Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Přenos dat z registru SITS

(Safe Impementation of Treatments in Stroke)

ve formátu DASTA

Plzeň, 2012 Bc. Pavel Karlík

Originální zadání práce

Prohlášení:

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, pokud není explicitně uvedeno jinak.

V Plzni dne ……

Bc. Pavel Karlík

……

Abstract

Data Transfer from the SITS Registry (Safe Implementation of Treatments in Stroke) in the Format of DASTA

The goal of this master´s thesis is to analyze the data standards in the field of medical applications with focus on the format DASTA, work with them and developing a suitable web-based tool for doctors in the FN Plzeň hospital to simplify the forms filling in the SITS registry. Application will be user-friendly with the focus on one-click access, multi-lingual, adaptive to changes, and fill in the forms with the data from hospital database stored in the DASTA. In the background will send same data also to the Department of Computer Science and Engineering in the University of West Bohemia for future research.

Poděkování

Tímto bych chtěl poděkovat paní Doc. Dr. Ing. Janě Klečkové za neutuchající optimismus a podporu, který mně byl motivací během celé doby mé práce. Dále si cením věnovaného času a pomoci, která mi byla velkou podporou při tvorbě. Poděkování patří také mé rodině a blízkým za jejich ohleduplnost a motivaci.

Obsah

1 Úvod

Doktoři z neurologických oddělení nemocnic po celém světě zadávají anonymní data o svých pacientech do mezinárodního registru SITS se sídlem ve Švédsku. Tento registr slouží pro sběr dat pacientů s mozkovými mrtvicemi a následnému výzkumu a analýzám vedoucím ke zlepšení akutní péče a léčby pro takto postižené osoby.

Problémem je ale rostoucí komplexnost daného registru, jeho uživatelská a jazyková nepřívětivost a také neexistující API pro napojení externích aplikací, které by mohly registr využívat jak pro vlastní analýzy, tak pro případné propojení s nemocničními systémy.

Z těchto faktů vychází potřeba tvorby aplikace pro FN Plzeň, která by usnadnila zadávání dat do tohoto registru.

Od aplikace vytvořené v této diplomové práci se očekává větší uživatelská přívětivost zaměřená na rychlé vyplnění klíčových dat, jednoduchá úprava a následná funkčnost i po případných změnách v registru SITS, podpora českého a anglického jazyka s možností jednoduchého přidávání a modifikace překladů, dále propojení s FN Plzeň a před-vyplnění již dostupných informací o pacientech z nemocniční databáze a nakonec také sběr zadávaných dat pro analýzy Medical Information Systém Research Group (MISRG) na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni.

2 Datové standardy

V této části se budu věnovat datovým standardům jako celku s detailnějším zaměřením na v aplikaci využitý datový standard Ministerstva zdravotnictví ČR DASTA.

2.1 Standardy obecně

Každé zdravotnické zařízení v ČR musí vést ze zákona zdravotnickou evidenci pacientů. Ta je ale dost často stále ještě vedena v papírové formě v kartotéce, což má několik nevýhod:

- **zálohování** – většina dokumentace existuje v jednom exempláři, při ztrátě nebo poškození dokumentace není možnost její obnovy
- **pomalé vyhledávání** – nutnost fyzického nalezení složky v kartotéce a dále nalezení konkrétního záznamu ve stohu papírů
- **výměna dokumentace** – dokumentace je prakticky dostupná pouze v místě uložení, jinak zdlouhavě kopírováním
- **komplikovaná statistika** – vzhledem k umístění je kvůli každé statistické veličině nutné fyzicky projít všechny složky
- **rozměrnost řešení** – kartotéka v nemocnicích běžně zabírá několik místností
- **neekologičnost** – spotřeba papíru

Tyto nevýhody se nezdají tak závažné, ale v návaznosti na akutní léčbu se jejich dostupnost či nedostupnost a rychlé či pomalé vyhledání dat pacienta stává zásadním problémem rozhodujícím o úspěšné léčbě pacienta, její délce, a v ojedinělých případech dokonce i o pacientově smrti.
Díky současné penetraci informačních technologií do všech oblastí, zdravotnictví nevyjímaje, je ale možné tato negativa odstranit či velmi dobře potlačit.

K obsluhování a hlavně udržování těchto dat slouží speciálně navržené aplikace od různých dodavatelů lišících se především podle určení:

- nemocniční informační systémy (NIS)
- laboratorní informační systémy (LIS)
- ambulantní informační systémy (AIS)
- informační systémy pro lékaře (IS PL)
- radiologické informační systémy (IS RDG)
- transfuzní informační systémy (HTS)¨
- lékárenské, patologické, dialyzační a jiné informační systémy

Problémem ale je, že každý výrobce si data pacientů ukládá v jiném formátu – není stanoven žádný jednotný standard jak uchovávat klinická data, což má za následek návrat jednoho ze zásadních problémů papírové kartotéky – výměna dokumentace, případně rychlé vyhledávání (například pokud se stane úraz a pacient je na akutním příjmu v cizím městě).

Typicky potřebujeme předávat tyto informace:

- identifikační údaje pacienta, adresy
- údaje o pojišťovně
- platební údaje
- důležité medicínské údaje pro včasnou léčbu – např. krevní skupina a alergie
- anamnéza, očkování, trvalé a aktuální diagnózy
- klinické události atd.

Protože jsou tyto zdravotnické informační systémy výnosným artiklem, je v tomto segmentu podnikání také mnoho konkurence, a tak není výjimkou, kdy má nemocnice rozdělené systémy mezi několik dodavatelů a problém výměny dat je už v samotném zařízení, natož na národní úrovni.

Aby si informační systémy mohly data vyměňovat, musíme specifikovat nějaký komunikační jazyk – protokol. Ten je nutné implementovat na obou stranách a pokud možno s co největší obecnostní. Zároveň musíme ale obsáhnout všechny možné stavy, které by si mohly systémy přenášet. Takovéto standardy výměny medicínských dat již nějakou dobu existují, ale problémem je jejich nejednotnost a absence hlavního datového standardu pro zdravotnictví.

V České republice například existují dva hlavní proudy, a to datový standard DASTA vyvinutý pro Ministerstvo zdravotnictví ČR nebo světový formát HL7 pocházející v USA.

2.1 DASTA

DASTA neboli DAtový STAndard Ministerstva zdravotnictví ČR je označení standardu pro uchovávání a výměnu medicínských dat spravovaným přímo oddělením informatiky Ministerstva zdravotnictví.
Na návrhu tohoto standardu se podíleli jak sami lékaři, tak dodavatelé zdravotnických informačních systémů souběžně s akademickou obcí a vznikl po důkladném prozkoumání všech v té době dostupných alternativ, včetně například výše zmiňovaného HL7.
Na vývoji pracuje Česká společnost zdravotnické informatiky a vědeckých informací České lékařské společnosti Jana Evangelisty Purkyně (ČLS JEP) pod záštitou Ministerstva zdravotnictví ČR.

2.1.1 Historie

Historie DASTA sahá do roku 1989, kdy vznikly první požadavky na datový standard pro Informační systém jednotek intenzivní péče (IS JIP). V roce 1992 začaly první kroky v realizaci, které spolu s návrhem z roku 1993 daly vzniknout první verzi DS 01.00

DS 01.XX

První verze DS 01.00 byla publikována ve věstníku Ministerstva zdravotnictví v částce 8-9 v listopadu 1994 pod názvem „Metodický návod MZ k datové struktuře pro přenos dat mezi informačními systémy zdravotnických zařízení, verze 1.00“ [1]
Tato verze se stala prvním důležitým krokem ke sjednocení komunikace mezi českými zdravotnickými zařízeními, avšak nebyla zatím plně použitelná pro nedostatek nadefinovaných datových bloků.

Následovala verze DS 1.10 publikovaná v červenci 1997, jejímž největším přínosem bylo zavedení Národního číselníku laboratorních položek (NČLP). S novými verzemi se seznam datových bloků rozšiřoval, až dospěla do finální verze DS 1.20, která však ale měla trvání pouze 2 roky - její platnost skončila k 31.12.2002.

Tato verze byla reprezentována souborem s datovými bloky. Každý soubor byl určen pro jednoho příjemce, jehož adresa je specifikována povinným blokem @PM. Údaje od více adresátů byla uvedená blokem @IS a stejně tak bylo možné předávat záznamy o více pacientech blokem @IP. Datové bloky byly obecně uvozovány znakem @ následující identifikátorem psaným velkými písmeny o maximální délce osm znaků.

Soubor byl zapisován textově do řádek dlouhých maximálně 255 znaků, jako oddělovač se používalo standardní odřádkování dle zvoleného kódování.

```
@VR
190820010700001I00581MR  NR
0.23
IP
0       0.6     0       1.2     0       0.91    0       1
S       ALT                     ukat/l      CATC

I:Rr-00061;A:L(Ebr Jaroslav RNDr.)
@
```

Obrázek 2.1 Ukázka datového bloku ve formátu DS 01.20 [2]

DS 02.XX

Tato verze platí od 1. 6. 2002 a její nejmarkantnější změnou je přechod od textového zápisu k XML formátu (eXtensible Markup Language), která výrazně zvýšila čitelnost datových souborů, dále možnost nalezení chyb a editaci pouhým textovým editorem bez nutnosti použití speciálního nástroje. Současně s těmito změnami byla inovována a doplněna řada datových bloků.
Pro komunikaci s laboratorními informačními systémy nyní nově používá Národní číselník laboratorních položek (NČLP), který bylo nutno pro toto využití upravit a rozšířit.

Další inovací bylo použití DTD (Document Type Definition), což je soubor pro snadné ověření správnosti daného XML.
Spolu s touto verzí byly vydány i nástroje pro práci s DASTA - například prohlížecí program s možností tisku nebo validační pomůcka.

DS 03.XX

Tato verze byla zveřejněna prostřednictvím věstníku Ministerstva zdravotnictví v částce 9 v červnu 2003 s platností od 1.1.2004.
Poprvé byla zachována vnitřní struktura z předchůdce. Změn se dostalo hlavně u vnitřní struktury bloků, které byly částečně upraveny případně doplněny. Dále byl samozřejmě také aktualizován i číselník laboratorních položek (NČLP), tentokrát na verzi 02.05.01.

Velkou změnou oproti předchůdci je ale podpora komunikace s Národním zdravotnickým informačním systémem (NZIS), jehož správcem a zpracovatelem dat je Ústav zdravotnických informací a statistiky ČR (UZIS). Podpora je zrealizována pomocí cca 60 bloků, které dokážou zjednodušeně řečeno serializovat formuláře. Ty s pomocí nástroje pro práci s formuláři UZIS „Program pro pořizování dávek NZIS“ uloží do souboru ve formátu DS a ten je poté možné zaslat do NZIS.

Vývoj této verze byl k 1.1.2007 ukončen, avšak bloky a číselníky jsou stále udržovány aktuální.

DS 04.XX

Nejnovější verze DASTA vznikla za souběhu s předchozí verzí datového standardu v prosinci 2006 a je platná od 1.1.2007.
Hlavní změna je opuštění validace pomocí DTD a přechod na použití XML schéma, které umožňuje definovat datové typy elementrů viz obrázek 2.2.

```
<!-- aType -->
<xs:complexType name="aType">
 <xs:sequence>
    <xs:element name="jmeno" type="dsComm:str255"/>
    <xs:element name="adr" type="dsComm:str35" minOccurs="0"/>
    <xs:element name="dop1" type="dsComm:str35" minOccurs="0"/>
    <xs:element name="dop2" type="dsComm:str35" minOccurs="0"/>
    <xs:element name="psc" type="dsComm:number9d" minOccurs="0"/>
    <xs:element name="mesto" type="dsComm:str48" minOccurs="0"/>
    <xs:element name="stat" type="dsComm:number3de" minOccurs="0"/>
    <xs:element name="vztah" type="dsComm:str35" minOccurs="0"/>
    <xs:element name="icl" type="dsComm:number8d" minOccurs="0"/>
    <xs:element name="as" type="ds:asType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="aAny" minOccurs="0" type="dsComm:AnyType"/>
  </xs:sequence>
  <xs:attribute name="typ" type="dsComm:typAdresyType" use="required"/>
  <xs:attribute name="ind_kont" type="dsComm:ind_kontType"/>
 <xs:attribute name="sr_typ">
   <xs:simpleType>
     <xs:restriction base="xs:string">
       <xs:enumeration value="L"/>
       <xs:enumeration value="O"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="sr_pois" type="dsComm:str4"/>
  <xs:attribute name="sr_kod" type="dsComm:str9"/>
</xs:complexType>
<!-- as Type -->
```

Obrázek 2.2 Ukázka typu adresa v XML schématu pro formát DS 04.01 [2]

Tato verze je v době vydání diplomové práce aktuální.

Národní číselník laboratorních položek (NČLP)

Vývoj začal v roce 1993, avšak součástí DS je až od roku 1997.

Číselník je souhrn a jednoznačná identifikace termínů použitých ve zdravotnictví a tím zabraňuje vzniku nedorozumění vzhledem k použití například lokálních synonym pro jiný termín.

Vychází z mezinárodní normy IFCC a je tvořen zástupci nejvýznamnějších klinických oborů s důrazem na precizní vyjádření a popis daného termínu a je průběžně doplňován. Aktualizace se provádí 4x do roka a nyní obsahuje přes 17 tisíc položek, z nichž tvoří například 5200 biochemie nebo 7000 imunologie.

Dále číselník umožňuje vkládání vlastních firemních a komunikačních bloků pro zahraničí, z čehož plyne větší variabilita použití, které se rozšířilo i na Slovensko (v informačních systémech českých i slovenských firem).

V EU se jedná zřejmě o nejrozsáhlejší obdobný číselník.

2.1.2 Struktura

Pokud není popsáno jinak, jedná se vždy o nejnovější verzi DS 4.XX.

Záznam v DS se skládá z datového souboru s přesně definovaným názvem obsahujícím jednotlivé datové bloky.

Na začátku záznamu je nutná deklarace XML s kódováním a použité verzi DS, v tomto případě se jedná o DS 02.01.01.

```
<?xml version='1.0' encoding='iso-8859-2' standalone='no' ?>
<!DOCTYPE dasta SYSTEM "ds020101.dtd">
```

Následuje hlavní blok datového souboru „dasta," který obsahuje povinné a volitelné kódy, mezi povinné patří například:

- id_soubor – jednoznačný identifikátor pro datový soubor, každý subjekt je povinen zajistit, aby byl skutečně unikátní. Tento řetězec musí začínat osmimístným kódem firmy, zpravidla bývá následován kódem programu, jeho verzí, časovým otiskem a dalšími proměnnými zajišťujícími jeho jednoznačnou identifikaci
- verze_nclp, verze_ds – kódy verzí použitých číselníků
- ur – zde určujeme typ přenášených dat. např:
    - S – urgentní data
    - R – rutinní zpracování
    - U – zprávy určené pro UZIS ČR
    - C – číselníky
    - B – laboratorní data
    - V – data o výkonech
    - H – hygiena
    - T – technická a testovací data

Po těchto povinných, a některých nepovinných, blocích je element dasta ukončen a následují klíčové datové bloky popsané viz níže, ve stejnojmenném odstavci.

Kódování

Čeština je dostupná od standardu 2.01.01 a výše v následujícím kódování:

- UTF-8

- Windows-1250
- ISO-8859-2 (aliass ISO 8859-2:1987, iso-ir-101, latin2, I2, csISOLatin2
- IBM852 (alias cp852, 852, csPCp852)

Název souboru

Pojmenování souboru je přesně dané a to jednou z následujících kombinací:

- UTTXXXXX.KKK – pro komprimované soubory
- UTTYYOOD.KKK – pro komprimované soubory určené pro UZIS ČR
- UTTXXXXX.xml – nekomprimované soubory
- UTTYYOOD.xml – nekomprimované soubory určené pro UZIS ČR

Význam jednotlivých zkratek:

- U – určuje urgentnost: S-urgentní, R-rutina, T-technický nebo testovací
- TT – určuje typ odesílajícího místo podle číselníku
- YY – rok, kterého se daný soubor týká (poslední dvojčíslí)
- OO – kód období dle číselníku CIS OBD(např. 01=leden, 21=1. čtvrtletí)
- KKK – komprimační nástroj: rar, zip, arj…
- XXXX – libovoný řetězec abecedních znaků nebo číslic bez mezer
- D – pořadové číslo dávky za období, slouží pro opravu případné doplnění dat

Datové bloky

Datové bloky tvoří základ DASTA, jsou popsány unikátním jménem v rámci celého standardu, stejně jako unikátním názvem XML elementu v souboru. Určuje jaké povinné a nepovinné informace bude daný blok obsahovat.

Definice těchto bloků se nalézá v textové podobě formou tabulek na stránkách DASTA [3]. Od verze DS 04.01 je možná také definice XML schématem, popřípadě dřívějším DTD. Oficiálně je však tabulková forma nadřazená všem ostatním.

Popis hlavních datových bloků:

- zdroj_js – zde najdeme kódy: kod_firmy, kod_programu, verze_prog určující jednoznačně zdroj dat
- pm – příjmové místo – místo kam zasíláme datový soubor – je možné vybírat z následujících kódů: ico, icl, icp, icz, pcz, oddel.

- is - jeden z nejdůležitějších bloků obsahující už přímo data o pacientech (v bloku ip). Další důležitou položkou je definice odesílatele datového souboru.
    - ip – blok definující jednoznačně pacienta – jsou zde položky jako id_pac, rodcis (rodné číslo), jmeno, prijmeni, sex (pohlaví – M pro muže a F pro ženy) a jiné

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dasta SYSTEM "http://medical.kiv.zcu.cz/data/ds031001.dtd">
- <dasta dat_vb="2012-01-11T11:57:14" potvrzeni="P" ozn_soub="99475" typ_odesm="NN" ur="R"
bin_priloha="T" verze_nclp="02.01.01" verze_ds="03.10.01"
id_soubor="MEDICALC_WMEXP_6.4.8.0_000000001499475">
    <zdroj_is verze_prog="6.4.8.0" kod_prog="WMEXP" kod_firmy="MEDICALC"/>
  - <pm icp="9999">
        <as typ="I"/>
    </pm>
  - <is>
      - <as typ="I">
            <obsah>Fakultní nemocnice Plzeň, NERV - iktová jednotka</obsah>
            <vnitrni>1255</vnitrni>
        </as>
      - <a typ="P">
            <jmeno>Fakultní nemocnice Plzeň, NERV - iktová jednotka</jmeno>
        </a>
      - <ip id_pac="13">
            <prijmeni>FNPL_13</prijmeni>
            <dat_dn format="D">1959-07-13</dat_dn>
            <sex>M</sex>
          - <a typ="1">
                <jmeno/>
                <psc>34601</psc>
                <mesto>Horšovský Týn</mesto>
            </a>
          - <dg dat_ab="2010-08-25T09:03:00">
              - <dgz ind_oprav_sd="N" typ_dg="P">
                    <diag poradi="1">I631</diag>
                    <dat_du typ="A" format="D">2010-08-26</dat_du>
                    <spec_dg>Mozkový infarkt způsobený embolií přívodných mozkových tepen</spec_dg>
                </dgz>
            </dg>
          - <z dat_ab="2010-08-25T10:47:59" oznaceni_o="43582522" stav="K" obsah="NL" vznik="H"
            zadost="D">
                <dat_du typ="A" format="DT">2010-10-26T01:30</dat_du>
                <nazev>004/000 - Anamnéza</nazev>
              - <text>
                    <ptext xml:space="preserve"/>
                </text>
            </z>
        </ip>
    </is>
</dasta>
```

Obrázek 2.3 Ukázka datového souboru ve formátu DS 03.10.01

3 Použité technologie

V této části se nalézá představení použitých technologií při tvorbě mé aplikace. Výčet je seřazen od nejdůležitějších komponent sestupně k nejméně využívaným. Konkrétní použití jednotlivých komponent a technologií bude podrobně popsáno v následující kapitole.

3.1 OWL

OWL je zkratkou z anglického Web Ontology Language, a je značkovacím jazykem vyvinutým organizací W3C (World Wide Web Consorcium) pro popis webových ontologií.

3.1.1 Ontologie

Pro pochopení si nejprve si vymezíme pojem ontologie.
V oblasti informatiky ontologie formálně reprezentuje popis určitého problému. Je to soubor pojmů a vztahů mezi nimi, který nám slouží k uchovávání a předávání znalostí dané problematiky.
Základní jednotkou ontologie je instance, pro příklad může sloužit jakýkoliv konkrétní i abstraktní objekt.
Třída je v tomto případě reprezentací množny určitého typu (např. savec), podtřídou se rozumí podmnožina (primát) atd. dokud se nedostaneme přímo ke konkrétnímu jedinci.
Každá třída může mít také definovatelné atributy, přičemž každý musí obsahovat minimálně název a hodnotu. A tím se dostáváme k samotnému principu ontologie, protože prozatím by nám na podobnou strukturu stačil obyčejný XML dokument. Hodnotou atributu totiž může být i odkaz na libovolný další objekt, třídu, a tím tvořit takzvané smyčky. Ontologie nám tak může popisovat až úplný graf oproti stromu, který popisuje XML. Shrneme si tedy výše v následující kapitole.

Požadavky pro jazyky ontologií

Ontologické jazyky umožňují uživatelům psát formálně i koncepčně přesné popisy doménových modelů.

Hlavními požadavky jsou:

- Jednoznačně a čitelně definovaná syntax
- Jednoznačně definovaná sémantika
- Efektivní podpora logiky
- Jednoduchost pro rychlé vyhledání informace
- Jednoduchost tvoření výrazů

Důležitost dobře definované syntaxe je jasná a známá z oblasti programovacích jazyků - je to nutná podmínka pro strojové zpracování informace. Samozřejmě je otázkou jestli syntax založená na XML-RDF je uživatelsky přívětivá – pro lidi samozřejmě existují lepší alternativy. Přesto tato nevýhoda není velice důležitá, protože valná většina uživatelů bude vyvíjet jejich ontologie používáním dalších nadstaveb, jako jsou knihovny, popřípadě celé programy, namísto ručního psaní v textovém editoru.

Formální sémantika popisuje přesně význam uložených informací. "Přesně"zde znamená, že sémantika se nevztahuje na subjektivní intuice, není ani otevřená různým (často odlišným) výkladům různých osob (nebo stroje). Význam formální sémantiky je dobře (až nadprůměrně) zaveden v oblasti matematické logiky.

Pro databází znalostí psanou formou ontologií můžeme uvažovat o:

- **Hierarchie tříd:** Jestliže x je instancí třídy C a třída C je podtřídou D, potom můžeme uvažovat, že x je instancí třídy D
- **Ekvivalence tříd:** Jestliže třída A je ekvivalentní třídě B a třída B ekvivalentní k třídě C, potom je také A je ekvivalentní k C.
- **Konzistence:** Předpokládejme, že jsme deklarovali x jako instanci třídy A. Dále předpokládejme, že:
  - Je podtřídou B ∩ C
  - Je podtřídou D
  - B a D jsou disjunktní

Pak máme rozpor, protože A by měla být prázdná, ale má instanci x. To je údaj o chybě v ontologii.

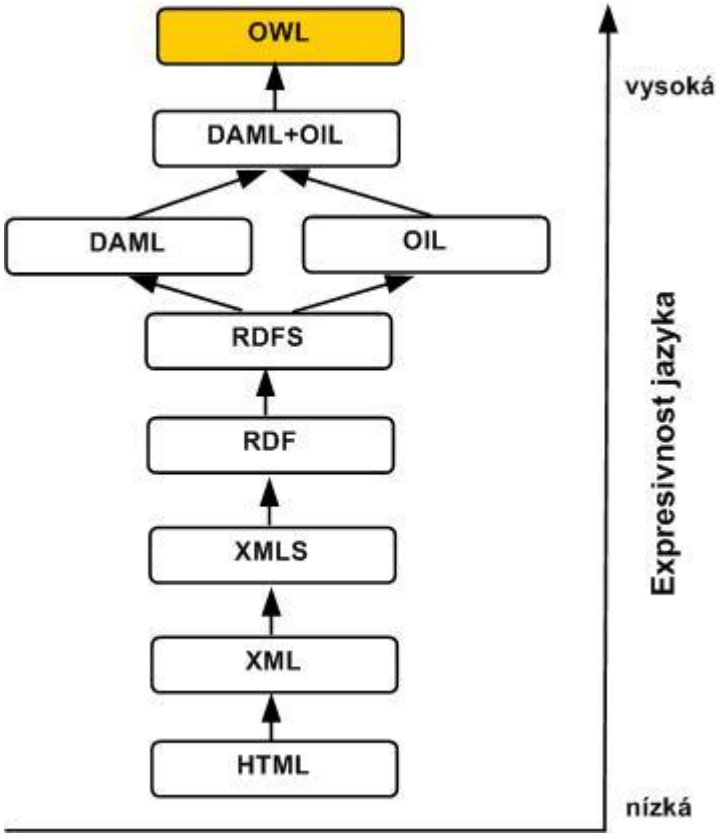
• **Klasifikace**: Pokud jsme deklarovali, že některé páry vlastnost-hodnota jsou dostačující podmínkou pro členství ve třídě A, potom když jednotlivec x splňuje tyto podmínky, můžeme učinit závěr, že x musí být instancí A.

3.1.2 Historie

OWL 1

Práce na vytvoření nového jazyka pro popis ontologií započaly na přelomu tohoto tisíciletí. První verze ontologie OWL standardu W3C vznikla následně v listopadu 2002 jako pracovní návrh, ze kterého se po necelých dvou letech v únoru 2004 stal uznávaný standard organizace W3C.

OWL byl vyvíjen z jazyka – DAML+OIL s požadavkem vytvořit reprezentaci znalostí zejména pro prostředí webu. Cílem bylo vytvoření podrobnějšího jazyka s bohatší sémantikou, než tomu bylo u všech jeho předchůdců a zároveň zachování čitelnosti díky formě využívající vyjadřovací prostředky jazyku RDF a RDFS s rozšířením o vlastní výrazy.



Obrázek 3.1 Cesta k OWL [4]

OWL 2

Vývoj nástupce začal v březnu 2009 prvním pracovním návrhem, ze kterého vznikla plnohodnotná a použitelná verze v říjnu 2009.

Rozdíly oproti předchozí verzi nejsou velké – takřka všechny části z OWL2 jsou přítomné i v předchozí verzi, i když pod jinými jmény. Došlo k jemné úpravě syntaxe a přiblížení k sémantice RDF Graph.

3.1.3 Struktura

Jazyk OWL se skládá ze tří dialektů, které se odlišují zejména rozsahem:

- OWL-Full
- OWL-DL
- OWL-Lite

OWL-Full

Plná verze jazyka OWL, obsahuje všechny elementy a konstrukce, plně kompatibilní s RDF. Syntakticky je tedy nejpreciznější - nemá žádná omezení, ale díky tomu je také výpočetně nejnáročnější a nejpomalejší.

OWL-DL

Kompromis mezi oběma verzemi, kde se snažíme o rozumný poměr mezi vyjadřovací silou a výpočetní náročností, není již plně kompatibilní s RDF.

OWL-Lite

Nejjednodušší verze, doporučovaná pro jednodušší struktury, kde je přínosem spíše efektivnější zpracování, než syntaktická přesnost. Neumí například sjednocení prvků, doplněk prvku, nebo disjunkci tříd.

Na následujícím obrázku je ukázka části struktury OWL ve formátu RDF/XML. Po nezbytných úvodních hlavičkách vidíme třídu *CheeseTopping* (sýr navrch), která je disjunktní s třídami jako *VegetableTopping* (zelenina navrch) atd. Jedná se o příklad

objednávkového systému z italské restaurace, kde vylučujeme, které potraviny se k sobě na pizzu hodí, a které ne.

```
<?xml version="1.0"?>
<rdf:RDF xml:base="http://labe.felk.cvut.cz/~obitko/spr/pizza-spr.owl"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns="http://labe.felk.cvut.cz/~obitko/spr/pizza-
spr.owl#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#">
<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365) http://protege.stanford.edu -->
  - <owl:Ontology rdf:about="">
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Pizza ontology adapted
            for SPR; for original see http://www.co-
            ode.org/resources/tutorials/ProtegeOWLTutorial.pdf</rdfs:comment>
    </owl:Ontology>
  - <owl:Class rdf:ID="CheeseTopping">
      - <owl:disjointWith>
            <owl:Class rdf:ID="VegetableTopping"/>
        </owl:disjointWith>
      - <owl:disjointWith>
            <owl:Class rdf:ID="SeafoodTopping"/>
        </owl:disjointWith>
      - <owl:disjointWith>
            <owl:Class rdf:ID="MeatTopping"/>
        </owl:disjointWith>
      - <rdfs:subClassOf>
            <owl:Class rdf:ID="PizzaTopping"/>
        </rdfs:subClassOf>
    </owl:Class>
  - <owl:Class rdf:ID="InterestingPizza">
      - <owl:equivalentClass>
          - <owl:Class>
              - <owl:intersectionOf rdf:parseType="Collection">
                  - <owl:Restriction>
                        <owl:minCardinality
                            rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</owl:minCardinality>
                      - <owl:onProperty>
                            <owl:ObjectProperty rdf:ID="hasTopping"/>
                        </owl:onProperty>
                    </owl:Restriction>
                    <owl:Class rdf:ID="Pizza"/>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>
```

Obrázek 3.2 Ukázka části souboru OWL [5]

3.1.4 Protégé

Protégé je volně šiřitelná (open-source) platforma poskytující nástroje pro konstrukci, vizualizaci a manipulaci s ontologiemi v několika různých formátech. Je vyvíjena v Stanford Center for Biomedical Informatics Research na Standford University School of Medicine, USA v kooperaci s University of Manchester, UK.

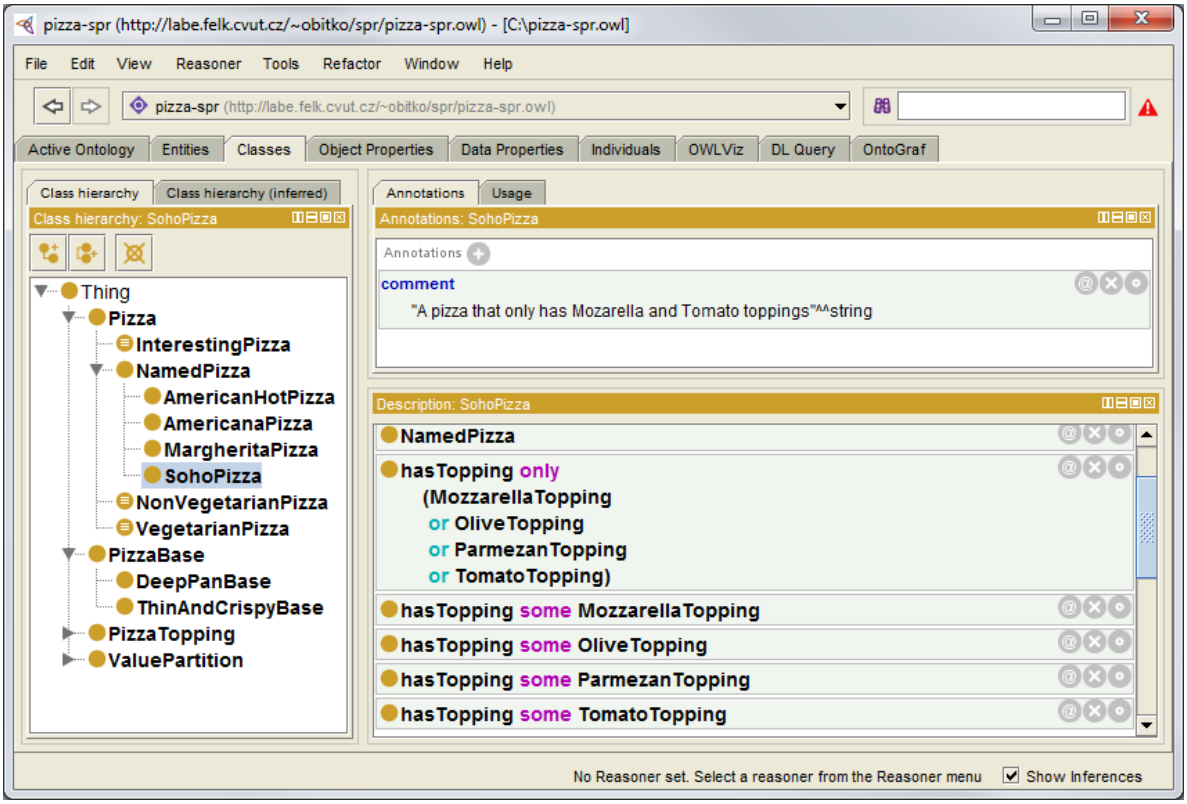
Platforma podporuje dva typy modelování ontologií:

- Protégé-Frames – podpora takzvaných frame-based ontologií dle standardu Open Knowledge Base Connectivity protocol (OKBC)
- Protégé-OWL – použití pro práci s ontologií pro sémantické vyjádření webu podle standardu OWL

Protegé je vytvořeno v programovacím jazyku Java s plnou podporou pluginů, což z něj tvoří základ pro vytváření vlastních aplikací, samozřejmě s přihlédnutím k licenci Mozilla Public License (MPL). Vzhledem k rozšířenosti a dominantnímu postavení má komunita kolem této platformy velmi širokou základnu tvořící cca 70 tisíc vývojářů především z akademické, ale i vládní a komerční sféry.
Díky tomuto rozšíření je k dispozici široká podpora a databáze znalostí, včetně podrobných návodů pro nejpoužívanější problémy.

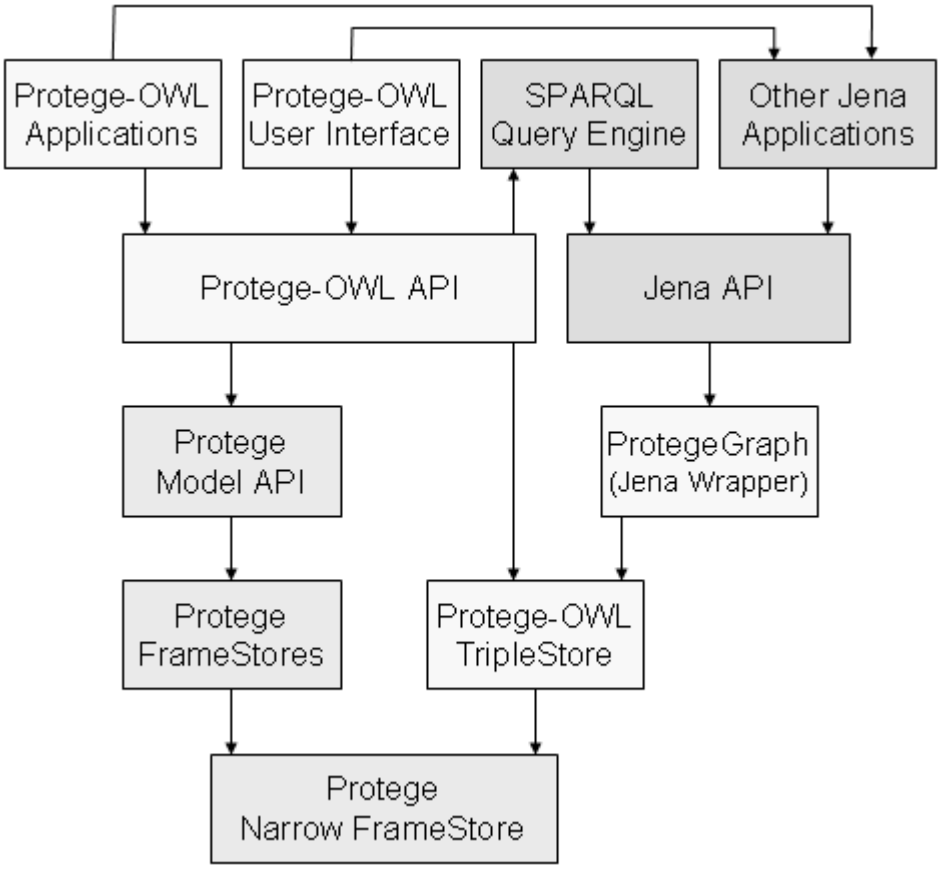
Protégé-OWL editor umožnuje uživatelům:

- Načítat a ukládat OWL a RDF ontologie
- Upravovat a vizualizovat třídy, vlastnosti a pravidla SWRL
- Definovat logické charakteristiky tříd jako výrazy OWL
- Spouštět odvozovače (reasoners)
- Editovat OWL individua (individuals) pro sémantické značkování webu



Obrázek 3.3 Ukázka předchozího OWL souboru otevřeného v Protégé

Protegé je silně navázáno na následující použitou technologii Apache Jena viz následující obrázek.



Obrázek 3.4 Struktura propojení Protégé s frameworkem Jena [6]

3.2 Apache Jena

Apache Jena je volně šiřitelná knihovna pro budování sémantických webových aplikací vytvářená Apache Software Foundation.

3.2.1 Historie

Vznik knihovny Jena sahá do roku 2000 do Hewlett Packard Labs, Bristol, UK, kde byla vyvinuta prvotní verze. Již od počátku se jednalo o volně šiřitelný (open source) projekt, který byl ale v roce 2009 vzhledem k přehodnocení priorit HP opuštěn. Rokem 2010 tedy došlo k přesunu vývojového týmu do inkubátoru Apache Software

foundation, kde byla letos povýšena do elitní skupiny Apache produktů jako takzvaný top-level project. Její vývoj stále pokračuje.

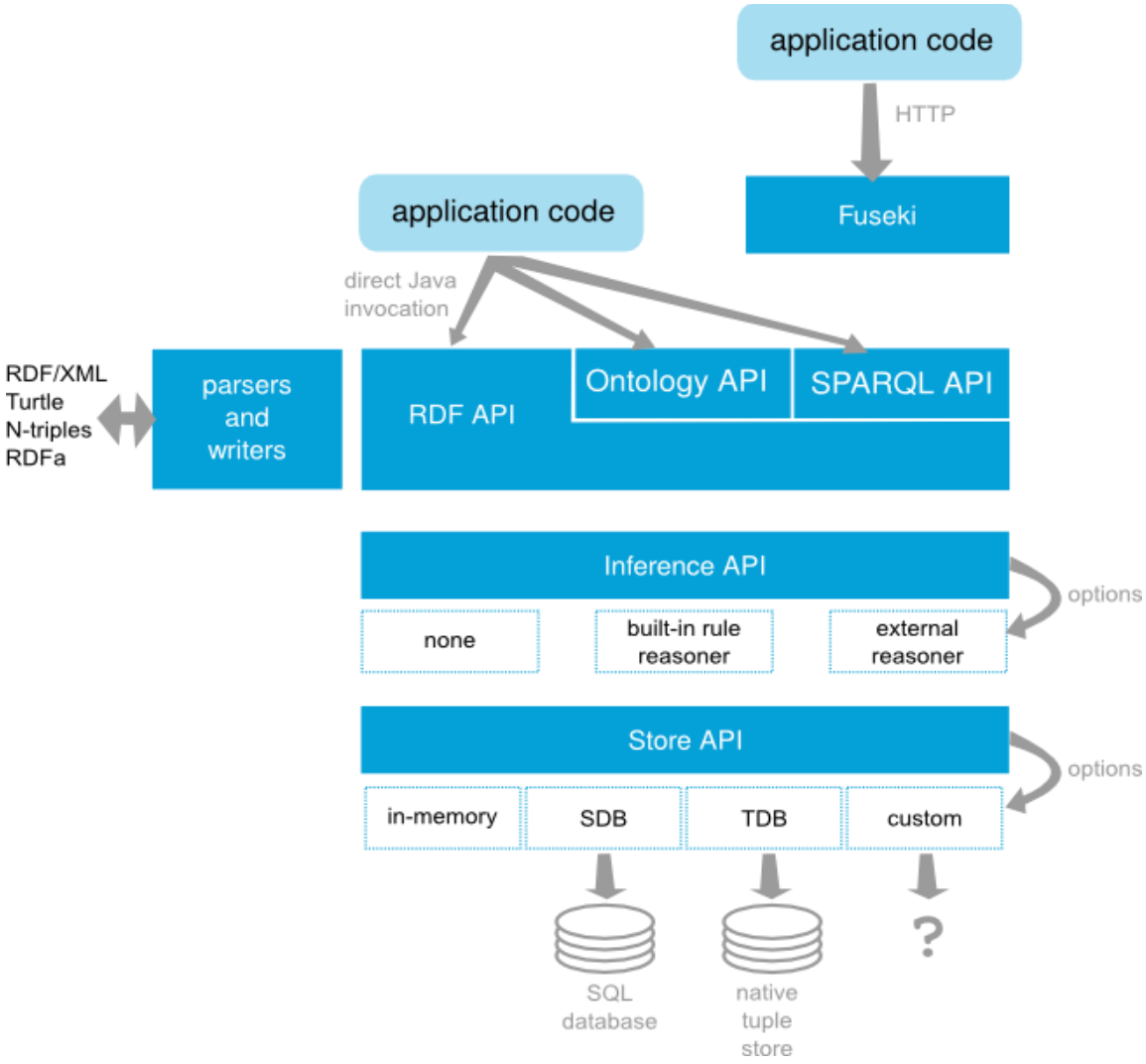
3.2.2 Struktura

Jena poskytuje soubor nástrojů a knihoven pomáhající tvorbě ontologií, sémanticky propojených aplikací, nástrojů a serverů.

Jena framework obsahuje:

- API pro čtení, zpracování a zápis RDF dat v XML, N-Triple (trojice) a Turtle formátu
- API pro přístup k ontologiím OWL a RDFS
- odvozovače a vyhodnocovače pravidel z dat RDF a OWL
- efektivní ukládání velkého množství RDF trojic na pevném disku
- dotazovací rozhraní s nejnovější specifikací dotazovacího jazyka ontologií SPARQL
- umožňuje serverům posílat RDF data jiným aplikacím prostřednictvím různých protokolů, SPARQL nevyjímaje.

Jena je široce rozšířená s více než 200 000 staženími a je používaná mnohými velkými společnostmi v jejich programových řešeních.



Obrázek 3.5 Struktura frameworku Jena [7]

3.3 Apache Struts 2

Apache Struts 2 je volně šiřitelné (open-source) řešení pro vývoj webových (enterprise) řešení založených na platformě Java EE vyvíjené pod Apache Software foundation pod stejnojmennou licencí.

3.3.1 Historie

Struts 1

Historie tohoto řešení sahá do roku 2001, kdy vyšla první verze Struts 1. Během krátké doby se tak stal Struts jedním z nejpopulárnějších frameworků a dodnes se o něm mluví jako o standardu pro webové aplikace vyvíjené v programovacím jazyku Java.

Struts 1 má velmi širokou členskou základnu, výbornou dokumentaci a existuje mnoho návodů na více či méně komplikovaná řešení. Většina později vytvořených frameworků se jím při svém vývoji inspirovala a snažila se napravit jeho nedostatky.

Struts 2

Tak také postupně vznikl jeho nástupce – Stuts2. V roce 2002 byl publikován framework WebWork, jehož cílem bylo právě opravení chyb v Struts 1 a využití dobrých nápadů z ostatních aktuálně dostupných řešení. Vzhledem k tomu, že WebWork již implantoval téměř vše, co bylo v plánu v nové verzi Struts, dohodli se oba vývojářské týmy na spojení, a tak v roce 2006 vznikl nový framework Struts 2.

Výhody, které Struts 2 přináší oproti předchozí verzi:

- zrušení nutnosti použít speciální třídy rozšiřující rozhraní ActionForm. Tím došlo k zjednodušení práce a zpřehlednění kódu
- zjednodušení akčních tříd – tou se nyní může stát jakákoliv třída obsahující metodu String execute()
- zjednodušená správa doplňků pouhým přidáním JAR archivu připojenému k aplikaci v dohodnutém formátu
- vystižnější a kratší názvy, díky kterým je framework lépe zvládnutelný pro začátečníky, naopak pro uživatele přecházející z předchozí verze je díky tomu práce složitější
- podpora AJAXu
- podpora značkovacího jazyku OGNL (nahrazení JSTL)
- jednoduchá integrace do Java Spring
- lepší implicitní nastavení, které vyhovuje většině uživatelů a není tak potřeba framework složitě nastavovat
- vylepšení a rozšíření knihovny značek

3.3.2 Struktura

Struts 2 vychází z návrhového vzoru MVC, který striktně rozděluje projekt na 3 části:

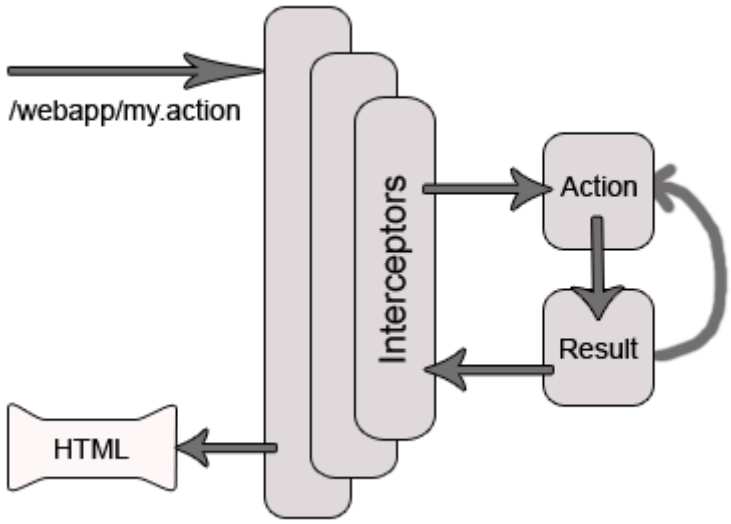
- Model – přístup k datům
- Controller – logická třída starající se o obsluhu událostí a operace nad modelem
- View – zobrazovací část s kterou komunikuje uživatel

View vrstva je nejčastěji reprezentovaná jako Java Server Pages (JSP) podporovaná knihovnou značek.
Controller je uvnitř reprezentována servletem, který zachytává požadavky z prohlížeče klienta a rozhoduje o dalších akcích. Toto je vyjádřeno v konfiguračním souboru struts.xml pomocí jasně definovaného vzoru.
Třída Model může být dle uvážení reprezentována jakoukoliv strukturou umožňující přístup k datům. V současné době například Java Persistence API (JPA) popřípadě stále populárnější Hibernate nebo jako v mém případě Apache Jena.

Životní cyklus jednoho požadavku odeslaného uživatelem v rámci Struts 2 by se dal shrnout následovně:

- uživatel zasílá požadavek na server prostřednictvím např. webového prohlížeče
- FilterDispatcher rozhodne o odpovídající akci
- přerušovače (interceptors) rozhodnou, zdali je zaslaný požadavek validní
- akční třída vybraná pomocí FilterDispatcher zareaguje na požadavek a vygeneruje odpověď
- odpověď je zaslaná zpět přes přerušovače k uživateli, to nám dovoluje dodatečné zpracování popřípadě vyčištění výsledku
- zobrazení výsledku uživateli

Uvedený cyklus nám vystihuje následující obrázek:



Obrázek 3.6 Životní cyklus požadavku v Apache Struts 2 [8]

3.4 HTML Parser

Jedná se o Java knihovnu používanou k práci s HTML kódem, která je primárně určená pro extrakci a modifikaci stránky podle zadaných filtrů, vlastních tagů a snadném propojení s JavaBeans. Je rychlá, robustní a velmi dobře otestovaná.

Extrakce umožňuje:

- extrakce textu – používáno např. webovými vyhledávači
- extrakce hypertextových odkazů – pro procházení kompletní struktury stránek nebo např. sbírání e-mailových adres
- tzv. screen scraping – programové získávání hodnot ze stránek
- extrakci souborů použitých na stránce, sběr obrázků, zvuků…
- kontrola validity odkazů
- monitorování stránek pro případné změny

Transformace umožňuje:

- modifikaci URL v libovolných elementech na stránce
- uložení celé stránky na lokální úložiště

- cenzurování obsahu – odstranění nevhodných slov a frází
- odstranění chyb v HTML kódu
- blokování reklam na základě URL
- konverzi existujících stránek do XML

Výhodou této knihovny je použití vlastního parseru, díky kterému dokáže pracovat i s poškozeným HTML kódem a nedojde k selhání.

3.5 JTidy

JTidy je portace oblíbené knihovny HTML Tidy do programovacího jazyka Java. Je používaná pro kontrolu správné syntaxe HTML kódu a jeho formátování.
JTidy poskytuje DOM parser, takže je možné ji také využít podobně jako výše uvedenou knihovnu.
Liší se ovšem v menším zaměření na extrakci dat a práci s HTML kódem a klade větší důraz na jeho formátování, které lze precizně nastavit.

3.6 Apache HttpClient

Apache HttpClient je volně šiřitelná knihovna, která je součástí balíku Apache HttpComponents vyvíjená stejnojmennou společností od roku 2001. Jedná se o Java knihovnu sloužící pro pokročilou komunikaci prostřednictvím HTTP protokolu. Obsahuje plnou implementaci všech metod (GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE) v objektově orientovaném návrhu. Dále podporuje šifrování https (http s SSL) a dokáže pracovat s cookies a sessions.

3.7 Ostatní použité technologie

3.7.1 JSP

Java Server Pages neboli JSP je technologie původně vyvinutá společností Sun Microsystems v roce 1999, nyní patřící pod Oracle. JSP je obdoba PHP či ASP

používající programovací jazyk Java. K použití je potřebný kompatibilní web server s kontejnerem pro servlety (ty jsou generovány pomocí JSP, rozdíl mezi nimi je, že JSP implementuje Java kód do HTML, zatímco servlet naopak). Nejtypičtějšími web serverem pro toto použití je např. Apache Tomcat.
Jedná se prakticky o HTML, do kterého je pomocí JSP značek vložen kód Java, který je dále zpracováván na straně serveru, protože je z něj vygenerován automaticky servlet. Ten může dále využívat třídy napsané v Javě, a tak umožňuje rychlý vývoj aplikací a částečné oddělení designu od výkonného kódu. JSP také umožňují použití takzvaných sessions, které uchovávají data u konkrétního uživatele a sdílí je u všech požadavky mezi klientem a serverem.
Struktura JSP stránky se nijak neliší od HTML, pouze přípona souboru je pro odlišení .jsp. Do stránky je možné poté vkládat takzvané skriptlety, což jsou části Java kódu oddělené vložené mezi tyto znaky <% a %>. Ukázka jednoduchého skriptletu:

```
<%     //Vystup do konsole
       System.out.println("Hello World - Console!");
       //Vystup do HTML
       out.println("Hello World !");
%>
```

Dále je možné použití JSP jako nástroji pro vkládání jiných stránech pomocí tagu jsp:include, použití:

```
<jsp:include page="relativeURL | <%= expression %>" flush="true " />
```

Posledním příkladem funkčnosti je novinka ze specifikace JSP 2.0 a tou je přístup přímo k proměnným v Java třídách:

```
Hodnota "promenna" v objektu "javabean" je ${javabean. promenna}.
```

3.7.2 JSTL

Knihovna značek JSTL neboli Java Server Pages Standard Library byla vydána v roce 2006 pod tzv. Java Community Process. Rozšiřuje funkčnost JSP stránek o práci se soubory, zpracování XML dokumentů, programovou logiku, cykly anebo např. podporu více jazyků.

Pro práci je potřeba vždy na začátku souboru načíst odpovídající knihovnu následujícím způsobem:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Dále je možné JSTL tagy používat např. takto (cyklus vypíše od 1 do 10):

```
<c:forEach var="i" begin="1" end="10">
            <c:out value="${i}" />
            <br/>
</c:forEach>
```

3.7.3 Webový server Apache Tomcat

Apache Tomcat (formálně Jakarta Tomcat) je volně šiřitelný web server a servletový kontejner vyvíjený Apache Software Foundation od roku 1999. Implementuje JSP a Java Servlet specifikace a vzhledem k jeho jednoduchosti a open-source licenci se stal jedním z nejpoužívanějších web serverů. Je konkurencí komerčním produktům jako je WebLogic aplikační server od Oracle popřípadě IBM WebSphere.

3.7.4 Javascript

Javascript (JS) je objektově orientovaný skriptovací jazyk původně vyvinutý společností Netscape v roce 1995 a od roku 1996 je světovým standardem Ecma International. Jeho primární použití je na HTML stránkách pro logiku na klientské straně (tzv. client-side), ale využití nachází i v aplikacích jako jsou např. PDF dokumenty.
Jedná se o jeden z nejpopulárnějších programovacích jazyků používaných na webu, který je díky rozšíření Ajax (Asynschonous JavaScript and XML) populární i u profesionálních projektů.

3.7.5 SPARQL

SPARQL (neboli rekurzivně SPARQL Protocol and Rdf Query Language) je dotazovací jazyk nad RDF daty, který s nimi dokáže pracovat podobně jako dotazovací jazyky databází. Od roku 2008 je SPARQL v 1.0 standardem World

Wide Web Consorcium a W3C a je jednou z klíčových technologií sémantického webu.

Ukázka jednoduchého dotazu:

```
SELECT
  ?id ?birth
WHERE
{
  ?s rdf:type ds:Patient   .
  ?s ds:id_pac ?id .
}
ORDER BY ASC(?id)
```

Tímto dotazem vypíšeme všechny pacienty, kteří mají v RDF dokumentu zadané parametry id a birth a seřadíme je vzestupně podle hodnoty id.

3.8 Shrnutí a výběr platforem

Použití výše uvedených frameworků a knihoven bylo z valné většiny dáno samotným problémem, jelikož šlo o navázání aplikace na již rozběhnuté projekty. Dále byl výběr ovlivněn předchozími zkušenostmi autora s uvedenými technologiemi.

Vstupní data z FN Plzeň byla ve formátu DASTA.

Údaje posílané do registru SITS byly na ZČU již reprezentovány ontologií ve formátu OWL, její použití tedy bylo nutností a z tohoto vycházelo také využití programu Protégé.

Dále pro napojení programovacího jazyka Java na OWL a DASTA neexistuje prakticky jiná varianta než Apache Jena, z toho také plyne použití dotazovacího jazyka SPARQL, který je její součástí.

Upřednostnění Apache Struts 2 oproti předchůdci bylo umocněno zjednodušením a výhodami popsanými výše, přestože měl autor již zažité zkušenosti se Struts 1.

Webový server Tomcat byl zvolen pro jeho integraci ve vývojovém prostředí NetBeans.

Poslední tři knihovny HTML Parser, HTTPClient a JTidy byly vybrány pro jejich jednoduchost, rychlost a uživatelskou rozšířenost.

4 Návrh aplikace

V této kapitole je podrobně rozebrána specifikace a návrh aplikace.

4.1 Specifikace

Autor měl navrhnout webovou aplikaci, která by propojila nemocniční databázi s mezinárodním registrem mozkových mrtvic SITS (Safe Impementation of Treatments in Stroke) a usnadnila tak zadávání dat lékařům z FN Plzeň. Zároveň by ukládala data pro analýzy Medical Information Systém Research Group (MISRG) na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni.
Dále musí být aplikace plně škálovatelná, aby mohla pružně reagovat na případné modifikace zadávaných dat v SITS.
Posledním požadavkem je podpora více jazyků, s možností jednoduché opravy případné přidání nových překladů.

4.2 Funkční požadavky

- Do webové aplikace bude přistupovat vždy jen jeden uživatel
- Přístup uživatelů nebude nijak autentizován
- Přidávání – zadávání nových pacientů do registru SITS
- Editace – možnost dodatečných úprav již vložených pacientů
- Přidávání a editace dat v jednotlivých formulářích u konkrétního pacienta
- Přehled stavu vyplnění formulářů u více pacientů spolu s rychlým přístupem pro jejich okamžité úpravy na jedné stránce
- Uzamykání již kompletně vyplněných formulářů
- Podle ID pacienta předvyplnit dostupné údaje z databáze FN Plzeň
- Umožnit přepínání mezi českou a anglickou verzí

4.3 Vedlejší požadavky

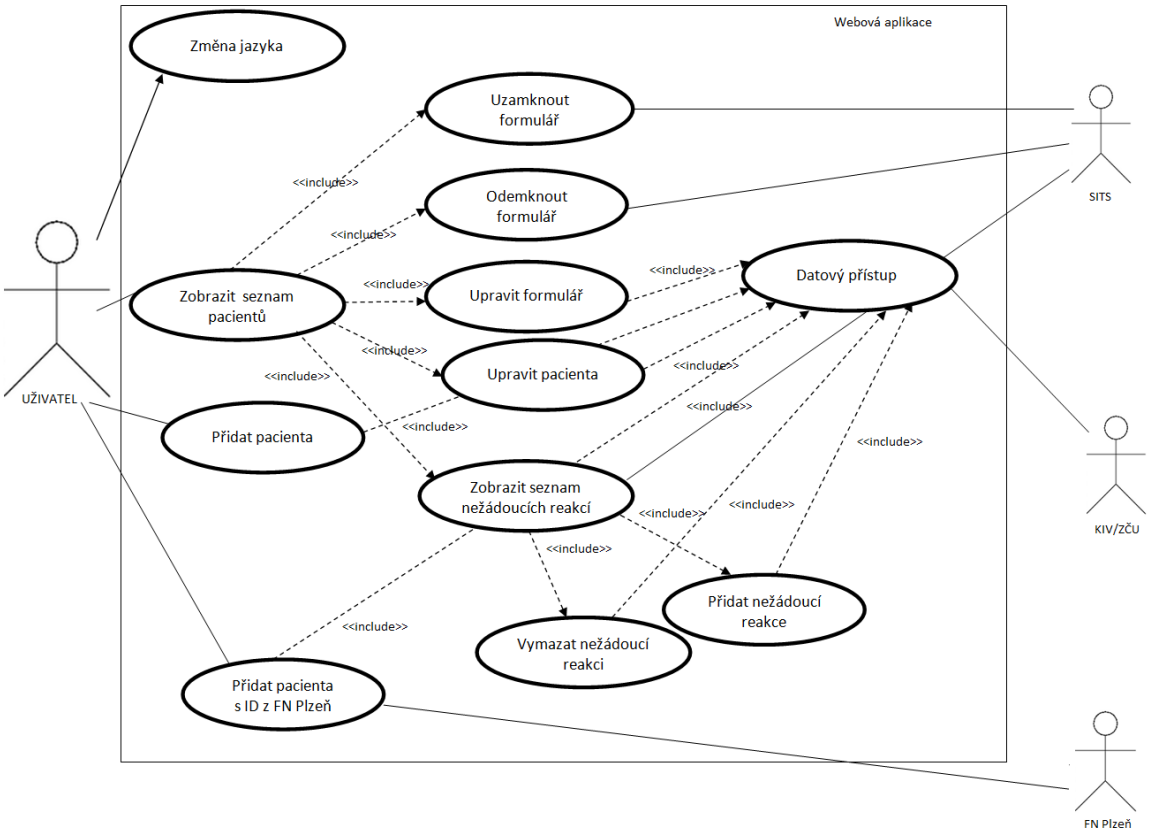
- Škálovatelnost – zajištěna díky generování formulářů z ontologie OWL
- Data z FN Plzeň budou načítána z Oracle databáze ve formátu DASTA
- Data budou také paralelně ukládána na Katedře informatiky ZČU podle ontologie OWL
- Webová aplikace bude vybudována pomocí Java EE

4.4 Uživatelské role

Vzhledem k zaměření systému existuje v aplikaci pouze jedna uživatelská role.
Globálně existuje ještě role administrátora ontologie OWL, ke které je řízen přístup serverem http://mre.kiv.zcu.cz, na kterém je umístněna. Uživatel s přístupovými právy na tento stroj může za pomocí programu (např. editor Protégé) měnit ontologii a tím i jednotlivé prvky formulářů, jejich skladbu a jazykové mutace.

4.5 Use case diagram

Na základě specifikace a požadavků byl vytvořen následující Use case diagram. Díky jeho komplikovanosti nebylo zohledněno vytváření webových stránek z ontologie OWL pro zlepšení celkové čitelnosti. Toto generování se týká všech procesů v diagramu kromě změny jazyka a zobrazení seznamu pacientů.
Zároveň je zde ze stejného důvodu zjednodušená komunikace s registrem SITS, která nutně musí probíhat obousměrně (formuláře je nutné nejdříve vyplnit již zadanými hodnotami z minulosti a pak se odesílají upravené zpět na server SITS).



Obrázek 4.1 Use case diagram aplikace

Název případu užití	**Zobrazení seznamu pacientů**
Identifikace případu užití	UC01
Aktéři	Uživatel, SITS
Kroky případu užití	1a. Uživatel zadá požadavek na zobrazení úvodní stránky 1b. Uživatel v menu vybere položku Moji pacienti 2. Aplikace se připojí do registru SITS a z jeho webových stránek vyfiltruje údaje o pacientech a stavech vyplnění jejich formulářů 4. Aplikace uživateli zobrazí seznam pacientů spolu se stavem vyplnění jednotlivých formulářů v přehledné tabulce
Výstupní podmínky	Uživateli je zobrazen seznam pacientů

Název případu užití	**Přidání pacienta**
Identifikace případu užití	UC02

Aktéři	Uživatel, SITS, KIV/ZČU, FN Plzeň
Kroky případu užití	1. Uživatel otevře úvodní stránku 2a1. Uživatel přidá pacienta bez vyplnění ID. 2b1. Do pole „Přidat pacienta“ vyplní uživatel ID pacienta z FN Plzeň. 2b2. Aplikace vyhledá ID, a pokud ho nalezne, předvyplní dostupné údaje do formuláře. 3. Systém zobrazí formulář přidání nového pacienta vygenerovaný z ontologie OWL na serveru Katedře informatiky. 4. Uživatel vyplní potřebné údaje a odešle je stisknutím tlačítka Uložit. 5. Systém vloží zadané údaje do registru SITS a zároveň na server na Katedře informatiky ve formátu trojic s identifikáterm z ontologie OWL. 6. Systém zobrazí úvodní stránku se seznamem pacientů, kde je vidět i nově přidaný jedinec.
Výstupní podmínky	Pacient je přidán

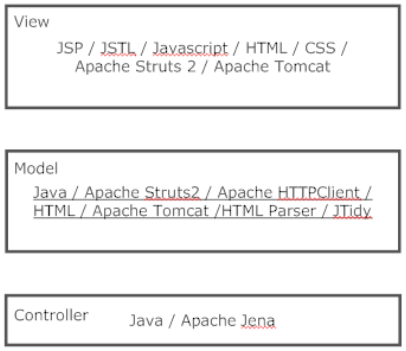
Název případu užití	**Úprava formuláře/pacienta**
Identifikace případu užití	UC03
Aktéři	Uživatel, SITS, KIV/ZČU
Kroky případu užití	1. Uživatel otevře úvodní stránku 2. Uživatel vybere jedním kliknutím z tabulky pacienta spolu s požadovaným formulářem. 3. Systém zobrazí formulář vygenerovaný z ontologie OWL na serveru Katedře informatiky. 4. Systém předvyplní formulář již zadanými údaji do registru SITS 5. Uživatel doplní případně opraví údaje a odešle je stitknutím tlačítka Uložit. 5. Systém vloží případně aktualizuje zadané údaje do registru SITS a zároveň na server na Katedře informatiky ve formátu trojic s identifikáterm z ontologie OWL.

	6. Systém zobrazí úvodní stránku se seznamem pacientů.
Výstupní podmínky	Formulář/pacient je upraven

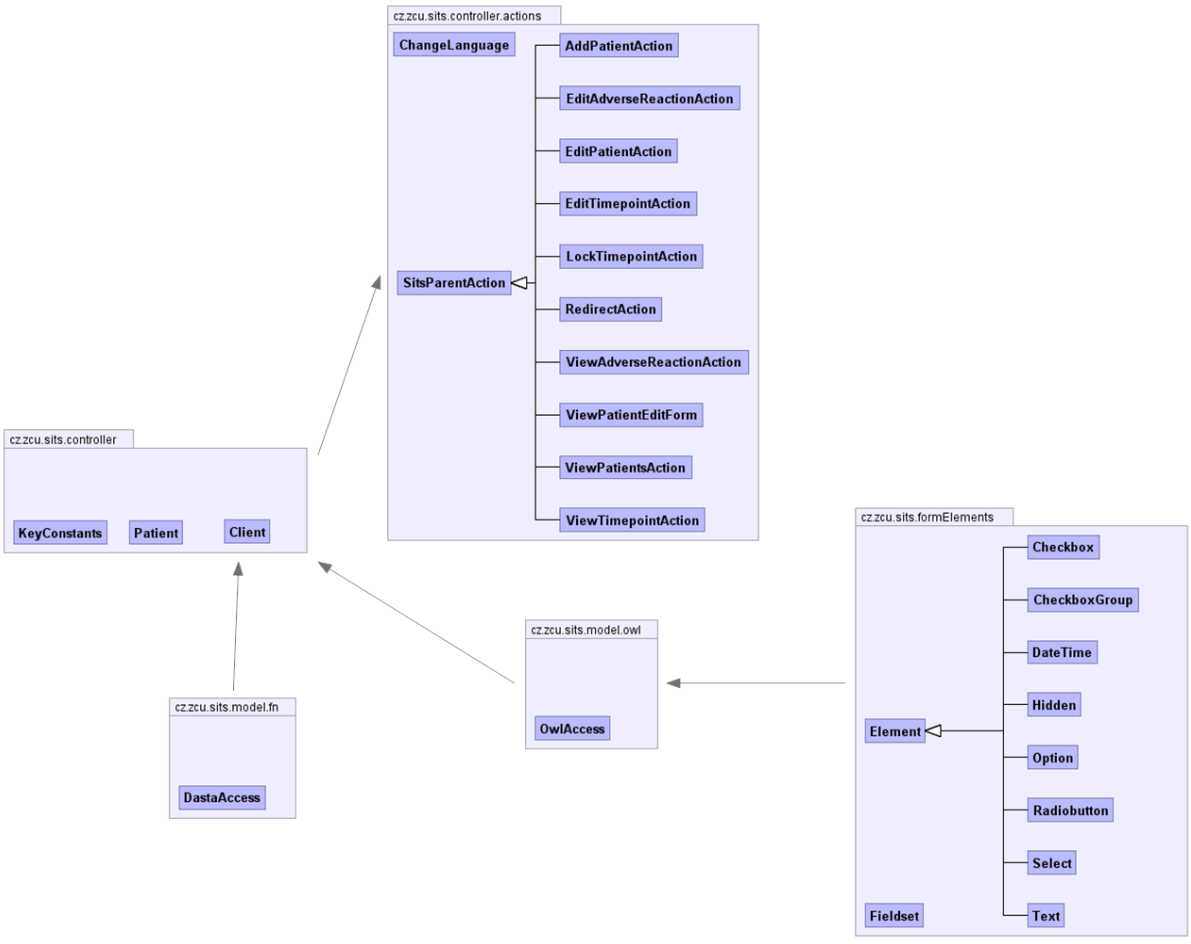
Název případu užití	**Přidání/mazání nežádoucích reakcí**
Identifikace případu užití	UC04
Aktéři	Uživatel, SITS, KIV/ZČU
Kroky případu užití	1. Uživatel zobrazí seznam pacientů 2. Uživatel v tabulce v konkrétním řádku pacienta ve sloupci Nežádoucí reakce klikne na tlačítko Přidej. 3. Systém zobrazí formulář vygenerovaný z ontologie OWL na serveru Katedře informatiky. 4. Systém se připojí do registru SITS a pokud již jsou nějaké reakce zadány, zobrazí je v tabulce nad formulářem s tlačítkem pro jejich případné odstranění 4a1. Uživatel smaže vybraný záznam nežádoucí reakce kliknutím na tlačítko Smazat 4a2. Systém se připojí do registru SITS a vymaže vybraný záznam 4a3. Uživateli je zobrazen zpět formulář, kde může pokračovat v zadání nové nežádoucí reakce, případně stránku opustit. 5. Uživatel vyplní potřebné údaje a odešle je stisknutím tlačítka Uložit. 5. Systém vloží zadané údaje do registru SITS a zároveň na server na Katedře informatiky ve formátu trojic s identifikátorem z ontologie OWL. 6. Systém zobrazí úvodní stránku se seznamem pacientů, kde je vidět i nově přidaný jedinec.
Výstupní podmínky	Nežádoucí reakce je přidána / smazána

4.6 Diagram tříd

Nejprve nastíním použité technologie na základě vzoru Model View Controller (MVC) pro lepší představu diagramu tříd, který bude následovat.



Obrázek 4.2 Použití technologií



Obrázek 4.3 Diagram tříd

4.7 Návrh ontologie

4.7.1 Výběr technologie

Vzhledem k již částečně vytvořené ontologii registru SITS ve formátu OWL uložené jako RDF/XML, byla technologie návrhu sémantiky webu prakticky rozhodnuta. Pro její modifikaci byl zvolen editor Protégé, který mi byl pro danou problematiku doporučen a je také používán na Katedře informatiky.

4.7.2 Komplikace při návrhu

Došlo k částečné úpravě již vytvořené ontologie pro systém hodnocení NIHSS (NIH Stroke Scale nebo také Nationals Institute of Health Stroke Scale) a dále k vytvoření kompletně nové ontologie pro popis jednotlivých formulářů SITS. Ontologie pro tento registr již byla navrhnuta a teoreticky byla vytvořená správně, data do ní lze bez problémů ukládat, avšak vzhledem k několika odchylkám ve „štábní kultuře“ webového portálu SITS, byla pro popis samotných formulářů a komunikaci se serverem špatně použitelná (struktura Ontologie SITS navržená Katedrou informatiky je přidaná do Přílohy 2).
Jednalo se o problémy řádově u jednotek prvků, ale stačila jedna odchylka od standardu a navržená struktura nebyla použitelná.

Aspekty znemožňující použití původně navržené ontologie:

- Stejné bloky obsahující totožné elementy mají různá ID pro odesílání na server SITS, ten potom daný parametr nepřijme, protože očekává jiný identifikátor.
- Stejné elementy mají při odesílání jinou hodnotu, než při jejich následném přijímání ze serveru.

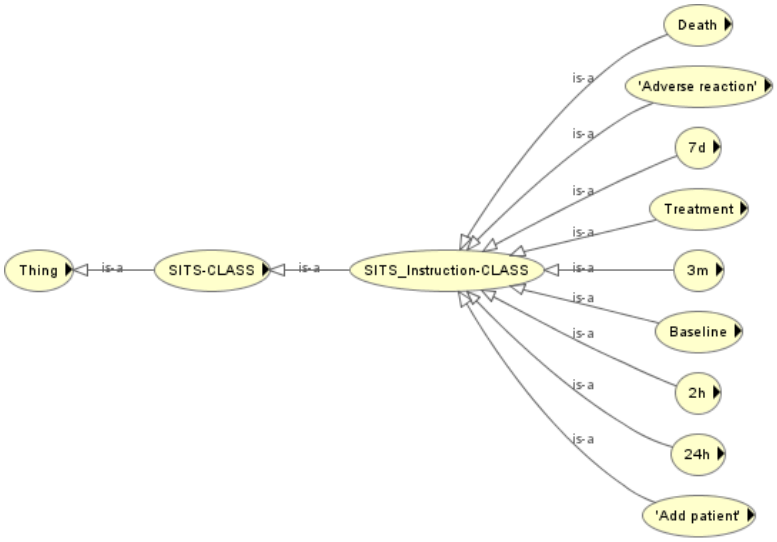
4.7.3 Struktura

Úroveň 1: Registr SITS

Navržená ontologie je reprezentován hlavní třídou SITS_Instruction–CLASS, bez parametrů.

Úroveň 2: Formulář

Následující podtřídy reprezentující konkrétní formuláře. Jejich jméno je zároveň identifikátorem pro odesílání dat na server SITS. Jedná se např. o třídy *Baseline, Treatment, 24h.*

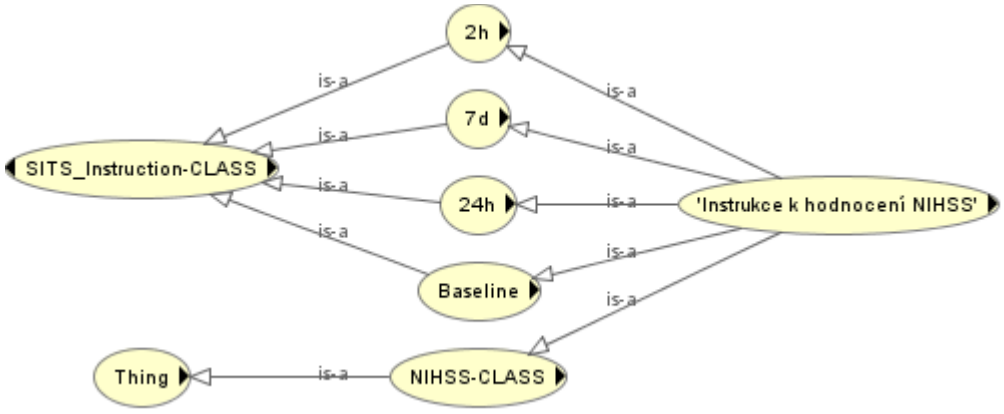


Obrázek 4.4 Třídy formulářů

Úroveň 3: Blok

Dalšími podtřídami jsou jednotlivé bloky, z kterých je formulář složen. Jedná se o více či méně opakované skupiny jako je např. *Krevní tlak, Hodnocení NIHSS, vyšetření CT či MR*. Tyto třídy jsou unikátní a mohou mít několik nadtříd (mohou se vyskytovat v několika formulářích), avšak kvůli odchylkám od standardu v registru existují i výjimky se stejnými jmény viz popsané problémy výše.

Na následujícím obrázku se nachází pohled na blok NIHSS, který se nachází v několika formulářích. Kompletní propojení bloků viz Příloha 1.



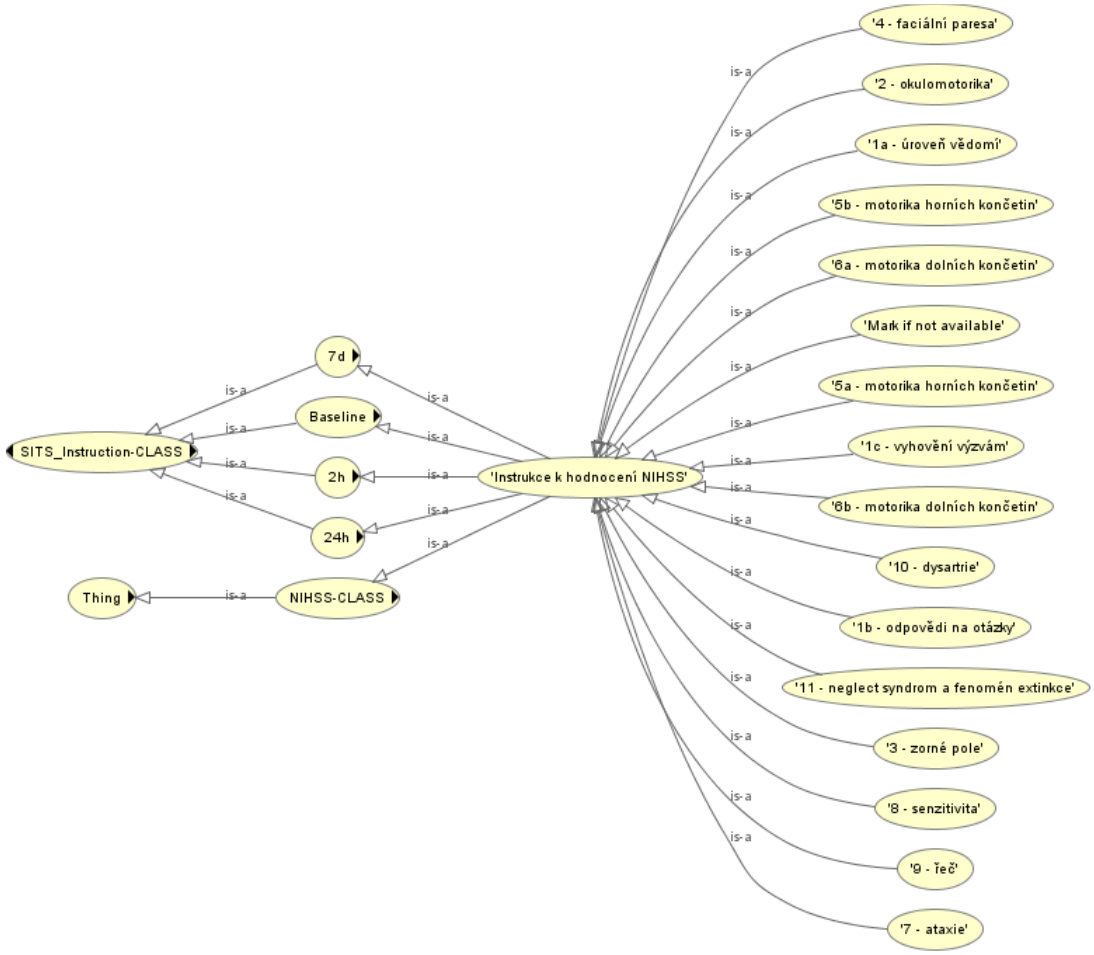
Obrázek 4.5 Třída NIHSS vyskytující se ve formulářích Baseline, 24h, 7d, 2h

Tyto třídy mají již definované pořadí, v jakém se skládají do formulářů pomocí anotace *order*. Dále obsahují nadpis bloku, který je možné uvádět včetně překladů pomocí parametru *label*.

Úroveň 4: Element

Tato úroveň je stále ještě reprezentovaná třídami a reprezenetuje konkrétní položky formuláře jako je např. *datum narození, datum mozkové příhody, podané léky, výška, hmotnost*. Třída již také obsahuje mnoho anotací potřebných pro generování HTML elementu.

- **label** – jmenovka prvku, možné uvádět vícejazyčně
- **identifier** – identifikátor pro datový přenos na server SITS
- **order** – pořadí prvku v rámci bloku
- **description** – popis elementu zobrazovaný u elementu jako nápověda, možný uvádět vícejazyčně
- **comment** – slouží k uložení hodnoty (value) pro prvky typu checkbox a hidden (viz níže)
- **isDefinedBy** – typ elementu, může nabývat jedné z násl. hodnot:
  - **select** – rozbalovací menu
  - **text** – textové pole
  - **checkbox** – zaškrtávací políčko
  - **checkboxgroup** – pole zaškrtávacích políček
  - **datetime** – pole rozbalovacích menu pro výběr data a času
  - **birthdate** – pole pro datum narození
  - **hidden** – skrytá pole formulářů



Obrázek 4.6 Elementy bloku NIHSS

Úroveň 5: Prvky elementu

Tato úroveň je využívaná pouze u elementů typu *select* a *checkboxgroup* a vyjadřuje jednotlivé prvky rozbalovacího menu, popřípadě jednotlivá zaškrtávací políčka.

Tyto objekty nejsou reprezentovány třídami, ale takzvanými individui (individuals). Každý je unikátní v rámci celé ontologie a opakovaně se používají, např. nejpoužívanější hodnota *no value* (bez hodnoty).

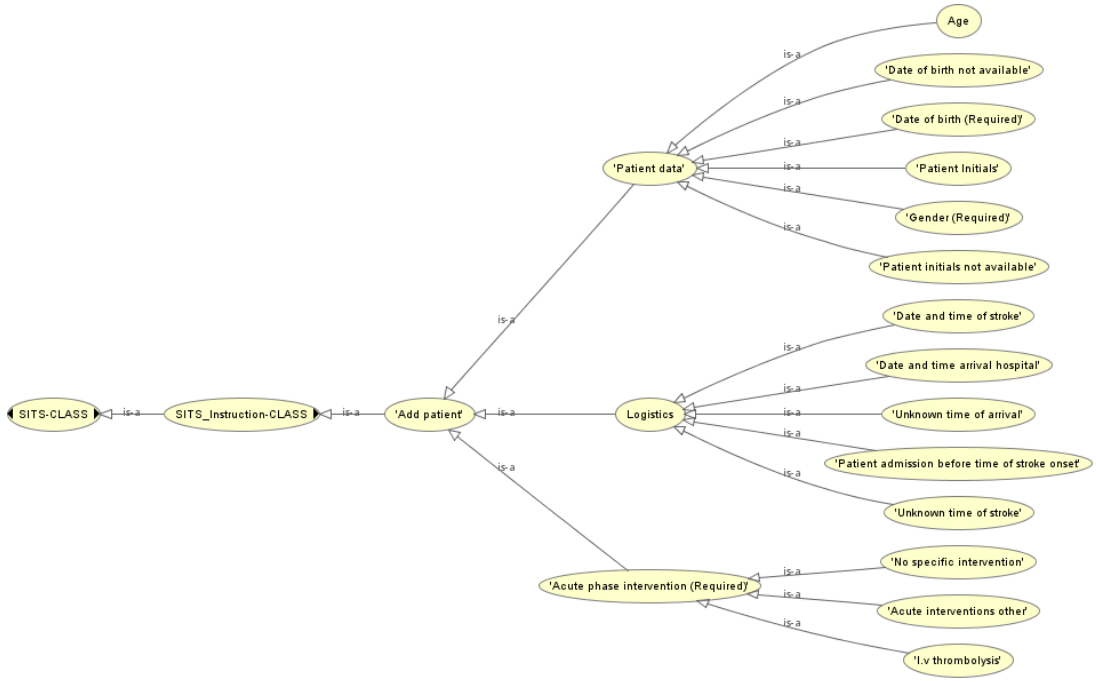
U této úrovně se definují parametry:

- **label** – jmenovka prvku, možné uvádět vícejazyčně
- **identifier** – identifikátor pro datový přenos na server SITS
- **order** – priorita pořadí prvku v rámci elementu (např. zmiňovaná hodnota *no value* bývá vždy první položkou, takže má absolutní prioritu 0.

Dále je zde zaveden speciální prvek, který reprezentuje výčet hodnot rozbalovací menu. Používá se, pokud se element skládá z několika po sobě jdoucích číslic. Typicky výška nebo hmotnost. Formát definice:

- **label** - popisek reprezentovaný klíčovým slovem *range*
- **identifier** – složený ze tří hodnot oddělených pomlčkou
    - prefix před identifikátorem každé hodnoty (SITS občas vyžaduje prefix „A“ před samotným číslem pro komunikaci se serverem
    - počáteční hodnota
    - koncová hodnota,

Zápis pro vygenerování výšky vypadá např. takto: *A-30-23.*



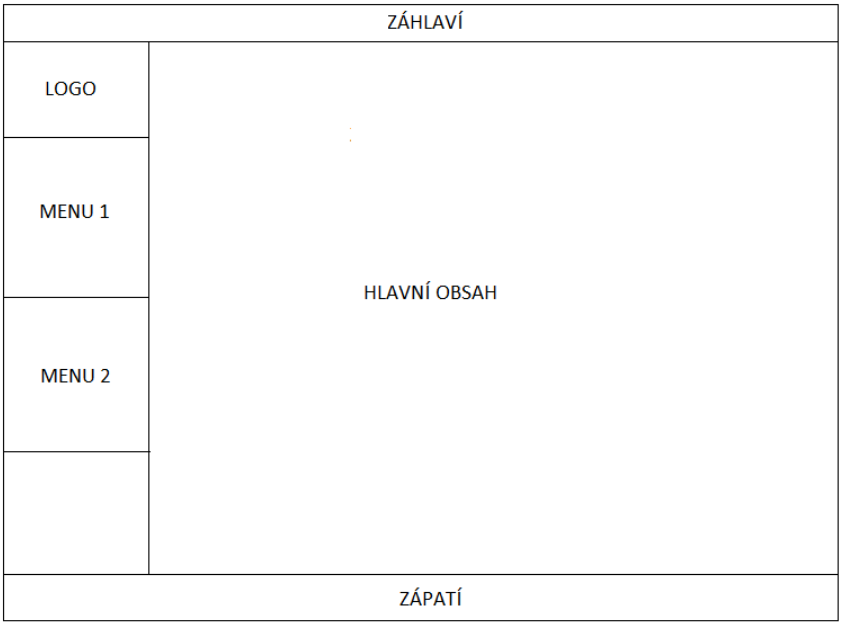
Obrázek 4.7 Ukázka kompletního formuláře přidání pacienta navrženého v Protégé

4.8 Struktura vzhledu

Vzhledem požadavkům, kterým byla jednoduchost aplikace a rychlý přístup k nejpoužívanějším funkcím, bylo rozhodnuto o hlavní stránce jako tabulce s pacienty, ze které bude možno plně editovat veškeré formuláře, které konkrétnímu

jedinci náleží. Tento pohled nebyl v registru SITS možný a zbytečně tím komplikoval práci uživatelům.

Dalším aspektem bylo zachování podobného vzhledu a funkčnosti, na jakou jsou uživatelé registru SITS zvyklí, aby nebylo nutné zaškolování a aplikace byla snadno uvedena do praxe.



Obrázek 4.8 Návrh vzhledu aplikace

5 Implementace

V této kapitole podrobně rozeberu konkrétní implementace technologií, popis tříd a celkovou funkčnost aplikace.

5.1 Verze použitých technologií

- DASTA – DS 03.10.01
- OWL – OWL-DL
- Protégé – v. 4.1.0
- Apache Jena – v. 2.7.0
- Apache Struts 2 – v. 2.2.3
- HTML Parser – v. 1.6
- JTidy – v. r918
- Apache HttpClient – v. 4.1.3
- Apache Tomcat – v. 7.0.22.0
- SPARQL – v. 1.0
- Oracle JDBC Thin Driver – v 6.0
- Java - JDK v. 7 Update 3
- JSP – v 2.1
- JSTL – v. 1.2
- HTML – v.5
- Javascript – v 1.5
- CSS – CSS3
- NetBeans IDE – v. 7.1.1
- Libmed – v. 0.4.0

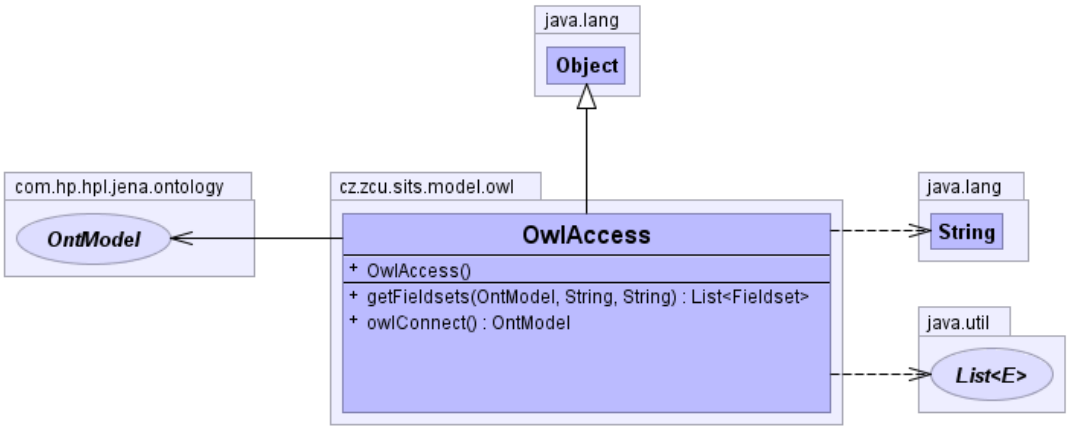
5.2 Popis tříd

V této kapitole popíšu podrobně významné třídy a jejich propojení v mé aplikaci.

5.2.1 Datové třídy

OwlAccess.java

V této třídě pracuji s knihovnou Apache Jena, která se připojuje k OWL ontologii na internetové adrese definované v KeyConstants. Dle zadaného názvu formuláře a jazykové mutace prochází ontologii, na základě které tvoří strukturu formuláře jednotlivými Elementy. Tu poté posílá zpět prostřednictvím Client.java akčním třídám. Připojení a načtení OWL modelu se děje pouze při prvním použití a dále se model po dobu plastnosti session na serveru SITS udržuje v paměti z důvodu úspory času při opětovném generování formulářů.



Obrázek 5.1 Diagram třídy OwlAccess

DastaAccess.java

Tato třída rovněž využívá knihovnu Apache Jena, avšak tentokrát pro podporu dotazovacího jazyka SPARQL a operace nad daty ve formátu RDF. Spolu s připojením k databázi Oracle provádí dotazy nad nemocničními záznamy pro případné předvyplnění již známých údajů do formulářů. Je opět obsluhována třídou Client a vrací seznam pacientů s dostupnými daty.

Dále je zde také uplatněn opačný přístup – do stejné databáze se ukládají všechna data, která jsou odeslána do registru SITS. Údaje budou sloužit pro další analýzy pracovníků Katedry informatiky ZČU.

Element.java

Třída reprezentující jednu položku formuláře. Je to rodič všech podtříd reprezentujících jednotlivé položky ve formuláři. Tedy textová pole, rolovací lišty, zaškrtávací tlačítka nebo například skrytá pole. Výčet všech potomků je patrný z diagramu tříd viz obr. 4.3.
Je prakticky komunikačním protokolem mezi datovou a aplikační logikou.

Každý potomek by měl mít minimálně tyto parametry:

- **label** - textový popis prvku
- **name** – identifikace prvku pro přenos po síti
- **value** – hodnota prvku
- **desc** – popis s nápovědou
- **order** – pořadí prvku v rámci aktuální úrovně zanoření

Fieldset.java

Tato třída sdružuje elementy jedné části formuláře. Z těchto bloků se postupně skládá celek a ve formulářích se různě obměňují. Jedná se o prvek v úrovni přímo nad elementem – ze seznamu instancí této třídy lze už vytvořit kompletní formulář.

5.2.2 Aplikační třídy

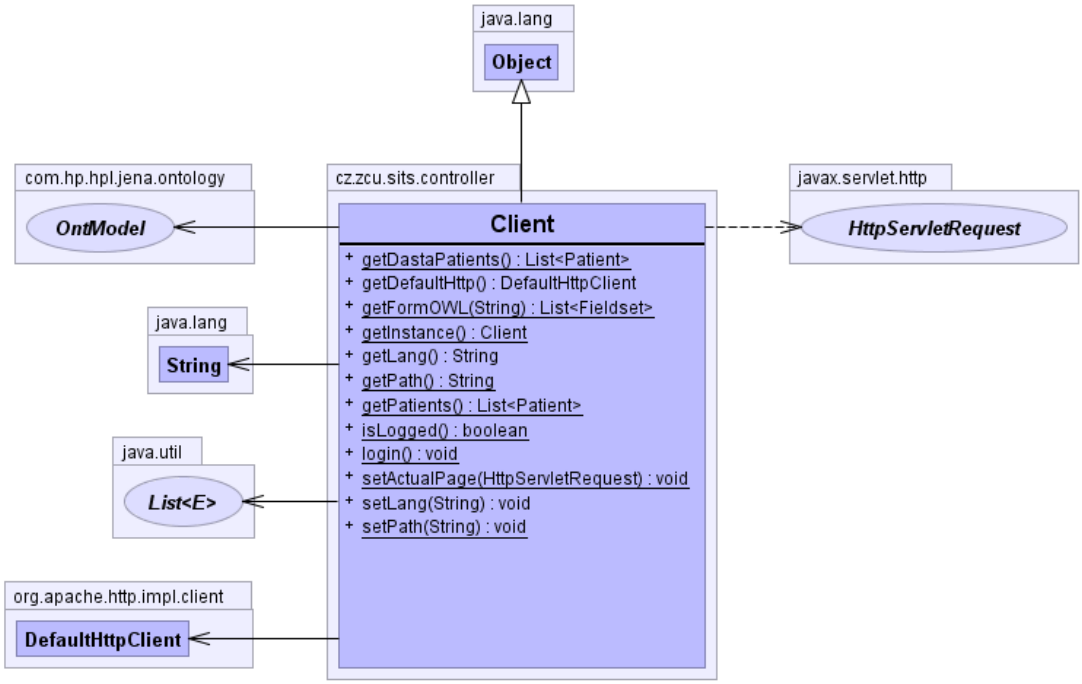
Client.java

Vytvořená dle návrhového vzoru Singleton. Udržují se zde klíčová data společná pro všechny třídy.

Je zde implementováno:

- přihlášení uživatele do webových stránek registru SITS

- udržování session, aby nebylo nutné se na pozadí neustále přihlašovat
- udržování informace o aktuálně vybrané jazykové sadě
- načtení do paměti modelu ontologie OWL a její udržování pro rychlý přístup
- načtení do paměti modelu DASTA/RDF pro stejný důvod jako výše
- udržování aktuální cesty, důležité pro datové třídy, které neznají absolutní cestu jejich umístění (často je totiž odkazovaná jako místo spuštění aplikačního serveru)



Obrázek 5.2 Diagram třídy Client.java

Patient.java

Tato třída slouží pro předávání dat mezi databází pacientů FN Plzeň a aplikační logikou. Reprezentuje jednoho pacienta spolu s jeho údaji, které se dají předvyplnit do formulářů.

SitsParentAction.java a všeobecně akční třídy

Akční třída zastřešující všechny její potomky pracující s formuláři, nalézají se zde pouze společné proměnné a jednoduché metody.

Obecně se dají akční třídy rozdělit na dvě skupiny, které jsou založené na cyklu vygenerování – odeslání formuláře.
První skupina je zaměřena na generování formulářů, zde patří třídy:

- ViewTimepointAction
- ViewPatientsAction
- ViewAdverseReactionAction
- ViewPatientEditForm

Tyto třídy pracují na stejném principu, jenom jsou vždy více či méně upraveny pro specifický problém.
Podle typu požadované stránky si pomocí Client.java nechají vygenerovat objektovou reprezentaci formuláře (z ontologie OWL na serveru Katedry informatiky). Dále se akční třída připojí na server registru SITS a stáhne analogickou stránku formuláře do paměti, kde se pomocí knihovny HTML Parser analyzují již vyplněná pole, zaškrtnutá políčka atd. a spárují se s vygenerovaným formulářem z OWL. Tímto principem se předvyplní daty celá stránka (která ale stále ještě existuje v podobě objektů). Následuje konverze struktury do podoby HTML kódu, pomocí JTidy dochází k posledním finálním úpravám, jako je nastavení odsazení tabulátorů a odřádkování. K vložení na JSP stránky se použije framework Struts 2.

Druhá skupina zpracovává a odesílá formulář zpět na server registru SITS, patří sem tyto třídy:

- EditTimepointAction
- EditPatientAction
- EditAdverseReactionAction

Tyto třídy zpracují zaslaný formulář, vyloučí z něj parametry určené pro vnitřní komunikaci s view vrstvou (např. id pacienta). A odesílají je na server registru SITS pomocí Apache HttpClient. Použití této knihovny bylo nutné vzhledem k tomu, že všechna data je nutné zasílat ve formátu mulitpart, který standardní Java http klient neumí.

AddPatientAction.java

Třída přidání pecienta je specifická, jelikož se zde nachází jak samotné generování formuláře, tak i odesílání na server SITS a předvyplňování formulářů funguje díky odlišnému principu – komunikaci s databází pacientů z FN Plzeň prostřednictvím třídy Client a na datové úrovni přeneseně s DastaAccess.java.

RedirectAction.java

Třída sloužící k jednoduchému přesměrování obsahu z vybraných SITS stránek do mnou vytvořené webové aplikace jako jsou příručky pro vyšetření pacientů, uživatelské příručky k registru SITS atd.

ChangeLanguage.java

V této třídě je implementováno přepnutí nastavení Locale v servletech Struts 2 na požadovanou jazykovou verzi a je navrácena původní stránka. Její adresa je uchovávána v session, takže vždy dojde k návratu na původní místo.
Implementace podpory více jazyků je zajištěna pomocí properties souborů s přeloženými dvojcemi identifikátor = jazykový překlad s podporou frameworku Struts 2.

5.2.1 Zobrazovací vrstva

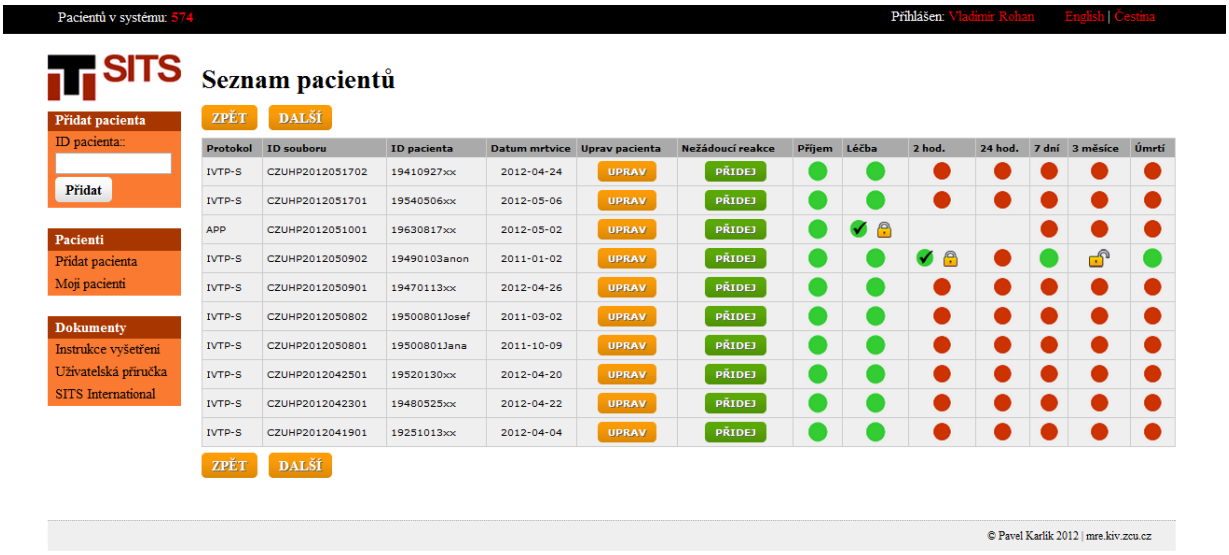
View vrstva je tvořena JSP stránkami obohacenými o Javascript kód. Ten je zde z důvodu nutnosti odesílání i parametrů bez hodnoty (specifikace převzatá z registru SITS, jinak je toto nestandardní procedura). Jsou tedy naprogramovány funkce, které formulář ještě před odesláním analyzují a přidají speciální atribut, který obsahuje výčet všech parametrů ve formuláři. V akční třídě se potom díky tomuto seznamu a jeho srovnání s přijatými hodnotami lehce objeví prázdné parametry, které slouží serveru SITS zřejmě pouze pro kontrolu validity formuláře. Bez těchto prvků registr požadavek odmítne.
Dále se zde také nachází technologie JSTL, která je použitá pro navázání CSS stylů na stránku viz příklad:

```
<link rel="stylesheet" href="<c:url value='/styles.css'/>" type="text/css" />
```

Jednotlivé stránky se do sebe skládají pomocí *jsp:include* tagů, což je dnes v JSP standardem. Byla možnost využít novou technologii Tiles v Apache Struts2, která toto řešení ještě vylepšuje, ale pro komplikovanost a nedostatek času byla zavrhnuta.

5.3 Design a navigace

Po zadání adresy webové aplikace je uživatel po krátkém zobrazení loga registru SITS (při kterém na pozadí probíhají veškeré přihlašovací procedury a napojení na ontoligii) přesměrován na úvodní stránku. Jazyk je pro první přístup nastaven na češtinu.



Obrázek 5.3 Úvodní stránka aplikace

5.3.1 Menu

Navigace v aplikaci je jednoduchá pomocí menu na levé straně obrazovky, které je rozděleno na tři části:

- první část obsahuje textové pole a odesílací tlačítko pro přidání uživatele z databáze FN Plzeň
- v druhé se nachází odkazy na práci s pacienty
- v třetí jsou odkazy na uživatelské příručky a také odkaz na stránky registru SITS, pro případ problému a jeho rychlého vyřešení přímo na stránkách jeho stránkách.

Menu dominuje ikona registru SITS, která slouží na návrat na úvodní stránku.

5.3.2 Seznam pacientů

Na této stránce se v hlavní obsahové části nachází seznam přidaných pacientů spolu se stavem vyplnění jejich formulářů. Velikost seznamu je zmenšena pro přehlednost obr. 5.3. V aplikaci je na jedné stránce možné zobrazit libovolný počet pacientů.

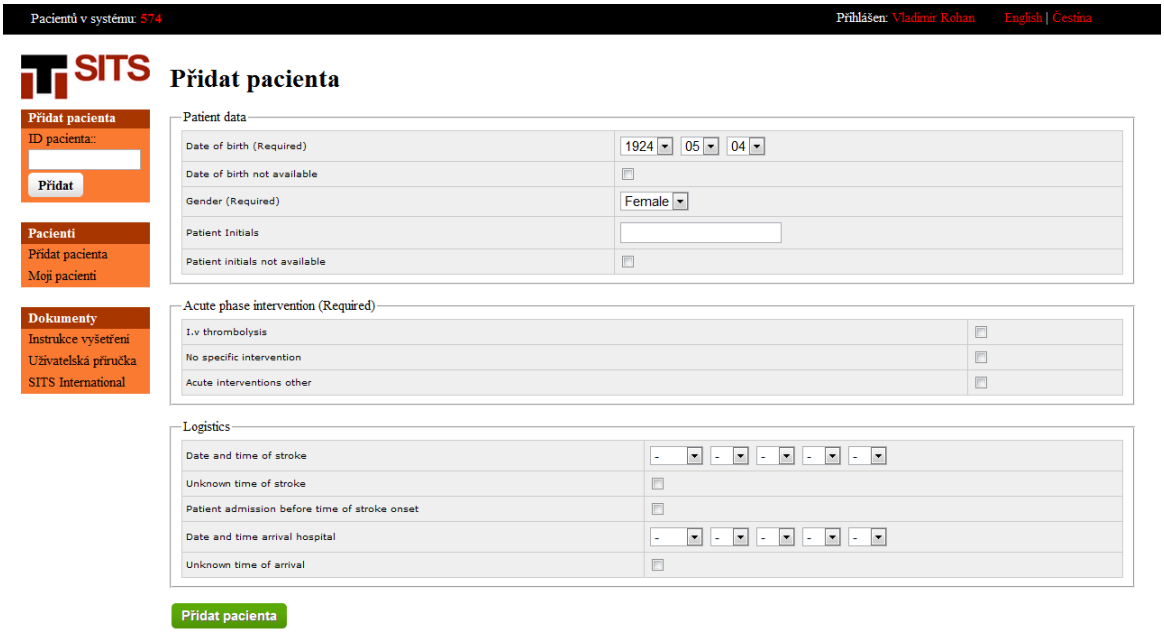
Jednotlivé ikony znamenají:

- **červená** – nevyplněno
- **zelená** - částečně vyplněno
- **zelená se zámkem** – kompletně vyplněno, připraveno na uzamknutí (potvrdí se kliknutím na zámeček)
- **odemknutý zámek** – označuje již uložený a potvrzený formulář. Pro odemknutí stačí kliknout na toto tlačítko a záznam je možné opětovně upravovat.

Dále se zde nachází přístupová tlačítka pro editaci pacienta a přidávání nežádoucích reakcí spolu s navigačními tlačítky procházení seznamu.

5.3.3 Přidat pacienta

Přidání pacienta do registru je možné dvěma způsoby. Pokud zná uživatel jeho ID ze systému FN Plzeň, je možné přes textové pole přejít již přímo k předvyplněnému formuláři z nemocniční databáze. Pokud je zadáno ID špatně nebo uživatel kliknul na odkaz *Přidat pacienta* z druhého menu, je přesměrován na stejný formulář, který je ale nutné celý manuálně vyplnit.



Obrázek 5.4 Přidání pacienta

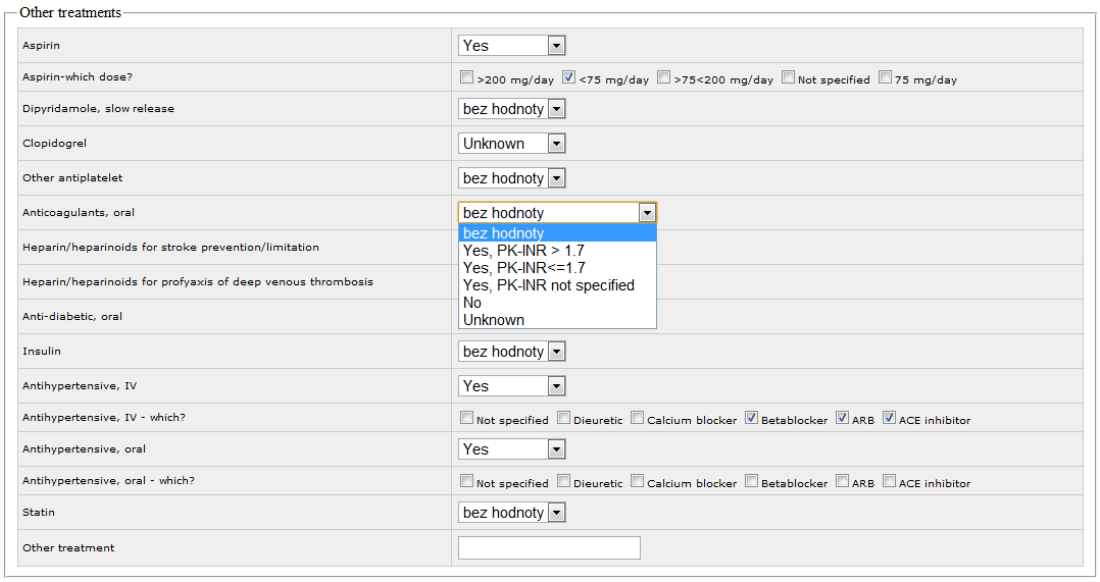
5.3.4 Editace formuláře

Formulář se skládá z jednotlivých bloků (viz obr. 5.5), které jsou v HTML reprezentovány tagy *fieldset*. K jeho editaci přecházíme vždy ze seznamu pacientů kliknutím na ikonku v příslušném sloupci, který určuje, jaký typ se má zobrazit. Pokud již byly dříve nějaké údaje zadány, jsou již ve formuláři předvyplněné.

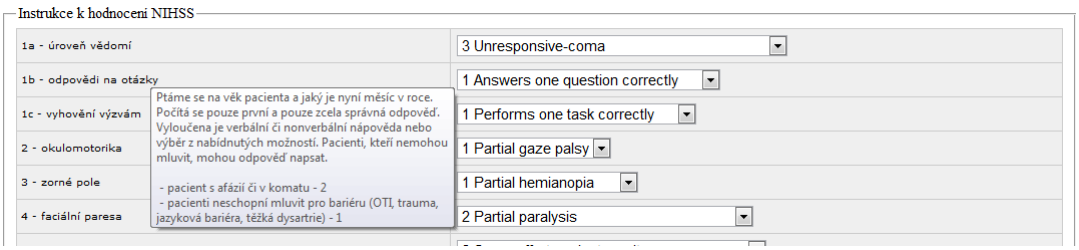
Pro představu nejsložitější formulář zabírá po vygenerování cca 3000 řádek HTML kódu.

Ve formulářích jsou použité hodnoty podle zvolené jazykové sady, pokud pro daný prvek neexistují, jsou použita výchozí data (angličtina). Toto je vidět například na obr. 5.6).

Dále je zde implementovaná nápověda, která se zobrazí po „najetí“ myší na popis príslušné položky viz obr. 5.6.



Obrázek 5.5 Část formuláře - výběr léčby



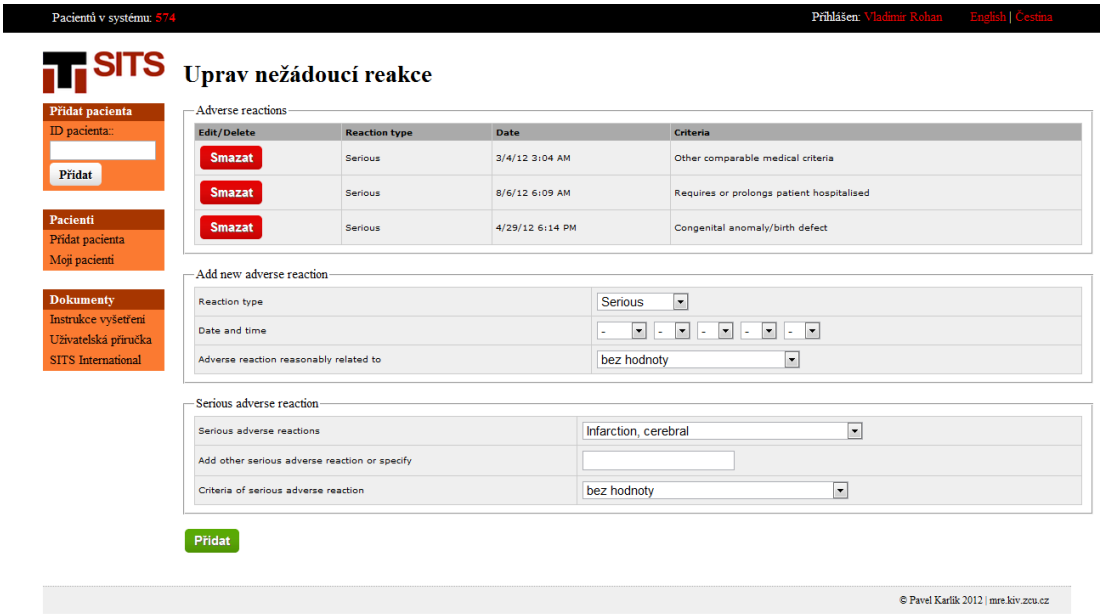
Obrázek 5.6 Zobrazení nápovědy

5.3.5 Nežádoucí reakce

Formulář nežádoucích reakcí se liší od ostatních v několika prvcích. Prvním rozdílem je použití Javascriptu na aktivní zobrazování/skrývání jednotliivých bloků. Tato funkčnost vznikla díky podobné konstrukci na serveru SITS. Je nutné zadávat vážné a lehké reakce v jednom formuláři. Skrytě se totiž odesílají data ze všech bloků, a pokud by se neodeslaly najednou, ale např. jenom blok vážné reakce, registr by požadavek odmítl.

Druhým rozdílem je možnost prohlížení a mazání nežádoucích reakcí.

Jako jediný formulář slouží totiž pouze k přidávání dat a ne k editaci jako ostatní, je tedy možné zadat N nežádoucích reakcí ke každému pacientovi, které se postupně zobrazují v tabulce nad formulářem. V té se také u každé řádky nachází políčko Smazat, kterým danou reakci můžeme odstranit.



Obrázek 5.7 Nežádoucí reakce

5.4 Nasazení systému

Aplikace je zkompilována pro web server Apache Tomcat (v. 7.0.22.0) a distribuována pomocí tzv. WAR souboru a knihoven Xerces. Tento soubor je archivem ve formátu ZIP s kompletní strukturou programu, se kterým lze pracovat pomocí klasických archivačních programů.

Před instalací je nutné vzhledem ke konfliktu XML parserů v Apache Struts2 a knihovny Apache Jena nakopírovat Xerces parser do složky *endorsed* v hlavní složce instalace Apache. Pokud není vytvořena je nutné ji nově vytvořit. Potřebné soubory knihovny jsou dodané v souboru *xerces.zip.*

Instalace na web serveru probíhá nakopírováním tohoto souboru do podsložky */webapps* v adresáři s instalací Apache Tomcat.
Aplikace se spustí ve webovém prohlížeči na *[adresa webového serveru]/Sits*

5.5 Možná rozšíření

5.5.1 Generování ontologií ze stránek registru SITS

Vzhledem k aktuálnímu stavu, kdy se ontologie tvoří manuálně, by byla vhodná aplikace, která by tvořila ontologii programově na základě formulářů přímo z registru SITS. Aktuální řešení je totiž problémové vzhledem k jakýmkoliv změnám ve formulářích SITS. Poté je potřeba ontologii manuálně aktualizovat, což sice není v nástroji Protégé náročné, ale vyžaduje okamžitý zásah. Do aktualizace pak není příslušný formulář plně použitelný.

5.5.2 Předvyplňování formulářů z databáze na KIV/ZČU

Díky ukládání veškerých odesílaných dat do SITS i na Katedře informatiky je možné postupně opustit parsování dat HTML formulářů z webového registru a jejich následnému vkládání do formulářů vygenerovaných z ontologie. Tato data by v budoucnu mohla být brána přímo z databáze na KIV/ZČU, ke které již teď program umí přistupovat pomocí dotazovacího jazyka SPARQL.
Problémem jsou akorát chybějící údaje pro pacienty zadané přes SITS před počátkem používání této aplikace.

6 Závěr

V úvodní části textu byla nejprve osvětlena problematika datových standardů a jejich využití v moderních zdravotnických informačních systémech, na kterou navazoval podrobný popis Datového standardu Ministerstva zdravotnictví DASTA, který jsem v části práce používal. Hlavním cílem byl návrh a implementace aplikace pro lékaře ve FN Plzeň usnadňující vyplňování složitých formulářů o pacientech s mozkovou příhodou do mezinárodního registru mozkových mrtvic SITS.

Aplikace je vytvořena v programovacím jazyce Java v propojení s frameworkem Apache Struts 2, konkrétní formuláře a struktura registru pomocí ontologického modelu sloužícího k popisu sémantického webu. Aplikace splnila veškeré nároky, které na ní byly kladeny, bohužel však prozatím není dostatek dat z FN Plzeň, které by bylo možné do daných formulářů importovat, což je ale otázkou času.

Za největší přínos autor považuje seznámení se s technologií popisu sémantického webu a prohloubení zkušeností s komunikací prostřednictvím HTTP protokolu. Dále je cennou zkušeností práce s novými technologiemi jako je Apache Struts 2, popřípadě problematika extrakce dat z HTML kódu.

Možné rozšíření aplikace spočívá zejména v programovém generování ontologií přímo z registru SITS, případně zobrazování již zadaných údajů do registru přímo z databáze na Katedře informatiky a tím opuštění extrakce dat z HTML kódu stránek registru.

Použitá literatura

[1] Datový standard pro předávání dat o pacientech mezi informačními systémy zdravotnických zařízení verze 01.01 [online]. Kladno: ČSZIVI, 1995 – [cit. 6. března 2012]. Dostupné na www: http://cszivi.cls.cz/doc/standroz.htm.

[2] Komunikace mezi zdravotnickými IS v praxi [online]. Praha: CompuGroup, 2002 – [cit. 10. března 2012]. Dostupné na www: http://dasta.lf2.cuni.cz/dsmz/hypertext/DSBBX.htm.

[3] Datový standard MZ ČR [online]. Praha: MZ ČR, 2012 - [cit. 1. dubna 2012]. Dostupné na www: http://ciselniky.dasta.mzcr.cz/.

[4] Znalostní technologie I. [online]. Praha: Martina Husáková, 2008 – [cit. 5. dubna 2012]. Dostupné na www: http://lide.uhk.cz/fim/ucitel/fshusam2/lekarnicky/zt1/zt1_kap04.html.

[5] OWL - Web Ontology Language [online]. Praha: Ing. Marek Obitko , 2009 – [cit. 12. dubna 2012]. Dostupné na www: http://labe.felk.cvut.cz/~obitko/spr/owl.html.

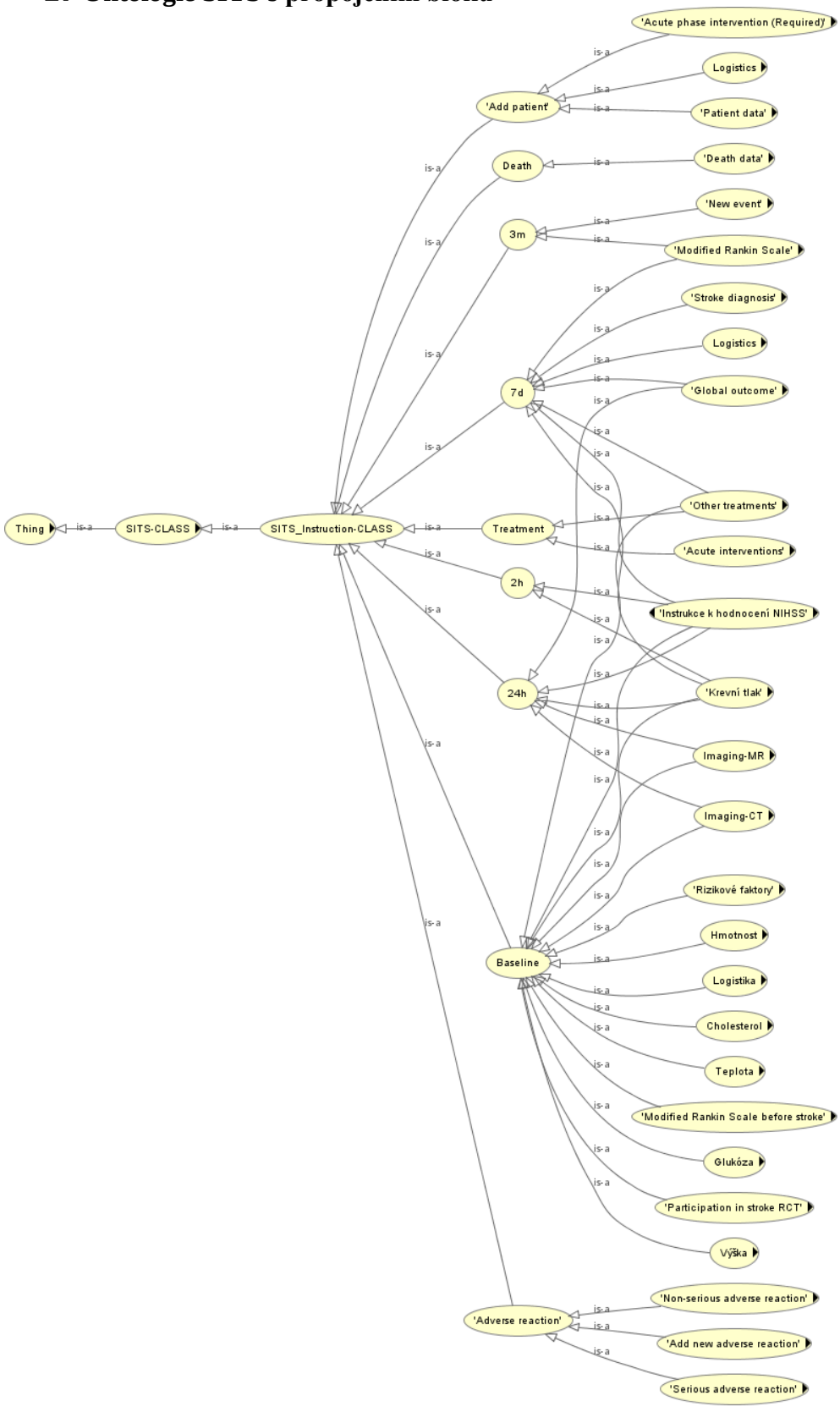
[6] Protégé [online]. Stanford, USA: Standford University School of Medicine, 2012 – [cit. 12. dubna 2012]. Dostupné na www: http://labe.felk.cvut.cz/~obitko/spr/owl.html.

[7] Apache Jena [online]. Los Angeles, USA: Apache Software Foundation, 2012 – [cit. 29. dubna 2012]. Dostupné na www: http://jena.apache.org/.

[8] Struts 2 Tutorial [online]. Delhi, India: Rose India, 2008 – [cit. 28. února 2012]. Dostupné na www: http://www.roseindia.net/struts/struts2/

Přílohy:

1. **Ontologie SITS s propojením bloků**



2. Ontologie SITS navržená na Katedře informatiky

Group	SITS Ontology		Property	Property in SITS Ontology
Treatment	class	sits:Treatment	Aspirin	sits:aspirin
	attribute	sits:hasTreatment	Which dose?	sits:dose
	member	sits:inTimepoint	No treatment (marker)	sits:noTreatment
			Dipyridamole, slow release	sits:dipyridamole
			Clopidogrel	sits:clopidogrel
			Other antiplatelet	sits:antiplateletOther
			Anticoagulants, oral	sits:anticoagulantsOral
			Heparin/heparinoids for stroke prevention/limitation	sits:heparinPrevention
			Heparin/heparinoids for profyaxis of deep venous thrombosis	sits:heparinProfyaxis
			Anti-diabetic, oral	sits:antidiabeticOral
			Insulin	sits:insulin
			Antihypertensive IV	sits:antihypertensiveIV
			Which treatment?	
			Antihypertensive Oral	sits:antihypertensiveOral
			Which treatment?	
			Statin	sits:statin
			Other treatment	sits:treatmentOther
			Dose Actilyse	sits:doseActilyse
			Date and time for Actilyse treatment	sits:datetime
			Serious adverse reaction	sits:seriousAdverseReaction
			Acute interventions	sits:acuteIntervetions
			Which	
Risc Factors	class	sits:RiscFactors	Hypertension: Diagnosis of hypertension	sits:hypertension
	attribute	sits:hasRiscFactors	Diabetes: Diagnosis of diabetes	sits:diabetes
			Hyperlipidemia: Diagnosis of hyperlipidemia	sits:hyperlipidemia
			Current smoker: Current smoker at stroke onset	sits:currentSmoker
			Previous smoker: Previous smoker, but stopped before stroke onset	sits:previousSmoker
			Previous stroke earlier than 3 months: Previous diagnosis of stroke (based on clinical symptoms)	sits:previousStrokeEarlierThan3M
			Previous stroke within 3 months: Previous diagnosis of stroke (based on clinical symptoms) within the latest 3 months	sits:previousStrokeWithin3M
			Previous TIA/ Amaurosis fugax: Previous diagnosis of TIA or Amaurosis fugax	sits:previousTIAOrAmaurosis
			Atrial fibrillation: Diagnosis of atrial fibrillation (permanent or paroxysmal)	sits:atrialFibrilation
			Severity scale	sits:
NIHSS	class	**sits:NIHSS**	1a. Level of Consciousness	*nihss:hasConsciousness*
	subClassOf	**nihss:NIHSS**	1b. LOC Questions	*nihss:hasQuestions*

	property	sits:hasNIHSS	1c. LOC Commands	*nihss:hasCommands*
	member	sits:inTimepoint	2. Best Gaze	*nihss:hasBestGaze*
			3. Visual	*nihss:hasVisual*
			4. Facial Palsy	*nihss:hasFacialPalsy*
			5a. Motor Right Arm	*nihss:hasMotorRightArm*
			5b. Motor Left Arm	*nihss:hasMotorLeftArm*
			6a. Motor Right Leg	*nihss:hasMotorRightLeg*
			6b. Motor Left Leg	*nihss:hasMotorLeftLeg*
			7. Limb Ataxia	*nihss:hasLimbAtaxia*
			8. Sensory	*nihss:hasSensory*
			9. Best Language	*nihss:hasBestLanguage*
			10. Dysarthria	*nihss:hasDysarthria*
			11. Extinction and Inattention (formerly Neglect)	*nihss:hasExtinctionAndInattention*
			NIH Score	*nihss:score*
			Date and time on NIHSS	sits:datetime
Blood Pressure	class	sits:BloodPressure	Systolic blood pressure	sits:systolicBloodPressure
	attribute	sits:hasBloodPressure	Diastolic blood pressure	sits:diastolicBloodPressure
	member	sits:inTimepoint	Blood pressure date and time	sits:datetime
Glucose	class	sits:Glucose	Glucose	sits:glucose
	attribute	sits:hasGlucose	Date and time	sits:datetime
	member	sits:inTimepoint		
Cholesterol	class	sits:Cholesterol	Total cholesterol	sits:cholesterol
	attribute	sits:hasCholesterol	Date and time	sits:datetime
	member	sits:inTimepoint		
Temperature	class	sits:Temperature	Temperature	sits:temperature
	attribute	sits:hasTemperature	Date and time	sits:datetime
	member	sits:inTimepoint		
Weight	class	sits:Weight	Weight estimated	sits:weightEstimated
	attribute	sits:hasWeight	Weight measured	sits:weightMeasured
	member	sits:inTimepoint	Date and time	sits:datetime
Length	class	sits:Length	Length	sits:length
	attribute	sits:hasLength	Date and time	sits:datetime
	member	sits:inTimepoint		
Link to other trial	class	sits:Trial	Is the patient participating in any trial?	
	attribute	sits:inTrial	Which acute stroke trial?	sits:inAcuteTrial
			Which prevention trial?	sits:inPreventionTrial
			Which other trial?	sits:inOtherTrial
CT	class	sits:CT	CT done	
	attribute	sits:hasCT	CT date & time	sits:datetime
	member	sits:inTimepoint	CT current infarct	sits:currentInfarct
			CT ASPECTS Score	sits:aspectsScore
			CT Dense artery sign	sits:denseArterySign
			CTA occlusion (optional)	sits:occlusion
			CT perfusion deficit (optional)	sits:perfusionDeficit
			CT perfusion deficit volume (optional)	sits:perfusionDeficitVolume

			CT infarct volume (optional)	sits:infarctVolume
			CT perfusion/infarct mismatch (optional)	sits:perfussionOrInfarctMismatch
			Local haemorrhage	sits:localHaemorrhage
			Remote haemorrhage	sits:remoteHaemorrhage
			Cerebral oedema	sits:cerebralOedema
MR	class	sits:MR	MR Done	
	attribute	sits:hasMR	MR date & time	sits:datetime
	member	sits:inTimepoint	MR current infarct	sits:currentInfarct
			MR ASPECTS Score	sits:aspectsScore
			MRA occlusion (optional)	sits:occlusion
			MR perfusion deficit (optional)	sits:perfusionDeficit
			MR perfusion deficit volume (optional)	sits:perfusionDeficitVolume
			MR infarct volume (optional)	sits:infarctVolume
			MR perfusion/infarct mismatch (optional)	sits:perfussionOrInfarctMismatch
			Local haemorrhage	sits:localHaemorrhage
			Remote haemorrhage	sits:remoteHaemorrhage
			Cerebral oedema	sits:cerebralOedema
Logistics	class	sits:Logistics	Date and time stroke onset	sits:strokeOnsetDateTime
	attribute	sits:hasLogistics	Time of alarm	sits:alarmDateTime
			Time for pick-up	sits:pickUpDateTime
			Time for arrival at hospital	sits:hospitalArrivalDateTime
			Logistics confirmation	
Stroke Type	class	sits:StrokeType	Type of stroke	sits:strokeType
	attribute	sits:hasStrokeType	Which	
			ICD code	sits:icd
			ICD code	sits:icd
			ICD code	sits:icd
			ICD code	sits:icd
			Involved vascular territorry	sits:involbedVascularTerritorry
Discharge	class	sits:Discharge	Date and time of discharge	sits:datetime
	attribute	sits:hasDischarge		
Death	class	sits:Death	Date and time of death	sits:datetime
	attribute	sits:hasDeath	Cause of death	sits:causeOfDeath
Event	class	sits:Event	New event	sits:newEvent
	attribute	sits:hasEvent	ICD	sits:icd
Rankin score	class	sits:RankinScore	Rankin score	sits:rankinScore
	attribute	sits:hasRankinScore	Rankin score date and time	sits:datetime
	member	sits:inTimepoint		
Global outcome	class	sits:GlobalOutcome	Global outcome	sits:globalOutcome
	attribute	sits:hasGlobalOutcome	Global outcome date and time	sits:datetime
	member	sits:inTimepoint		
mRS score	class	sits:MRSScore	mRS score	sits:mRSScore
	attribute	sits:hasMRSScore	Prior disability (mRs 1-5) has other cause than stroke	sits:otherCauseForPriorDisability

SITS Report

class	sits:SITSReport	Members Instance Count	
member	sits:hasTreatment	multiple (more timepoints)	
member	sits:hasRiscFactors	1	
member	sits:hasNIHSS	multiple (more timepoints)	
member	sits:hasBloodPressure	multiple (more timepoints)	
member	sits:hasLogistics	1	
member	sits:hasGlucose	multiple (more timepoints)	
member	sits:hasCholesterol	multiple (more timepoints)	
member	sits:hasTemperature	multiple (more timepoints)	
member	sits:hasWeight	multiple (more timepoints)	
member	sits:hasLength	multiple (more timepoints)	
member	sits:inTrial	1	
member	sits:hasCT	multiple (more timepoints)	
member	sits:hasMR	multiple (more timepoints)	
member	sits:hasStrokeType	1	
member	sits:hasDischarge	1	
member	sits:hasDeath	1	
member	sits:hasEvent	1	
member	sits:hasRankinScore	multiple (more timepoints)	
member	sits:hasGlobalOutcome	multiple (more timepoints)	
member	sits:hasMRSScore	1	