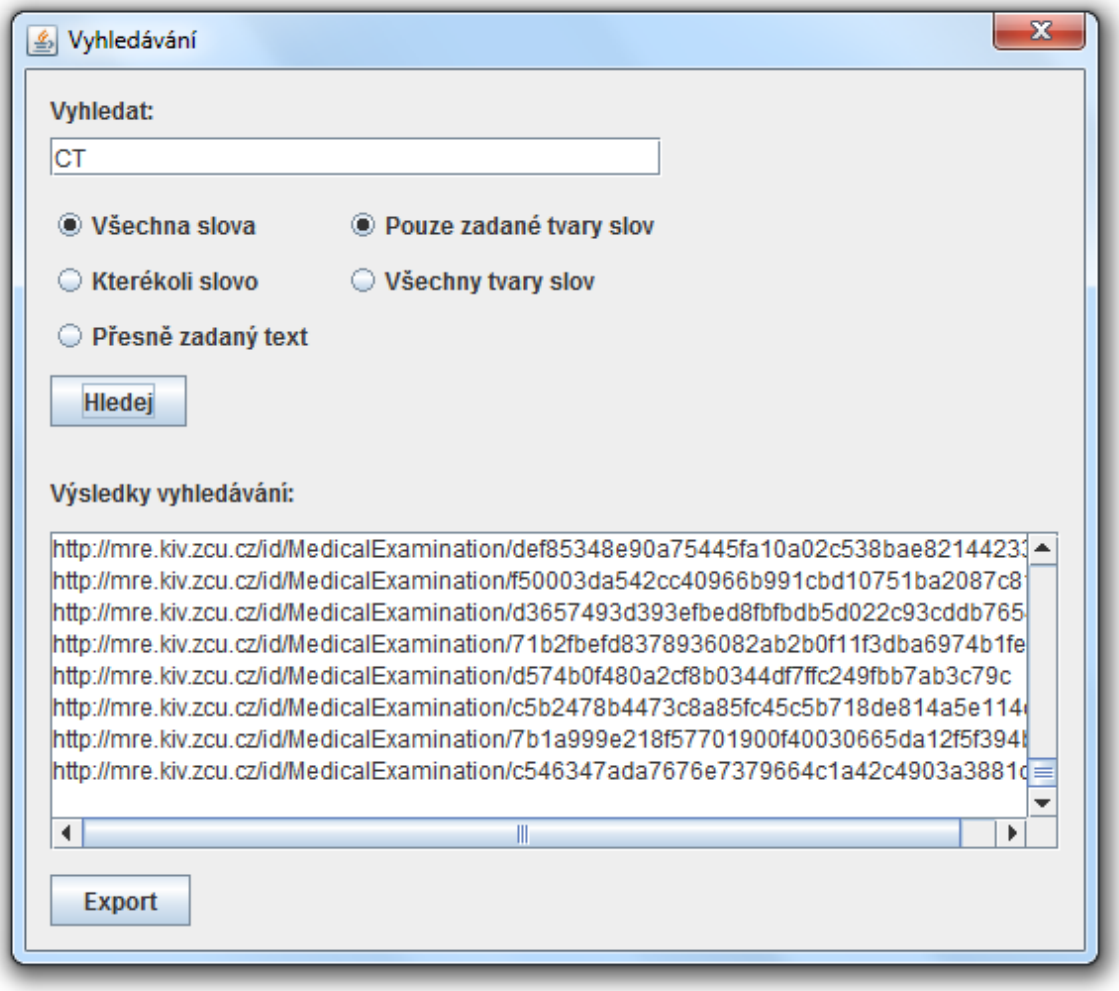
Obr. 8 – Dialogové okno výsledků frekvenční analýzy, zobrazující kompletní výsledky analýzy s použitím slovníku

Seřazení slovníku dle výsledků

Byla-li provedena analýza s použitím slovníku, program po uzavření okna s výsledky nabídne seřazení tohoto slovníku dle výsledků frekvenční analýzy. Slovník tedy bude namísto výchozího abecedního řazení seřazen dle počtu výskytů slov v analyzovaných zprávách. Slova s nulovým počtem výskytů jsou i nadále seřazena abecedně na konci slovníku. Toto seřazení je rovněž relativně výpočetně náročné a může trvat delší dobu, zároveň však může zkrátit časy provedení budoucích frekvenčních analýz, pokud budou analyzované zprávy podobné těm, podle jejichž frekvenční analýzy byl slovník seřazen. **Nezapomeňte, že po seřazení je nutno slovník uložit, jinak budou změny po ukončení programu ztraceny.**

Vyhledávání

Tato funkce umožňuje vyhledávat v právě načtených záznamech zadaná slova či kombinace slov v závislosti na zvolených parametrech. Při spouštění vyhledávání je třeba mít na paměti, že jeho funkčnost je založena na výsledcích frekvenční analýzy. Pokud tedy nebyla frekvenční analýza nad záznamy dosud provedena, může první spuštění vyhledávání trvat relativně dlouho v závislosti na výkonu počítače. Po spuštění vyhledávání v hlavním grafickém rozhraní se zobrazí dialogové okno vyhledávání:



Obr. 9 – Dialogové okno vyhledávání s ukázkou výsledků vyhledávání

Jako parametr vyhledávání je možné zadat libovolné slovo či kombinaci slov. Pomocí přepínačů lze nastavit parametry vyhledávání.

Pokud je zvolena možnost *Všechna slova*, jsou vyhledány jen ty záznamy, které obsahují kombinaci všech zadaných slov. Možnost *Kterékoli slovo* oproti tomu vyhledá všechny záznamy, které obsahují alespoň jedno ze zadaných slov. Oba tyto způsoby vyhledávání je možno ještě přizpůsobit volbou, zda mají být vyhledávány *Pouze zadané tvary slov* (pokud je zadáno např. „mozek“, jsou nalezeny pouze zprávy, obsahuící přesně slovo „mozek“) či *Všechny tvary slov* (při zadání „mozek“ jsou nalezeny i zprávy obsahující „mozku“, „mozkem“, „moz.“ a podobně).

Poslední možností je pak vyhledat *Přesně zadaný text*, kdy jsou nalezeny pouze zprávy, jejichž část se kompletně shoduje se zadaným textem, a to včetně číslic, interpunkčních znamének a podobně. Tento způsob vyhledávání je zároveň jediný, který lze použít pro hledání číselných údajů, předchozí režimy lze aplikovat pouze při hledání slov.

Výsledky vyhledávání jsou zobrazeny v podobě seznamu URI a je možné je exportovat do textového souboru pro archivaci či další využití.