

Univerzita Karlova v Praze
Matematicko-fyzikálna fakulta

BAKALÁRSKA PRÁCA



Juraj Mišúr

Inteligentní proxy pro mobilní uživatelská rozhraní elektronického zdravotního záznamu

Katedra softwarového inženýrství

Vedúci bakalárskej práce: Ing. Lubomír Bulej

Externý konzultant: Mgr. Miroslav Nagy

Študijný program: informatika, programovanie

2006

Ďakujem vedúcemu práce Ing. Lubomírovi Bulejovi za vedenie, rady a pripomienky ohľadom programovania a bakalárskej práce. Taktiež chcem poďakovať Josefovi Špidlenovi za zadanie projektu a Miroslavovi Nagyovi za pravidelné konzultácie ohľadom problematiky elektronického zdravotného záznamu MUDR.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

V Prahe dňa 8.8.2006

Juraj Mišúr

Obsah

1 Úvod	7
1.1 Zadanie projektu	7
1.2 Motivácia a ciele	8
1.3 Štruktúra textu	9
2 Systémy EHR	11
2.1 Systémy EHR v zahraničí	11
2.2 Systémy EHR v Českej a Slovenskej Republike	12
2.3 Projekt MUDR	13
2.4 Projekt MUDRC	14
2.5 Projekt MUDR Proxy	15
3 Návrh MUDR Proxy	16
3.1 Rozbor požiadaviek projektu	16
3.1.1 Integrácia pacientových dát z MUDR serverov	16
3.1.2 Poskytovanie pacientových dát užívateľom	17
3.1.3 Použitie konkrétnych technológií	18
3.2 Návrh architektúry	18
3.2.1 Rozdelenie funkčnosti	18
3.2.2 Databáza pacientov	18
3.2.3 Zloženie a funkčnosť proxy serveru	20
3.2.4 CGI skript	22
3.2.5 Zloženie a funkčnosť webovej stránky	22
4 Implementácia MUDR Proxy	24
4.1 Proxy server	24
4.1.1 Implementačné prostredie	24
4.1.2 Programovací jazyk	24
4.1.3 Implementácia proxy servera	25

4.1.4	Komunikácia s MUDR servermi	26
4.1.5	Komunikácia s CGI skriptom	26
4.1.6	Komunikácia medzi proxy servermi	27
4.1.7	Vývojové a implementačné postupy	27
4.1.8	Práca s dátami pacienta a znalostnej bázy	30
4.2	Databáza	31
4.3	CGI skript	32
4.4	Webová stránka	32
4.4.1	Univerzálnosť	33
4.4.2	Kostra stránky	33
4.4.3	Generovanie obsahu	33
4.4.4	Užívateľské prostredie	34
5	Prínos a porovnanie projektu	38
5.1	Zhodnotenie vývoja aplikácie	38
5.2	Porovnanie	39
5.3	Prínos	39
6	Budúcnosť projektu	41
6.1	Proxy server	41
6.2	CGI skript	42
6.3	Webová stránka	42
7	Záver	43
A	Použité knižnice	44
B	Schéma prvotného návrhu	46
	Literatúra	47

Názov práce: Inteligentní proxy pro mobilní uživatelská rozhraní elektronického zdravotního záznamu

Autor: Juraj Mišúr

Katedra (ústav): Katedra softwarového inženýrství

Vedúci bakalárskej práce: Ing. Lubomír Bulej

e-mail vedúceho: Lubomir.Bulej@mff.cuni.cz

Abstrakt: V predloženej práci študujem možnosti využitia elektronického zdravotného záznamu, navrhnutého a implementovaného Josefom Špidlenom, na tzv. tenkých užívateľských klientoch, ako je napr. webový prehliadač na PC, vreckovom počítači alebo mobilnom telefóne. Úlohou práce je sprostredkovať užívateľovi (lekárovi, sestričke, záchranej službe) informácie o jednotlivých pacientoch čo najrýchlejšie a v prehľadnej a užívateľsky prívetivej forme. Hlavným prínosom projektu je integrácia pacientových dát z rôznych zdrojov (databáz elektronického zdravotného záznamu MUDR), ktoré majú iba útržky znalostí o pacientovi, tak ako v reálnom živote jednotliví doktori alebo nemocničné oddelenia. Proxy združuje dáta o pacientovi zo všetkých dostupných zdrojov a prezentuje ich ako jeden úplný a celistvý obraz pacienta. Takáto globálna integrácia dát je obrovská pomoc pre akúkoľvek interakciu s pacientom s ohľadom na jeho zdravotnú situáciu.

Kľúčové slová: Elektronický Zdravotný Záznam, EHR, MUDR, MUDRC

Title: Intelligent proxy for mobile user interfaces to electronic health record

Author: Juraj Mišúr

Department: Department of Software Engineering

Supervisor: Ing. Lubomír Bulej

Supervisor's e-mail address: Lubomir.Bulej@mff.cuni.cz

Abstract: In the present work we study possibilities of using the electronic healthcare record, created and implemented by Josef Špidlen, on thin user clients as web browser on PC, PDA or mobile phone. The task of this work is to present patient's medical informations to the users (physicians, nurses, emergency staff) in the fastest, well arranged and user-friendly form. The main benefit of tis project is integration of patient's data from several sources (databases of electronic healthcare record MUDR), which every of them has only the fragment of knowledge about patient, as in real life separate

physicians or hospitals. Proxy keeps the patient's data from all available sources and presents them as complete and compact image of the patient. This kind of global EHR data integration is a major help for every interaction with the patient with respect to his/her health condition.

Keywords: Electronic Healthcare Record, EHR, MUDR, MUDRC

Kapitola 1

Úvod

1.1 Zadanie projektu

V rámci aplikovaného výskumu v **Európskom centre pre medicínsku informatiku, štatistiku a epidemiológiu** (EuroMISE centrum) bola vyvinutá aplikácia pilotného elektronického zdravotného záznamu nazvaná **MUDR**^[1] postaveného na 3-vrstvej architektúre s dátovou, aplikačnou a klientskou vrstvou.

Komunikácia s aplikačnou vrstvou prebieha pomocou **MUDR API** definovaného vo forme platných XML dokumentov odpovedajúcich definovanej XML schéme. MUDR klient typicky posiela požiadavky zabalené do XML pomocou HTTP POST žiadosti na HTTP server aplikačnej vrstvy. MUDR API definuje približne 20 rôznych príkazov pre základnú prácu s dátami elektronického zdravotného záznamu a jeho kompletnú správu úplne nezávisle na množine atribútov, ktoré sú v zázname uchovávané.

Práca s týmto rozhraním je pomerne náročná ako po stránke výpočtovej, tak po stránke komunikačnej. Navyiac, užívateľské rozhranie sa typicky pripojuje práve na jednu aplikačnú vrstvu a tá získava dáta práve z jednej dátovej vrstvy. V prípade, kedy by mal pacient informácie o svojom zdravotnom stave na rôznych miestach, nie je poskytnutá jednoduchá možnosť ich hromadného prehliadania.

Úlohou projektu **Intelligentní proxy pro mobilní uživatelská rozhraní elektronického zdravotního záznamu** je vytvoriť aplikáciu pracujúcu ako proxy server elektronického zdravotného záznamu MUDR, ktorá bude mať nasledujúce vlastnosti:

- Umožní jednoducho integrovať dáta z rôznych zdrojov (inštancií elek-

tronického zdravotného záznamu). Podporované inštanície elektronického zdravotného záznamu budú variabilne konfigurovateľné a proxy si navyše bude priebežne zisťovať a udržiavať zoznam pacientov obsiahnutých v jednotlivých inštaniciách.

- Proxy zahrnie výpočtovú logiku pre prácu s MUDR API a sprístupní tak elektronický zdravotný záznam pre jednoduché (tenké) mobilné užívateľské rozhrania. Za týmto účelom bude navrhnutý formát konfigurácie proxy, ktorý bude špecifikovať, akého druhu sú atribúty ukladané v elektronickom zdravotnom zázname a ako sa tieto atribúty budú zobrazovať (rozmiestňovať) na zariadeniach určitého typu. Pre mobilné užívateľské rozhrania bude poskytovaný výstup v tvare HTML či WAP konfigurovateľne prispôbostený druhu mobilného zariadenia. Pilotné otestovanie prebehne na niekoľko málo mobilných zariadeniach.
- Proxy bude navrhnutá do istej miery modulárne, aby do budúcnosti bolo možné pridať podporu ďalších typov rozhraní pre komunikáciu s klientskými aplikáciami (napr. CORBA, .NET Remoting, Web Services).

Pre lepšiu predstavu je v prílohe B priložená schéma prvotného návrhu.

1.2 Motivácia a ciele

EHR alebo **Electronic Healthcare Record**, teda elektronický zdravotný záznam, je zdigitalizovaná forma medicínskeho záznamu o pacientovi. Každý človek má už od narodenia svoj vlastný administratívny záznam, ako napríklad rodný list. Počas života sa k nemu pridávajú záznamy v kartotékach lekárov, či už praktických, kde sa registruje zdravotný stav a vývin osoby, alebo špecializovaných, ako napríklad záznamy z chirurgického, imunologického alebo ORL. V Slovenskej a Českej republike sa tento záznam v drvivej väčšine uchováva v klasických papierových kartotékach. Samotnou štrukturalizáciou nemocničného prostredia sa automaticky rozdeľuje záznam o pacientovi po spomenutých oddeleniach, pričom najkompletnejší obraz sa uchováva u praktického lekára, aj to vo veľmi obmedzenej podobe.

Takýmto spôsobom je pacient nútený podstupovať procedúry vyšetrenia u každého nového lekára, ku ktorému príde, čo môže viesť k strate dôležitých informácií o minulom stave pacienta, či aktuálnej liečbe. Zdigitalizovaná forma pacientovho záznamu, ktorá by obsahovala jeho zdravotný stav aj s

históriou, výsledky vyšetrení a diagnózy, by výnimočne uľahčila a zefektívnila prácu lekárov a život pacientov.

Najdôležitejšou súčasťou tejto digitalizácie by bola ale neporovnateľne ľahšia integrácia všetkých záznamov o pacientovi z jednotlivých oddelení, nemocníc a súkromných ambulancií, ktorú sa prakticky nedá predstaviť pri papierovom vedení pacientovho záznamu. Takisto sa dajú domyslieť iné administratívne výhody (napríklad komunikácia s poisťovňami). Preto sa už dlhší čas vo svete ale aj v Českej a Slovenskej Republike vyvíjajú a uvádzajú do prevádzky rôzne zdravotnícke systémy, ktoré by splňali horeuvedený popis a výhody elektronického spracovania zdravotného záznamu pacienta. Jednou z nich je aj projekt MUDR, na ktorom stavia táto práca.

1.3 Struktúra textu

Tu je zoznam jednotlivých kapitol, ktoré práca obsahuje, spolu so stručným popisom.

Druhá kapitola (Systémy EHR) popisuje zahraničné ale aj domáce projekty zaoberajúce sa problematikou elektronického zdravotného záznamu. Takisto informuje o základoch pre tento projekt - aplikácií MUDR a MUDRC a ako s tým všetkým súvisí MUDR Proxy.

Tretia kapitola (Návrh MUDR Proxy) obšírnejšie rozoberá požiadavky projektu a vysvetľuje a obhajuje návrh architektúry, ktorú si autor zvolil ako najlepšie riešenie pre uskutočnenie daných požiadaviek.

Štvrtá kapitola (Implementácia MUDR Proxy) je rozdelená na štyri časti - proxy server, databáza, CGI skript a webová stránka. Jednotlivé časti sú v nej podrobne rozopísané spolu s implementačnými detailami, spôsobom rozdelenia, funkčnosťou a zvolenými programovacími technikami.

Piata kapitola (Prínos a porovnanie projektu) sa snaží zhodnotiť celkový prínos aplikácie. Porovnáva projekt s inými projektami spomenutými v kapitole 2 a hľadá prínos aplikácie do prostredia medicínskej informatiky. Zároveň informuje o postupe vývoja a problémoch s tým spojenými.

Šiesta kapitola (Budúcnosť projektu) diskutuje nad pokračovaním vo vývoji aplikácie. Sú v nej vymenované možnosti rozšírenia jednotlivých

časť MUDR Proxy, ktoré sa naskytli počas doterajšieho vývoja, alebo sa nestihli doprogramovať.

Siedma kapitola (Záver) stručne zhodnocuje a uzatvára túto bakalársku prácu.

Kapitola 2

Systemy EHR

2.1 Systemy EHR v zahraničí

Na medzinárodnej scéne sú projekty EHR už v značne pokročilom štádiu vývoja a používania. Napríklad vo Francúzsku a Taliansku sa využíva elektronická zdravotnícka dokumentácia u 20-50% praktických lekárov, v Belgicku a Holandsku 50-60%. Na špičke sú Dánsko, Švédsko a Veľká Británia s 80% využívania EHR, pričom v niektorých prípadoch sa dokonca úplne vynecháva papierová dokumentácia. Na porovnanie: USA využívajú EHR iba v 2-5 percentách prípadov [1]. V Európe sa výskumom a vývojom EHR zaoberá značné množstvo pracovných skupín.

Jeden z hlavných projektov, na ktorom spolupracuje 9 európskych štátov a vyše 26 vývojových centier, nemocníc a univerzít, je **SynEx** (Synergy on the Extranet). Má za úlohu integrovať projekty ako **Synapses** (zdravotné záznamy pacienta), **HISA** (healthcare information systems architectures - architektúry zdravotných informačných systémov), **GALEN** (terminologické služby) a **ProForma** (formalizmy pre podporu rozhodovania). Používajú podobnú komunikáciu ako MUDR - pomocou protokolu HTTPS s dátami vo forme XML a takisto objektovo orientovanú databázu [2].

Európske centrum pre medicínsku informatiku spolupracovalo na významnom projekte **I4C - TripleC** (Integrace a komunikace pro kontinuitu kardiální péče), implementujúcom otvorený multimedialny systém pre elektronický záznam o pacientovi **ORCA** (Open Record for Care), z ktorého si MUDR berie zaujímavé vlastnosti ako štruktúrovaný záznam informácie, podporu viacerých jazykov a hierarchicky usporiadanú množinu údajov (tzv. znalostnú bázu) [3].

Projekt **GEHR** (The Good European Health Record) zase skúma lepšie možnosti integrácie a spolupráce heterogénnych EHR. Je zároveň hlavnou súčasťou tzv. **OpenEHR Foundation**, kde sa vyvíjajú open source distribuovateľné softwarové komponenty podporujúce všeobecný a prenosný elektronický zdravotný záznam (spolupráca 21 organizácií zo siedmich európskych krajín) [4].

Na medzinárodnej scéne figuruje veľa projektov zaoberajúcich sa vývojom EHR, pričom na najdôležitejších aktívne spolupracujú vlády a vývojové centrá, čo predstavuje serióznym záujemom o túto problematiku a predpovedá značné výsledky v krátkodobom hľadisku.

Takisto scénu EHR neobišli ani medzinárodné organizácie zaoberajúce sa štandardmi, ako napr. **ISO**, ktorá vyčlenila špeciálnu komisiu pre zdravotnícku informatiku - **TC215** (Health Informatics) [5]. Technická komisia **TC251** európskej organizácie **CEN** pripravuje predovšetkým európske normy a harmonizačné dokumenty [6]. Na medzinárodnej scéne sa takisto pracuje na klasifikácii chorôb, **WHO** presadila do používania napr. **International Classification of Diseases - ICD10** [7].

Veľmi dôležitou súčasťou EHR je komunikácia medzi zdravotníckymi zariadeniami, ktorú implementuje napr. v USA rýchlo vyvíjajúci sa a veľmi populárny štandard **HL7** (Health Level 7) [8], alebo v Českej republike DASTA (*"Datový štandard pro přenos dat mezi informačními systémy zdravotnických zařízení"*) [9].

2.2 Systémy EHR v Českej a Slovenskej Republike

U nás sa v tomto čase používajú prevažne tzv. **nemocničné informačné systémy** (NIS). Napr. v Českej Republike má integrovaný NIS asi 40 nemocníc a ďalších 60 používa rôzne izolované softvéry, ktoré medzi sebou nespupracujú a tak netvorí kompletný NIS. Ďalších vyše 100 nemocníc nemá vôbec integrovaný nemocničný informačný systém. [1]

Medzi najväčších dodávateľov v ČR patrí firma Stapro s.r.o, ktorá dodáva NIS **MEDEA** rozšírený v asi 20 inštaláciách. Bohužiaľ sa tento NIS obmedzuje na administratívne a riadiace úlohy nemocnice, ako sú napr. centrálna evidencia, evidencia hospitalizovaných, štatistika, informačná kancelária, vedenie záznamov o lôžkách a novorodencoch, poisťovňa, rádiodiagnostika, stravovanie, lieky a správa systému. Nemá integrovaný plnohodnotný elek-

tronický zdravotný záznam, takisto ako ďalšie produkty iných firiem, ktoré sa zaoberajú nemocničnými systémami alebo špecializovaným softvérom [10]. **Medical Process Assistant** je klinický informačný systém od fi. ICZ, ktorý má navyše podporu pre integráciu so systémom Európskej únie či medzinárodným štandardom **HL7**, **DICOM** (uchovávanie obrazovej dokumentácie), či **DASTA** [11].

Takisto informačný systém pre podporu činnosti zdravotníckych zariadení **HERMES** od fi. Logis s.r.o podporuje rôzne podnikové činnosti ako administratíva, finančné a ekonomické záležitosti, marketing či personalistiku, ale napríklad aj záznamy o bežných či špecializovaných vyšetreniach pacienta, ale ešte nemá integrovanú podporu medzinárodných komunikačných štandardov [12].

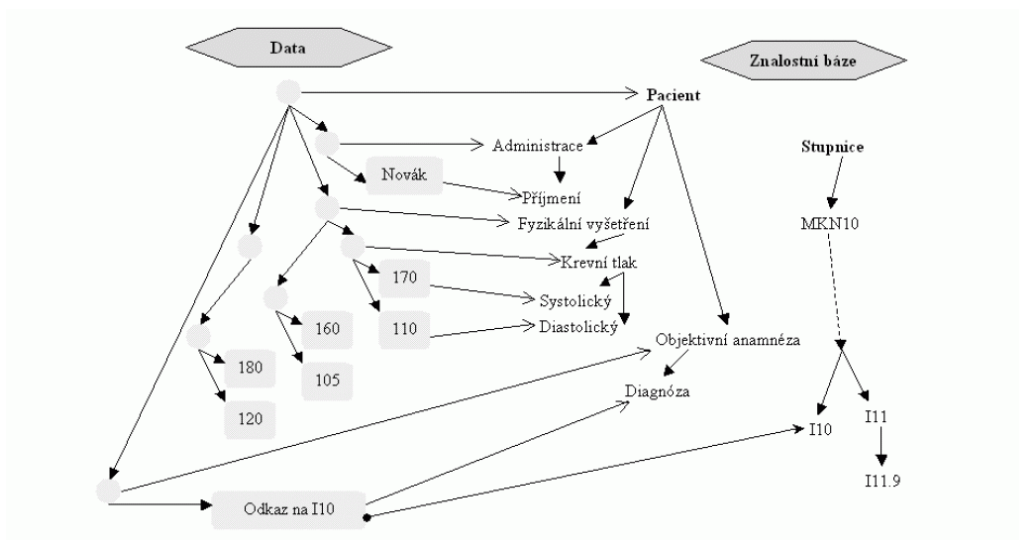
Slovenská firma LCS Electronics dodáva tzv. **Kompletný Nemocničný Informačný Systém**, zložený z modulov so štandardnými úlohami ako u ostatných systémov a má takisto podporu ambulantných vyšetrení a je zavedený v 12 nemocničných zariadeniach, avšak v aktuálnej dokumentácii sa takisto nevyskytla podpora pre medzinárodné komunikačné štandardy, čo ho značne izoluje od začlenenia do globálneho systému [13].

Ako je vidieť, na Slovensku či Českej Republike je viacero dodávateľov NIS alebo iných zdravotníckych softvérov, ale bez výraznejšej integrácie či spolupráce na vnútroštátnej či medzinárodnej úrovni, pričom žiaden z nich sa neorientuje na profesionálne spracovanie EHR.

2.3 Projekt MUDR

Projekt MUDR bol vyvinutý Josefom Špidlenom ako diplomová práca *”Databázová reprezentace medicínskych informací a lékařských doporučení”* [1] na KSI MFF UK v Prahe. Cieľom projektu bola implementácia elektronického zdravotného záznamu v databázovom systéme s efektívnou formou prístupu k dátam. Bol navrhnutý 3-vrstvový model: databáza - aplikačná (riadiaca) vrstva - klient.

Ako databáza bola zvolená **Oracle 9i**. Aplikačná vrstva je založená na **Win32 API** a s databázou komunikuje pomocou **Oracle Call Interface API**. Komunikácia s klientami prebieha pomocou protokolu **HTTP** s dátami vo forme **XML**, pre ktorú bolo vytvorené komunikačné **XML schéma**. Komunikácia prebieha pomocou pripojenia sa na **HTTP server** a predania XML požiadavku špeciálnemu **CGI skriptu**, ktorý ho ďalej predá aplikačnej vrstve. Tá po spracovaní požiadavku a komunikácii s databázou pošle



Obr. 2.1: Pacientove dáta a znalostná báza

späť odpoveď CGI skriptu, ktorý ju jednoducho vypíše na štandardný výstup, čím HTTP server zabezpečí prenos odpovede ku klientovi.

Samotné pacientove dáta sú uložené v hierarchickej štruktúre. Tá je tvorená uzlami, ktoré obsahujú odkaz na uzol **znalostnej bázy** (knowledge base, viď obr. 2.1). Uzly znalostnej bázy popisujú jednotlivé lekárske úkony, merania či administratívu. Dátové uzly pacienta naopak obsahujú okrem odkazu aj samotnú hodnotu - meranie či administratívny text.

2.4 Projekt MUDRC

Projekt MUDRC [14] bol vyvinutý na pôde **EuroMISE** centra v rámci softvérového projektu na MFF UK v Praze. MUDRC (Multimedia Distributed Record Client) je aplikácia, ktorá slúži ako klient pre aplikačnú vrstvu MUDR. Je naprogramovaná v Jave a poskytuje užívateľské rozhranie pre lekárov pracujúcich s elektronickým záznamom pacienta uloženým v databáze MUDR a obsahuje aj pilotnú implementáciu využitia lekárskeho odporúčenia pre podporu rozhodovania (napr. pri stanovovaní diagnóz). Obmedzením MUDRCa je to, že sa vie pripojiť vždy iba na jeden MUDR server.

2.5 Projekt MUDR Proxy

Keďže každý MUDR server je samostatná jednotka bez možnosti komunikácie s ostatnými MUDR servermi a každý MUDRC je odkázaný iba na jemu pridelený server, bolo ďalším logickým krokom vyvinúť aplikáciu, ktorá by poskytovala rozhranie pre jednotný a celistvý pohľad na pacientove dáta, bez ohľadu na to, na ktorom serveri sa nachádzajú. To je hlavnou úlohou MUDR Proxy - integrovať dáta pacientov z jednotlivých MUDR databáz a poskytnúť ich v požadovanej forme užívateľom používajúcim rôznych klientov na rôznych platformách a zariadeniach.

Kapitola 3

Návrh MUDR Proxy

3.1 Rozbor požiadaviek projektu

Za hlavné požiadavky projektu sa dajú označiť:

1. Integrácia pacientových dát z viacerých MUDR serverov
2. Prezentácia dát rôznym tenkým klientom
3. Použitie technológii pre komunikáciu medzi MUDR Proxy

3.1.1 Integrácia pacientových dát z MUDR serverov

Systém MUDR serverov je oddelený a navzájom nespolupracuje, rieši iba uloženie pacientovho záznamu v elektronickej podobe. Integrácia dát z viacerých serverov je preto ponechaná na proxy aplikáciu. Tá musí implementovať logiku pre komunikáciu s MUDR servermi a zároveň vedieť analyzovať pacientove uzly a pospojovať ich z viacerých serverov. Uzly obsahujú rôzne atribúty, ako napr. typ uzlu (adresár / text / číslo), hodnotu (textovú / číselnú), rôzne dátumy (zadania, potvrdenia, platnosti, zmazania) a iné.

Jedna zo základných myšlienok spájania pacientových uzlov je unikátnosť pacienta formou rodného čísla. MUDRC si pacienta pamätá ako trojicu <meno, priezvisko, rodné číslo>, ale proxy vychádza z jednoznačnosti rodného čísla, ktorá je akceptovaná a samozrejماً aj v štátnej administratíve. Pacient si mohol napr. zmeniť priezvisko, ale proxy ho bude považovať za toho istého človeka. Jediným problémom môže byť výnimočná duplicita rodného čísla, ktorá ale nie je problémom proxy a rieši sa okamžitým zmenením

čísla. Takéto chyby sa stávali v minulosti v ČR aj SR, ale v týchto dňoch by už mali byť eliminované.

3.1.2 Poskytovanie pacientových dát užívateľom

Základnou požiadavkou tohto projektu bolo integrovať systém, ktorý by dovolil pristupovať k dátam pacientov nezávisle na type užívateľovho webového prehliadača - tzv. tenkého klienta. Jednoduché príklady človeka hneď napadnú, keď si predstaví nasledujúce situácie:

1. Doktor v ordinácii má síce svoju kartotéku pacientov, ale v konečnom dôsledku v nej má iba zlomok informácií, ktoré už pacient poskytol množstvu lekárov počas svojho života. Nie je nič jednoduchšie ako prihlásiť sa na webovú stránku a skôr než pacient začne rozprávať o svojich zdravotných problémoch, už má o ňom všetky potrebné informácie (samozrejme iba tie informácie, na ktoré má právo).
2. Na miesto nehody je privolaná záchranná služba. Keď dorazí na miesto, je treba rýchlo konať na záchranu životov. Treba napr. podať lieky, ale lekári nevedia, či na ne nie je pacient náhodou alergický. Pomocou mobilu si vyhľadajú jeho záznam a dozvedia sa, čo mu môžu a čo nesmú podať.
3. Sanitka prinesie zraneného človeka po autonehode a je treba urýchlená operácia. O človeku nikto nič nevie, má síce pri sebe doklady, ale všetky údaje o ňom sú v databáze inej nemocnice. Sestrička si počas prípravy alebo operácie vyhľadá na PDA pacientove dáta, ktoré sú posťahované z nemocníc po celej republike. Dozvie sa o ňom všetko, napr. že ho treba špeciálne resuscitovať, pretože pacient má na srdci kardiostimulátor, alebo hrozí nebezpečenstvo lekárom pretože je HIV pozitívny.

Ako vidno, prípadov využitia proxy je nesčítateľne, a preto je potrebné, aby sa na web dalo pristupovať z čo najväčšieho množstva klientov. Webové stránky musia byť navrhnuté tak, aby typ užívateľovho klienta rozoznávali automaticky pri nahrávaní prvej stránky a poskytl mu interface priamo pre neho.

3.1.3 Použitie konkrétnych technológií

V zadaní projektu je požiadavka na komunikáciu aj medzi proxy servermi, nielen medzi proxy a MUDR. Zoberme si hypotetický príklad: nemocnica má niekoľko MUDR serverov, ktoré ale nie sú prístupné z internetu (napr. kvôli bezpečnosti). Na komunikáciu s internetom používa jeden server, na ktorom beží aj MUDR Proxy, aby boli dáta dostupné zvonku. V prípade dvoch takýchto architektúr nemocničných systémov by mala každá MUDR Proxy prístup iba k MUDR databázam jej nemocnice, čo je nedostatočné a nevyhovuje požiadavke kompletnej integrácie pacientových dát. Preto je dôležitá *"inter-proxy"* komunikácia, kde si každá proxy okrem MUDR serverov môže vypýtať pacientove dáta aj od ďalších dostupných MUDR Proxy, pričom si v tom prípade ušetrí značná časová a prenosová záťaž na komunikáciu s MUDR servermi. Pôvodný návrh zohľadňoval komunikáciu pomocou: CORBY, Web Services alebo .NET Remoting. Týmito technológiami sa dajú prenášať celé objekty a netreba vymýšľať vlastný komunikačný protokol. Vybraná implementácia je rozpísaná v kapitole 4.1.6.

3.2 Návrh architektúry

Po určení hlavných požiadaviek je nutné rozobrať jednotlivé detaily týkajúce sa architektúry MUDR Proxy.

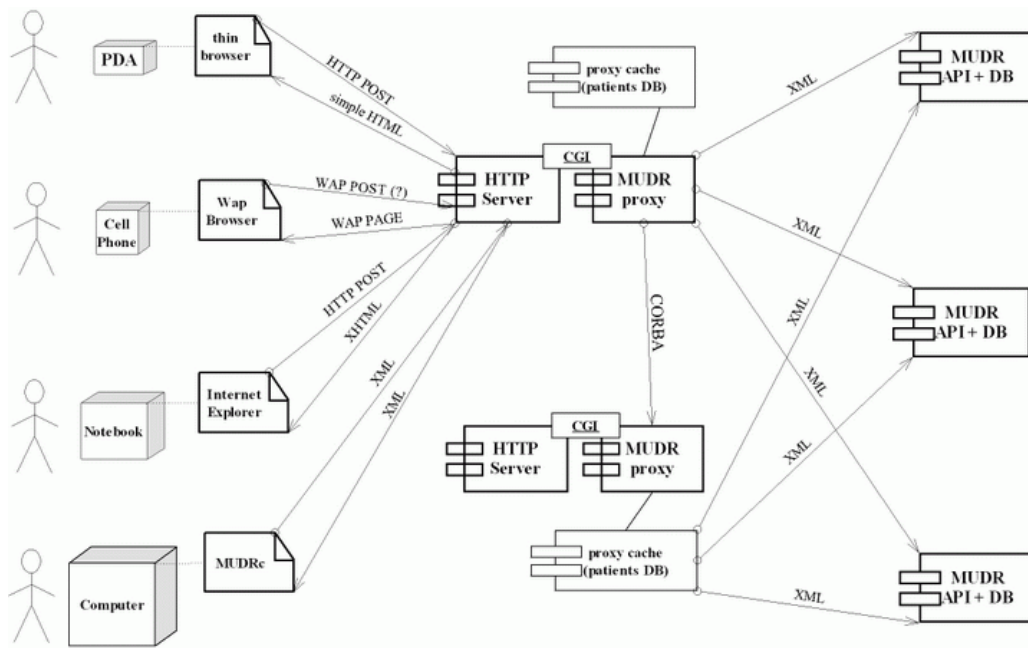
3.2.1 Rozdelenie funkčnosti

MUDR Proxy si berie príklad z MUDR servera, ktorý je rozdelený na viacero vrstiev (obr. 3.1).

- Databáza pacientov
- Riadiaca vrstva
- Prezentačná vrstva

3.2.2 Databáza pacientov

Pri vyhľadávaní pacientov je potrebný rýchly prístup k menám, priezviskám a rodným číslam ľudí, ktoré sú k dispozícii na MUDR serveroch. Preto je pre proxy dôležité pamätať si tieto údaje vo vlastnom zozname - databázi.



Obr. 3.1: Návrh architektúry

Pre užívateľa, napr. lekára je najdôležitejšou funkciou vyhľadanie pacienta podľa rodného čísla, ale keď nie je k dispozícii, tak podľa mena či priezviska. Tieto tri atribúty sú dostačujúce pre efektívne vyhľadanie pacienta, a preto je potrebné si ich v databáze udržiavať. Na MUDR serveroch sa ale pacienti môžu (a aj budú) vymazávať a pridávať, teda je potrebné si ich pravidelne aktualizovať. Komunikačný protokol ale poskytuje iba jeden prostriedok na prezeranie zoznamu pacientov - a to stiahnutie kompletného zoznamu. Neobsahuje jednoduchý prístup k pacientom novším ako daný dátum, alebo vymazaným po danom dátume, a teda je potrebné si pravidelne sťahovať celý zoznam. Preto sa databáza musí pravidelne aktualizovať stiahnutím nových zoznamov zo všetkých serverov.

Pri požiadavke na stiahnutie pacienta sa jeho dáta stiahnu zo všetkých dostupných MUDR serverov a MUDR Proxy a uložia k danému pacientovi. To je aj hlavnou výhodou proxy - ak bol raz pacient stiahnutý, proxy poskytuje rýchly prístup k jeho dátam bez potreby opätovného sťahovania z MUDR serverov. Pri aktualizácii zoznamu pacientov sa ale tieto dáta zmažú a pacient sa bude musieť opätovne stiahnuť.

Výhody sú:

- Najaktuálnejšia verzia pacientových dát po každom aktualizovaní databázy
- Uvoľnenie miesta v databázi (ak príde požiadavka na pacienta raz za dlhé obdobie (rádovo mesiac), je zbytočné si jeho dáta neustále udržiavať)

Nevýhody:

- Medzi jednotlivými aktualizáciami databázy sa pacient sťahuje iba raz a pri opätovnej požiadavke už môžu byť jeho dáta aktualizované na nejakom MUDR serveri. Táto chyba sa nedá obísť, pretože, ako je vyššie spomenuté, pacient na MUDR serveri nemá automaticky aktualizované časové razítka po zmene jeho uzlov, teda je nemožné zistiť, či sa medzičasom niečo k jeho dátam pridalo. Preto je to riešené kompletným stiahnutím raz za aktualizáčn é obdobie.
- Potreba sťahovať pacientove dáta každé aktualizáčn é obdobie

3.2.3 Zloženie a funkčnosť proxy serveru

Aplikácia MUDRP musí bežať na serveri neustále kvôli požiadavkám zo strany webového rozhrania, teda najlepšia možnosť je navrhnuť ju v štýle *service* (služba) vo Windows alebo *daemon* pod Unixom. Po analýze požiadaviek na funkčnosť ich autor rozdelil do nasledujúcich kategórií:

1. **Komunikácia s MUDR servermi** - proxy musí mať nástroj na jednoduchú a efektívnu komunikáciu s databázami pacientov (MUDR servermi), aby si mohla sama stiahnuť napr. znalostnú bázu alebo dátové uzly pacienta. Keďže komunikácia sa odohráva pomocou HTTP protokolu, je navrhnuté vlastné rozhranie, ktoré má za úlohu zostaviť hlavičku a telo HTTP žiadosti. Dátová zložka pri komunikácii aplikačnej vrstvy so svetom je implementovaná pomocou XML, takže je potrebné aj rozhranie pre narábanie s XML, predovšetkým vytváranie validných XML požiadaviek a parsovanie odpovedí.
2. **Vnútoraná reprezentácia dát** - je dôležité, ako si proxy bude pamätať dáta, s ktorými disponuje. Pomoc v návrhu poskytla aj reprezentácia pacienta aplikačnej vrstvy MUDR formou dátových uzlov. Pacienta a znalostnú bázu je najlepšie si reprezentovať ako strom dátových uzlov

s presne definovanou štruktúrou. Keďže proxy má za úlohu iba prezentáciu existujúcich dát, nie je potrebné, aby sa takáto štrukturalizácia implementovala aj na miesto perzistentného uloženia pacienta. Tým pádom stačí jednoducho *serializovať* dátovú štruktúru a uložiť ju perzistentne ako obyčajné binárne dáta alebo text.

3. **Komunikácia s vlastnou databázou** - jednou z najdôležitejších funkcií proxy je vyhľadanie pacienta vo vlastnej databáze, keďže tá integruje zoznamy pacientov zo všetkých aplikačných vrstiev. Vzorom pre návrh tabuľky pacientov bola aplikácia MUDRC. K hlavným identifikačným údajom pacienta ako je rodné číslo, priezvisko a meno sa tu pridávajú aj samotné serializované dáta pacienta. Tie sa mohli uložiť aj perzistentne na disk, ale keď už program disponuje databázou, ktorá vie sama efektívne narábať s uloženými dátami a vyhľadávať v nich, bolo rozumné uložiť dáta pacienta do nej.
4. **Komunikácia s webovým rozhraním** je aj forma riadenia samotnej aplikácie. Najpoužívanejším riešením komunikácie z webu je formou CGI programov, ktoré sa priamo dorozumejú s danou aplikáciou (CGI skript je analyzovaný v nasledujúcej sekcii). Proxy teda musí disponovať jednoduchým serverom, ktorý bude vedieť prevziať požiadavky od CGI. Keďže sa autor rozhodol pre aplikáciu vo forme *daemon*, je treba spravovať proxy iným spôsobom ako cez grafické užívateľské rozhranie (GUI). CGI skript a webová stránka sú najjednoduchším riešením, samozrejme nastavenia sa musia dať meniť aj cez parametre pri spúšťaní proxy a v textových konfiguračných súboroch. Cez web je to však týmto spôsobom jediná forma dynamickej konfigurácie. Poskytuje napr. aj možnosť manažovania aplikačných serverov, z ktorých proxy čerpá dáta.
5. **CORBA** Po analyzovaní výhod a nevýhod jednotlivých technológií spomenutých v zadaní projektu vychádza ako najlepšia voľba komunikácie medzi proxy CORBA. Dôvody:
 - Veľká škála rôzne náročných open source implementácií pre Windows a Unix a rôzne programovacie jazyky (Open source implementácie pre Web Services sú väčšinou orientované na Javu, .Net Remoting zas na Windows)
 - Rýchly komunikačný protokol IIOP (na rozdiel od SOAP pri Web

Services, ktorý je tvorený XML a používa HTTP, čo spôsobuje zväčšenie nárokov na prenos dát)

- Jednoduchá práca a definícia objektov v pseudokóde (na rozdiel od zložitých XML schém)
- Jednoduchá implementácia a použitie
- Dobrá užívateľská základňa a podpora pre rôzne distribúcie

Samozrejmosťou by mala byť dostatočná modulárnosť pre zmenu / pridanie ďalšej požadovanej technológie.

3.2.4 CGI skript

By mal byť čo najjednoduchší program, ktorý by sprostredkoval komunikáciu medzi webovým rozhraním a proxy serverom (a takisto v budúcnosti medzi MUDRCom (vid 2.4) a proxy). Pri každej požiadavke na stiahnutie pacienta HTTP server spustí CGI skript, ktorý odovzdá informácie proxy. Týchto požiadaviek môže prísť značné množstvo a preto musí CGI pracovať čo najrýchlejšie a mať malé nároky na výkon a pamäť. Takisto je treba zabezpečiť odovzdanie požiadavky proxy v čo najkratšej dobe.

3.2.5 Zloženie a funkčnosť webovej stránky

Úlohou webového rozhrania je poskytnúť užívateľovi jednoduchý prístup k dátam pacientov a spravovaniu proxy. Zo začiatku sa zdalo rozumné riešenie prenechať všetku logiku CGI skriptom, ktoré boli aj tak potrebné pre komunikáciu s proxy. Pre každý typ klienta by sa naprogramoval CGI, ktorý by zahrňoval logiku zobrazovania dát pacienta, ako aj jednotlivých stránok. Tento prístup sa ale ukázal ako veľmi neefektívny, príliš náročný na implementáciu a hlavne nerozumný. Pre programovanie webových stránok sú dostupné rôzne nástroje, ako je napr. PHP, ktorý bol zvolený pre vytvorenie základnej kostry webu. Bol tu však ďalší problém, a to požiadavka na odlišné prezentovanie tej istej informácie na rôznych tenkých klientoch. Tu sa naskytla príležitosť pre XML a XSL, čo mi dovolilo oddeliť dáta od ich prezentácie a cez nástroje PHP podporujúce prácu s XML a XSL ich jednoducho zlúčiť do požadovanej formy. Logika CGI skriptu sa tým pádom stenčila na jednoduché odovzdanie požiadavky medzi PHP a proxy. Výhoda je jasná - netreba programovať viacero CGI, stačí navrhnúť XSL stylesheet pre jednotlivých klientov. Veľká výhoda je, že sa to dá robiť počas ostrého

behu celej aplikácie, stačí pridať stylesheety a nakonfigurovať stránku aby ich používala ak sa pripojí užívateľ s daným klientom.

V rámci funkčnosti bolo treba splniť nasledujúce požiadavky:

- Prezeranie zoznamu pacientov
- Vyhľadávanie pacienta prostredníctvom mena, priezviska alebo rodného čísla
- Výber jazyka
- Prihlasovanie a nastavovanie užívateľských údajov
- Výber spôsobu zobrazovania pacientových dát
- Administráciu užívateľov, MUDR a proxy serverov

Výsledná funkčnosť sa môže samozrejme časom meniť, hlavne pridávať nové funkcie pri práci s pacientovými dátami či možnosť administrácie serverov. Základom je potreba jednotnej kostry, aby sa akákoľvek aktualizácia stránky objavila na každom klientovi okamžite, čo bolo hlavným dôvodom použitia PHP, XML a XSL.

Kapitola 4

Implementácia MUDR Proxy

4.1 Proxy server

Ako vyplynulo z návrhu, proxy server by mala byť osobitná aplikácia, ako prostredník medzi MUDR servermi a webovou stránkou, určená na sťahovanie pacientových dát, zlučovanie a uloženie v databázi vo forme prístupnej pre webovú stránku. Jednotlivé implementačné časti sú rozpísané v nasledujúcich sekciách, na obr. 4.1 je možné vidieť výsledný *class diagram* proxy servera.

4.1.1 Implementačné prostredie

Implementačným prostredím sa myslí prostredie operačného systému. Mezi hlavné OS patria **Windows**, **Unix/Linux** a **MacOS**. Keďže proxy bude aplikácia serverového typu, je najlepšou voľbou implementovať ho pre platformy Unix/Linux. Výhodou je vzdialená administrácia, stabilita systému a samozrejme výkon, ktorý aplikácia tohto typu potrebuje. Keďže databázový MUDR server je implementovaný pre platformu Windows, bolo logickým krokom naprogramovať proxy aj pre túto platformu, predsa len sa používa v nemocničnom prostredí veľmi často ako serverové riešenie (a skoro všade ako klientské stanice).

4.1.2 Programovací jazyk

Zo zvolených platforiem vyplýva, že by proxy mala byť naprogramovaná programovacím jazykom dostupným na oboch platformách. Medzi naj-

používanéjšie patria **C**, **C++**, **C#**, **Java** a **.NET**. Posledná možnosť je samozrejme nevyhovujúca pretože poktýva iba platformu Windows, jazyk **C#** zase vyšiel zo súboja zle kvôli nedostatku knižníc, neskôr potrebných pre správne fungovanie MUDR Proxy.

Java je jedným z najrozšírenejších a najpopulárnejších jazykov dneška, pokrývajúca obidve potrebné platformy, avšak autor sa rozhodol ju nepoužiť kvôli oslabenému výkonu oproti natívnemu jazyku obidvoch platforiem - C. Java má veľmi veľkú užívateľskú základňu, výborné balíčky, knižnice a vývojové prostredia, ale nehodí sa na aplikáciu v štýle *daemon* (server, služba). Ku svojmu chodu potrebuje Java Runtime Environment, čo je záťaž na pamäť a výkon systému.

Najrozumnejším riešením vyšiel jazyk C++, ktorý má už stabilne dobrú programátorskú základňu, množstvo open-source knižníc a výborný výkon na Windows aj Unixe. Nemenej dôležitý bol aj fakt značnej autorovej znalosti tohto jazyka na rozdiel od Javy. Program v C++ sa musí kompilovať pre každú platformu zvlášť, čo je ale bežné a nepovažujem to za hendikep, ak ide o serverovú aplikáciu. Okrem toho nemá žiadnu nevýhodu pri použití správnych knižníc a dodržiavaní **ANSI/ISO** štandardov, ktoré zaručujú prenositeľnosť kódu.

4.1.3 Implementácia proxy servera

MUDR server je implementovaný pomocou **Win32 API**, čo ho predurčuje iba na platformu Windows. Aby sa nemusela písať MUDR Proxy na každú platformu zvlášť, je potrebná vhodná knižnica, na ktorej by bola aplikácia postavená, a ktorá by pokrývala obidve platformy. Medzi najznámejšie patria **wxWidgets** [15] (v starších verziách **wxWindows**) a **Qt**, obidve predovšetkým zamerané na grafické prostredie. V prvotnom návrhu sa uvažovalo urobiť MUDR Proxy s grafickým rozhraním pre pohodlnú administráciu, ale časom sa aplikácia vyvinula do čisto konzolového rozhrania, aby sa zvýšili jej možnosti použitia a nasadenia na rôznych systémoch. Samotná administrácia sa presunula do externých aplikácií ako CGI skript (v spolupráci s web stránkou, čo prinieslo tiež lepšie možnosti administrácie).

Týmto sa na grafické prvky obidvoch knižníc prestal brať ohľad a uvažovalo sa iba o funkčnej časti. Tu sa autor rozhodol pre **wxWidgets** (keďže mala lepšie licenčné podmienky ako **Qt**), čo sa neskôr ukázalo ako mierna komplikácia pre nedostatočnú funkčnosť a kvalitu jadra knižnice, na ktorej je aplikácia postavená.

Jednou z dôležitých úloh aplikácií v štýle *daemon* je uchovávanie tzv. **logov** - textových súborov s obsahom ich činnosti. Takéto programy pracujú samostatne a teda ich treba monitorovať cez logy. Je to potrebné pre ladenie, vyhľadávanie chýb a inšpekciu činnosti. wxWidgets síce poskytuje logovacie mechanizmy, ale po širšom prieskume v oblasti logovacích knižníc sa autor rozhodol použiť knižnicu **log4cplus** [16] (C++ port známej Javovskej log4j), ktorá poskytuje veľmi dobrú škálovateľnosť logovacej činnosti.

4.1.4 Komunikácia s MUDR servermi

Jednou zo základných častí MUDR Proxy je komunikačné rozhranie s MUDR servermi. Model komunikácie je presnejšie zobrazený na obr. 3.1. Knižnica wxWidgets obsahuje dostatočné nástroje na prácu so socketmi a sieťovým rozhraním. Komunikáciu pomocou HTTP sa autor rozhodol riadiť sám, čo nebol problém pri odosielaní požiadavku MUDR HTTP serveru (v momentálnej verzii je to Apache [17]). Ťažkosti nastali až pri prijímaní odpovede, kde bolo treba "vyčistiť" odpoveď od nepotrebných HTTP hlavičiek a taktiež dátovú časť, ktorá bola určitým spôsobom deformovaná.

Komunikačný protokol MUDR je založený na XML schéme, pomocou ktorej je potrebné vytvárať validné XML požiadavky a parsovať prijaté odpovede. Na tento účel je potrebný XML parser použiteľný na požadovaných platformách. Po vyskúšaní viacerých distribúcií vyšiel ako najkvalitnejší a najrobustnejší **Xerces-C++** od **Apache** [18]. Ostatné (ako napr. XML vo wxWidgets) boli buď nedostatočné, málo zdokumentované, určené iba pre jednu platformu alebo mali malú podporu.

Kvôli komunikácii s MUDR servermi bolo treba použiť ďalšiu knižnicu, a to na prekonvertovanie textu z kódovania CP1250 (Windows-1250), ktoré používa MUDR server, na UTF-8, ktoré vie spracovať Xerces-C++ parser (okrem toho sú v UTF-8 aj web stránky a databázové tabuľky, takže to bolo potrebné kvôli konzistencii). Rozhodnutie o použití knižnice bolo jednoznačné, bola použitá najznámejšia pod licenciou GNU - **iconv** [19].

Integrácia komunikácie vlákna s MUDR servermi je zobrazený na diagrame sťahovania pacientových dát na obr. 4.2.

4.1.5 Komunikácia s CGI skriptom

Komunikácia s CGI skriptom je na MUDR serveri implementovaná pomocou zdieľanej pamäte (Win32 API). wxWidgets má tiež vstavanú medziproce-

sovú komunikáciu, ktorá môže byť formou **DDE** (Direct Data Exchange, iba na Windows) alebo **TCP/IP**. Keďže sa vyskytli isté problémy s DDE, na oboch platformách sa použilo rozhranie TCP/IP, ktoré prináša určité spomalenie v komunikácii, ale na druhú stranu väčšiu použiteľnosť, pretože tým pádom sa MUDR Proxy a CGI skript nemusia nachádzať na rovnakom stroji. To umožňuje rozdeliť proxy server a webovú časť na oddelené počítače, čo prináša opäť väčšiu modularitu. Samotná komunikácia je prostredníctvom socket serveru na strane proxy, na ktorý sa pripája CGI skript a odosiela požiadavky.

4.1.6 Komunikácia medzi proxy servermi

”Inter-proxy” komunikácia je riešená pomocou **CORBA**. Rozhodnutie padlo po analýze jednotlivých technológií (CORBA, Web Services, .NET Remoting, viac v časti 3.2.3). Každá proxy má svoj vlastný server, na ktorý sa môžu pripájať ostatné a požiadať o pacientove dáta. V momentálnej verzii sa táto komunikácia obmedzuje na predávanie pacientových objektov (dát v podobe hierarchickej štruktúry - stromu, viac v časti 4.1.8). Každá proxy si vo svojej databáze (ktorá je administrovaná cez webové rozhranie) udržiava adresy iných MUDR Proxy. Pripojenie je možné viacerými spôsobmi, avšak každý z nich potrebuje rozpoznať tzv. **IOR** (Interoperable Object Reference) kód. Ten, keďže je k dispozícii web stránka ku každej proxy, je poskytnutý na stiahnutie z danej web stránky. Proxy ho tam môže sama nahráť, pokiaľ zdieľa s web stránkou jeden systém, alebo ho tam nahrá administrátor, ak sú systémy oddelené. Tento postup sa môže do budúcnosti meniť, napr. použitím tzv. **Naming Service** (ktorý bol tiež testovaný v pilotnej verzii). Ako distribúcia CORBY je použitá (po porovnaní s inými) malá knižnica MICO [20] postačujúca na tento účel.

4.1.7 Vývojové a implementačné postupy

V tejto sekcii sú popísané implementačné postupy (design patterns a ich použitie), na ktorých je založené fungovanie MUDR Proxy. Takisto sú tu obsiahnuté niektoré detaily približujúce prácu s dátami pacienta a konkrétnejšie detaily o použitých programovacích technikách.

Thread pool

Kvôli simultánnym vybavovaniam požiadaviek na stiahnutie pacientov, ktoré môžu trvať rozlične dlho, sú v proxy implementované vlákna. Jednou z hlavných zlepšení výkonu je vždy obmedzenie volania systémových funkcií a pri značnom vyťažení akejkoľvek aplikácie je vytváranie vlákien zbytočná réžia. V prvotnej implementácii sa pri každej novej požiadavke (z CGI skriptu) vytvorilo nové vlákno, čo malo za následok ich nekontrolovateľné vytváranie a zanikanie a s tým spojené potenciálne vyvolanie výnimky pri nedostatku pamäte či nemožnosť regulovať záťaž proxy na systém. Dobrým vzorom na odstránenie tejto nedokonalosti bol webový server Apache, ktorý má implementovaný tzv. *thread pool* [21]. V jednoduchosti je thread pool vlastne zoznam vlákien, dopredu vytvorených a zanikajúcich až pri skončení programu. Tieto vlákna postupne obsluhujú požiadavky (podľa určitého algoritmu), žiadne nové nevznikajú či nezanikajú počas chodu programu. Týmto dostáva aplikácia relatívne pevné systémové prostriedky a jej náročnosť na systém sa počas chodu viac-menej nemení. Takisto sa dá veľmi dobre manažovať chod aplikácie, keďže každé vlákno, ktoré treba pri skončení programu osobitne zastavovať, je registrované v zozname. Samozrejmosťou proxy je variabilný počet spustených vlákien, ktorý sa dá nastaviť pri spúšťaní alebo cez konfiguračný súbor. S pridaním thread poolu sa ale skomplikovalo vybavovanie požiadaviek, keďže pri príchode požiadavky nemusí byť každá hneď vybavená (ako pri dynamickom vytváraní vlákien), ale musí sa dať do nejakej fronty, kde bude čakať na vybavenie. Toto ma primälo k vytvoreniu **work queue**.

Idea work queue

Work queue [22] je vlastne jednoduché zapuzdrenie **Jobov** a prístup k nim. Job je objekt, ktorý v sebe obsahuje parametre úlohy, ako napr. typ úlohy, pointer na socket (pri komunikácii s CGI), pointer na znalostnú bázu či pacienta a v neposlednom rade aj pointer na počet vlákien, ktoré spracovávajú tú istú sériu Jobov (tzn. sťahovanie jedného pacienta z viacerých MUDR serverov viacerými vláknami). Trieda WorkQueue poskytuje rozhranie pre thread-safe pridávanie či uberanie Jobov.

Práca s Jobom

Pre názornú ukážku, ako funguje thread pool s work queue v programe, je tu rozpísaný "životný" cyklus Jobu (na príklade požiadavky z CGI: "Stiahni pacienta s rodným číslom 1234567890"):

Vytvorenie nového Jobu na základe požiadavky z CGI. CGI server iba naalokuje štruktúru Job s danými objektmi (socket) a vloží do WorkQueue podľa priority (na začiatok alebo na koniec fronty). Je to kvôli urýchleniu vybavovania - požiadavky všeobecného charakteru (stiahni pacienta *XXX*), ktoré vygenerujú ďalšie množstvo Jobov, sa dávajú až za požiadavky konkrétnejšie (stiahni pacienta *XXX* zo serveru *YYY*), aby voľné vlákna vybavovali rovnakého pacienta a ďalší príde na radu až po ňom. Teda CGI server vygeneruje Job s tým istým socketom spájajúcim proxy a CGI.

Prevzatie Jobu z WorkQueue. Pri vložení Jobu do WorkQueue sa automaticky zavolá metóda semaforu, čo spôsobí zobudenie jedného vlákna. Zobudené vlákno si vypýta Job (WorkQueue mu poskytne prvý vo fronte - tj. ten s najväčšou prioritou).

Spracovanie Jobu. Vlákno si zoberie socket a načíta požiadavku, rozparuje ju a zistí, že je treba stiahnuť pacienta. Vygeneruje teda nový Job typu "stiahni pacienta zo všetkých serverov" s daným rodným číslom a uzavrie socket s tým, že akceptuje požiadavku od CGI. Toto vlákno nečaká kým sa stiahne pacient, ale vymaže svoj už nepotrebný Job (na komunikáciu s CGI) a uspí sa.

Vytvorenie série Jobov. Vo fronte je určite aspoň jeden ďalší Job, ktorý vlákno práve vygenerovalo a tak si ho hneď zoberie, v tomto prípade sa vnorí do funkcie, v ktorej musí vygenerovať pre každý server a uzol pacienta na ňom osobitný Job. Dynamickou alokáciou vytvorí nového zdieľaného pacienta a z databázy si naťahá zoznam *serverX-uzolY,serverZ-uzolQ* a podľa počtu týchto dvojíc vytvorí dynamickú premennú **counter** - počítadlo. V nej je uložený počet vlákien, ktoré sťahujú toho istého pacienta z rôznych serverov MUDR a zapisujú do zdieľaného stromu pacienta. Tak isto zapíše do každého Jobu dáta o serveri z ktorého sa má sťahovať pacient a jeho uzol.

Prevzatie, spracovanie a ukončenie série Jobov. Vlákno dostalo ďalší Job, ktorého úlohou je stiahnutie pacienta z konkrétneho serveru. Z

Jobu si vytiahne jeho uzol, server, pointer na pacienta a `counter`. Pomocou `NetHandleru` a `XMLHandleru` (konkrétnejšie sú spomenuté triedy rozpísané v programátorskej dokumentácii na priloženom cd) vytvára, posiela, prijíma a spracováva požiadavky - uzly pacienta. Zároveň každý nový uzol zapisuje do zdieľaného objektu `Patient` ???. Keď skončí svoju činnosť, zmenší `counter` o 1, takže keď sa dostane na nulu, vlákno spozná, že ostatné vlákna spracovávajúce túto sériu už skončili a celý pacient sa stiahol. Môže ho teda pomocou `ODBCHandleru` zapísať do databázy. Zároveň uvoľní zdieľané premenné a objekty `Patient` a `counter` a ako každé vlákno zmaže svoj Job (pri obyčajnom zmazení Jobu sa logicky zdieľané premenné nemažú - preto je potrebný `counter`).

4.1.8 Práca s dátami pacienta a znalostnej bázy

Uzly znalostnej bázy (KB - knowledge base) ako aj uzly pacienta sú na MUDR API serveroch identifikované desaťmiestnym unikátnym číslom. Na každom serveri má ten istý uzol znalostnej bázy iné id. Toto bol prvotný problém pri spracovávaní uzlov z rôznych serverov - ako zistiť, že uzol je ekvivalentný? Pamätať si pre každý server osobitne id uzlov pacienta a znalostnej bázy by bolo veľmi komplikované. Preto si autor vytvoril iný model založený na názvoch uzlov, ktoré sú unikátne a spoločné pre všetky MUDR servery.

Štruktúra stromov

Stromy pacienta a znalostnej bázy pozostávajú z uzlov, ktoré obsahujú dátové položky (v prípade pacienta ďalšie štruktúry obsahujúce namerané či zadané hodnoty lekárom; v prípade KB štruktúra obsahujúca popisy daného uzla v rôznych jazykoch). Tieto uzly ďalej obsahujú zoznam nasledovníkov tvorený dvojicami <názov uzlu, uzol>, kde názov uzlu je štandardizovaný názov v znalostnej báze. Tento názov mi značne pomohol, pretože pre každý uzol KB na každom serveri je rovnaký, na rozdiel od `node_id` a `dt_id` (desaťmiestne čísla) pacienta. Zároveň sa posiela aj v komunikačnom protokole pri každom uzle, takže pamätanie si `node_id` a `dt_id` úplne odpadlo. Týmto spôsobom je aj štruktúra stromov tried `KB` a `Patient` identická, čo zase uľahčuje zlučovanie. OBR

Zlučovanie stromov

Keďže sa pacient sťahuje z viacerých serverov, musia sa uzly pacienta zlučovať. Prvotná idea bola iba uložiť korene stromov a do zoznamu. Tu by však bol problém so serializovaním pacienta a vylučovaním uzlov, ktoré užívateľ nechce vidieť. Takže trieda `Patient` poskytuje rozhranie pre pridávanie uzlov na základe ich názvov, a to thread-safe, pričom ak strom už daný uzol obsahuje, iba sa do neho pridá nová dátová položka, ktorá sa ukladá zároveň s uzlom. V tomto prípade sa teda pri sťahovaní KB a pacienta obchádza zlučovanie stromov.

4.2 Databáza

Databáza sa po zanalyzovaní požiadaviek a možných dotazov skladá zo štyroch tabuliek. Samozrejmosťou je tabuľka pacientov, k nej sa pridávajú tabuľky pre uchovávanie MUDR a MUDR Proxy serverov, s ktorými proxy komunikuje a takisto tabuľka prihlasovacích údajov pre správny a bezpečný chod webovej stránky.

- Tabuľka **PATIENTS** obsahuje meno, priezvisko a rodné číslo pacienta. Ako primárny kľúč slúži rodné číslo, pretože je to unikátne 10miestne číslo (bez lomítka), najviac používané na vyhľadávanie konkrétneho pacienta. Samozrejme zlepšenie vyhľadávania umožňuje indexovanie položiek meno a priezvisko, aby si užívateľ (lekár) mohol efektívne vyhľadať pacienta bez znalosti rodného čísla. Pre technické účely slúži položka `nodes` - uzly. Sú to dvojice <číslo MUDR servera, číslo uzlu>, pomocou ktorých prebieha sťahovanie pacientových dát z jednotlivých MUDR serverov, na ktorých má pacient záznam. Poslednou položkou sú samotné pacientove dáta. Uložené sú v textovej forme XML, ktorá je automaticky vygenerovaná (serializovaná) z C++ objektu (internej reprezentácie pacientových uzlov) pomocou knižnice **Boost** [23].
- **USERS** je tabuľka užívateľov webovej stránky. Je dôležitá hlavne z bezpečnostných dôvodov, aby sa k údajom pacienta nedostal nepovolaný užívateľ. Okrem základných prihlasovacích údajov obsahuje typ užívateľa, ktorý môže byť administrátor alebo lekár. Takisto si pamätá nastavený jazyk, či skupinu uzlov, ktorá sa má zobrazovať pri prezeraní pacientových dát (bližšie v ??).

- **MUDR_SERVERS** - v tejto tabuľke sú uložené informácie potrebné pre komunikáciu s MUDR servermi. Nutná je prezencia jednej z dvoch položiek - IP adresa alebo hostname, ďalej port, na ktorý sa treba pripájať, adresu CGI skriptu, ktorú treba špecifikovať v hlavičke HTTP protokolu a ID - identifikátor, podľa ktorého sa rozlišujú uzly pacientov.
- **PROXY_SERVERS** tabuľka slúži na uchovávanie informácií k pripojeniu na iné MUDR Proxy, obsahuje tie isté informácie ako predchádzajúca tabuľka, až na CGI skript, ktorý nahradila adresa IOR reťazca, pretože komunikácia prebieha pomocou CORBA.

V testovacej verzii MUDR Proxy je použitá databáza MySQL [24].

4.3 CGI skript

CGI skript je nakoniec v projekte iba jeden - na rozdiel od počiatočnej idey programu. Je to v podstate jednoduchá konzolová aplikácia, taktiež ako proxy založená na knižnici wxWidgets, ktorá jej umožňuje nezávislosť na platforme či pohodlnú prácu so socketmi a TCP/IP.

CGI vykonáva iba jednu jednoduchú funkciu - načítať parametre príkazového riadka (ktoré mu z webového rozhrania vygeneruje PHP skript), spracovať ich do podoby v akej ich predá proxy a spojiť sa s ňou. Pri spojení automaticky predá dáta a skončí.

Komunikáciu medzi PHP a proxy by sa dalo riadiť aj bez CGI skriptu rôznymi PHP direktívami, avšak existencia CGI skriptu je podnietená (okrem projektu MUDR, vid 2.3) budúcou možnosťou komunikácie MUDRC - MUDR Proxy, kde by už bol potrebný. Preto sa dopredu začal vývoj CGI skriptu, aby sa prišlo na prípadné komplikácie.

4.4 Webová stránka

Webová stránka je hlavnou interakčnou časťou celého projektu. Slúži najmä pre lekárske účely - zoznam pacientov, vyhľadanie pacienta, prezeranie pacientových dát. Možno ju ale využiť aj na prvky manažmentu proxy ako je napríklad stiahnutie zoznamu pacientov, či samotné zapnutie alebo vypnutie proxy v prípade potreby. Hlavné požiadavky boli jednoduchý prístup k pacientovým dátam a podpora rôznych tenkých klientov, čo sa navonok

zdá ako celkom jednoduchá úloha, avšak pod povrchom obsahuje viacero netriviálnych záležitostí, ktoré sa museli vyriešiť. V nasledujúcich sekciách sú podrobnejšie rozobraté technológie a metódy použité pri spĺňaní týchto požiadaviek.

4.4.1 Univerzálnosť

Hlavným problémom bolo vymyslieť spôsob oddelenia dát a štruktúry stránok od prezentácie. Samozrejmosťou bolo zvolenie jazyka PHP [25] pre dynamické vytváranie stránok, či už kvôli komunikácii s databázou alebo spúšťanie CGI skriptu. Jednou z možností bolo programovať v PHP dynamické stránky pre každý typ klienta, čím by sa mohli využiť výhody každého klienta osobitne. Toto sa však ukázalo ako neprijateľné, keďže klientov je neznámy (potenciálne vysoký počet). Tým pádom zostalo na PHP iba logická kostra web stránky, ktorá je nezávislá na spôsobe zobrazovania.

4.4.2 Kostra stránky

Web stránka sa skladá z viacero PHP skriptov, ktoré používajú univerzálne funkcie. Medzi ne patrí napríklad volanie XML parsera, rozlišovanie agenta či komunikácia s databázou. V neposlednom rade sa tu dynamicky zisťuje, či má užívateľ právo pristupovať k danej stránke, ako aj pomocou premennej `$_SESSION` si pamätať jeho nastavenia, ako je napríklad jazyk. Každá stránka sa skladá z troch častí - hlavička, telo a päta. Je to z toho dôvodu, že hlavička zostáva stále rovnaká: je v nej menu pre daný typ užívateľa a takisto hlavičkové informácie HTML súboru. V tele sa menia stránky podľa užívateľovej interakcie a v päte je ukončenie html tagov + ďalšie informácie. Každá časť sa generuje z XML súborov pomocou XSL stylesheetu.

4.4.3 Generovanie obsahu

XML [26] je stále populárnejší formát ukladania dát v užívateľsky prívetivej forme. Zároveň sa dá jednoducho spracovať tak, že dané dáta sa zobrazia na základe naprogramovaného algoritmu spracovania - XSL transformácia [27]. Toto umožňuje naprogramovať obsah každej stránky iba raz a zobraziť ho nepreberným množstvom rôznych spôsobov. Tým pádom sa logicky oddelili dáta od ich prezentácie - dáta sú uložené iba raz v súboroch XML a prezentácia pomocou XSL je pre každý typ klienta odlišná. Zmena dát sa auto-

matically prejaví pri každom spôsobe ich zobrazenia. Na spôsob uchovávanía dát v XML si bolo potrebné vytvoriť vlastnú schému. Táto schéma dovoľuje pohodlne pracovať s rôznymi nastaveniami, ako napr. prístup k položkám stránky. Napríklad hlavné menu, ktoré sa zobrazí všetkým užívateľom (neprihláseným, administrátorom a lekárom) sa nachádza v jedinom súbore. Je však rozdiel, na ktoré položky menu má užívateľ prístup a preto sa niektoré neprihláseným nezobrazia, pričom administrátorovi sa zobrazia informácie o užívateľoch a lekárovi naopak databáza pacientov. Jednoduchým manažovaním prístupových práv priamo v dátových položkách v XML sa dá ovplyvňovať prístup užívateľa k určitým dátam. Tak isto všetky položky web stránky - od odkazov až po formuláre sú uložené v špeciálnej forme, ktorá ich dovoľuje spracovávať rôznymi štýlmi (stylesheets) a tak ovplyvňovať ich spôsob zobrazenia. Dôležitý prvok takejto formy je tiež určovanie jazyka jednotlivých položiek web stránky, ako je napríklad menu. Každý prvok stránky (ak je z neho niečo viditeľné) má uložený popis v štruktúre, ktorá udáva ISO kód jazyka, v ktorom je položka popísaná. To poskytuje neobmedzené možnosti pridávania jazykových mutácií stránok - stačí ku každému prvku stránky pridať popis v danom jazyku.

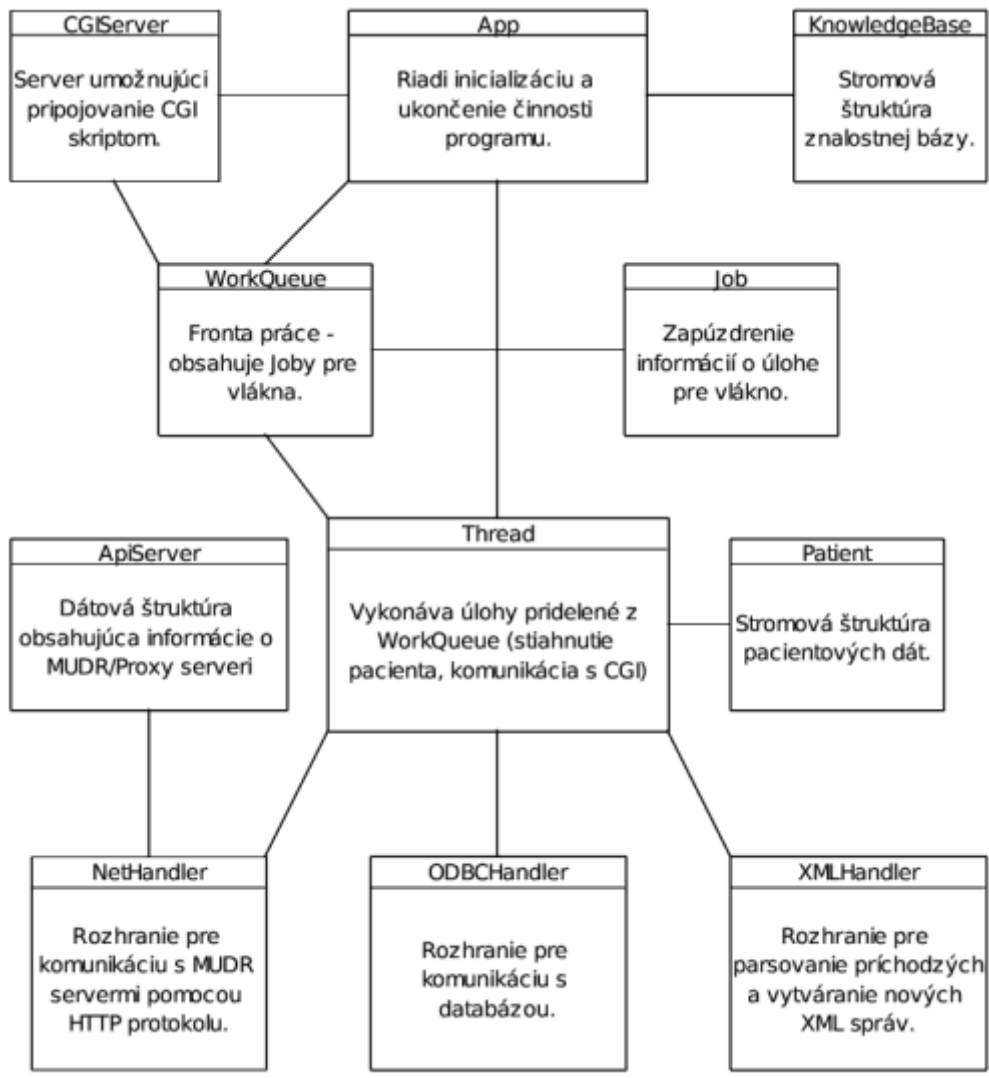
4.4.4 Užívateľské prostredie

V neposlednom rade treba poskytnúť aspoň minimálny manažment v zmysle užívateľských kont, či sledovanie stavu proxy. Samozrejme, že pacientove dáta nie sú verejné, a preto by k nim mali mať prístup iba povolení užívateľa (lekári). Dokonca ani administrátori webových stránok by k nim nemali mať prístup. Preto majú webové stránky vlastnú databázu užívateľov rozdelených na administrátorov a lekárov. Táto databáza môže byť súčasťou databáze proxy (ako jedna z tabuliek; v prípade že proxy beží na rovnakom stroji ako web), alebo ju môžu mať stránky vlastnú. Medzi základné dáta o užívateľoch patria:

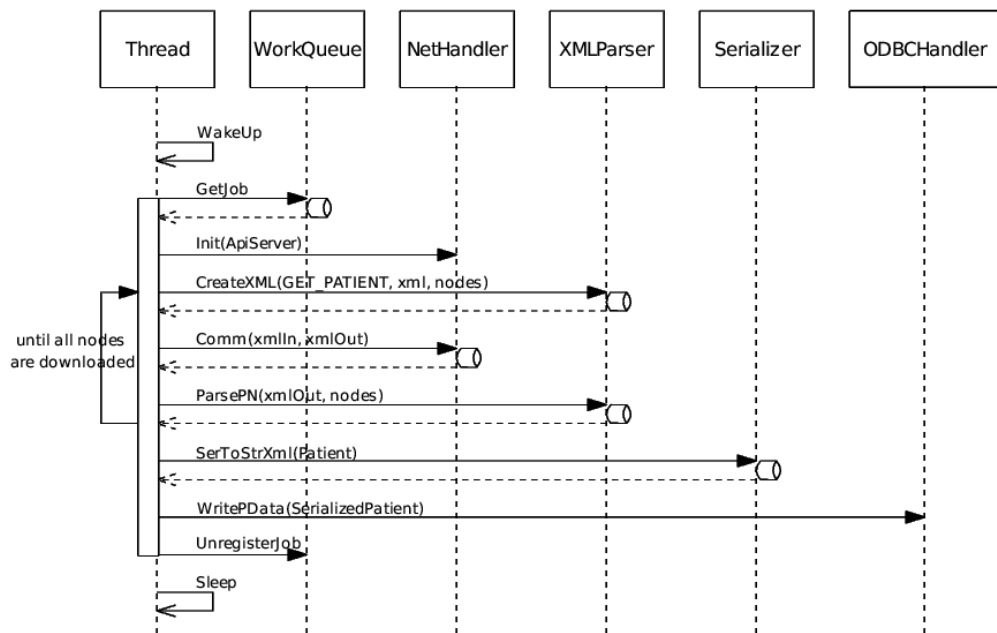
- **meno a priezvisko** - prihlasovacie údaje rovnaké ako v MUDRCovi
- **heslo** - prístupové heslo na stránky
- **jazyk** - pre zobrazenie stránky a dát pacienta v rodnom jazyku užívateľa
- **blokovanie** - konto môže byť aktívne alebo zablokované

- **uzly pacienta** - tie sa užívateľovi budú implicitne zobrazovať

Všetky položky sú nastaviteľné v závislosti na ich logickej príslušnosti (administrátor môže pridávať či mazať kontá alebo ich zablokovať; lekár si môže nastavovať heslo, jazyk a uzly, ktoré chce aby sa mu implicitne zobrazovali).



Obr. 4.1: Class diagram proxy servera



Obr. 4.2: Priebeh sťahovania pacientových dát

Kapitola 5

Prínos a porovnanie projektu

5.1 Zhodnotenie vývoja aplikácie

Počas návrhu a vývoja sa ukázalo, že aplikácia MUDR Proxy je komplexný projekt, ktorý sa skladá z viaceru netriviálnych častí. Pre dosiahnutie aspoň základnej funkčnosti, čo sa týka proxy serveru (komunikácie s MUDR, parsovanie XML, návrh a implementácia dátových štruktúr), CGI skriptu (komunikácia s proxy) a webovej stránky (PHP kostra, spolupráca s XML a XSL, komunikácia s databázou), bolo potrebné značné množstvo času (August - September 2005, Február - August 2006). Počas tohto času bolo potrebné nielen funkčne a v dostatočnej miere implementovať požadované funkcie, ale aj zoznamovať sa s technológiami potrebnými na tieto úlohy, ako napr. CORBA, knižnice wxWidgets, Xerces-C++, Boost a takisto s programami pre použitie v praxi, ako napr. konfigurácia HTTP servera či databázy. Počas vývoja sa muselo upustiť od niektorých "vedľajších požiadaviek" (ako napr. komunikácia s MUDRCom), aby sa v dostatočnej miere implementovali tie hlavné. Nezanedbateľné množstvo času sa muselo použiť na spustenie do pilotnej prevádzky a testovanie chýb, keďže je to aplikácia skladajúca sa z viaceru častí, ktoré k svojmu chodu potrebujú správne nastavený systém (web server, databázu, knižnice, atď). Výsledný projekt pokrýva iba najhlavnejšie požiadavky, ktoré ale splňuje v dostatočnej miere (stiahnutie pacienta z MUDR serverov a poskytnutie ho užívateľom na rôznych tenkých klientoch). Počas vývoja sa ukazovalo obrovské množstvo možností zlepšenia celej aplikácie, či už funkčnosti proxy alebo webovej stránky (viac rozpísané v nasledujúcej kapitole), alebo z kvalitovania stávajúcich funkcií, či zlepšenie robustnosti a univerzálnosti kódu.

5.2 Porovnanie

Pri porovnaní s ostatnými dostupnými projektami zaoberajúcimi sa elektronickým zdravotným záznamom autor dospel k záveru, že žiadny z komerčných alebo nekomerčných českých a slovenských produktov nepodporuje takúto formu integrácie či prezentácie dát. Väčšina z nich je založená na bázi komplexného nemocničného systému, čím sa vlastne projekt MUDR/MUDR Proxy stáva podmnožinou týchto systémov. Na druhej strane žiadny z nich neimplementuje takýto rozsiahly model uchovania elektronického záznamu pacienta v databázach podobne ako MUDR a u žiadnom z nich sa nevyskytol podobný systém integrácie týchto databáz a tým pádom vytvorenie distribuovaného elektronického zdravotného záznamu. V celosvetovom meradle ale nie je projekt MUDR Proxy jediný, ktorý sa pokúša o túto činnosť, ako je spomenuté v časti 2.1. V momentálnej fáze síce projekt MUDR Proxy neintegruje podporu medzinárodných komunikačných štandardov (čo nebude problém vyriešiť), ale ako jeden z mála však prináša riešenie dostupnosti pacientových dát prostredníctvom veľkého množstva užívateľských klientov. Jeho rozšírením o tieto štandardy a zdokonalením by sa mohol stať konkurencieschopným aspoň v česko-slovenskom meradle a sľubným pre budúce nasadenie do zdravotných systémov.

5.3 Prínos

Čistý prínos projektu by sa dal zhodnotiť nasledujúco:

- Aplikácia je dobrým základom pre distribuovaný model elektronického záznamu pacienta projektu MUDR
- Umožňuje prístup ku kompletným dátam pacienta rýchlo so základnými možnosťami prispôsobenia a nastavenia
- Podporuje prezeranie web stránky (a teda prezentáciu pacientových dát) na rôznych užívateľských zariadeniach, čo značne zvyšuje možnosti využitia, zároveň je navrhnutá spôsobom jednoduchého rozšírenia pre ďalších tenkých klientov
- Je navrhnutá modulárnym, ľahko modifikovateľným systémom, ktorý umožňuje využitie a ďalšie rozšírenie aplikácie v rôznych funkčných smeroch (proxy aj web stránky)

- Podporuje platformy Windows a Unix/Linux, pričom používa open-source knižnice, ktoré jej umožňujú dostatočnú flexibilitu pre použitie v nekomerčných a komerčných sférach

Nezanedbateľným prínosom je zrušenie potreby aplikovania logiky s narábaním pacientových dát, ako je napr. u MUDRCa, ktorý si celú komunikáciu a pacientove dáta riadi sám. Pri využití webových stránok MUDR Proxy netreba tenkého klienta žiadnym spôsobom modifikovať, alebo pridávať špecializované súčasti, všetko je obsluhovateľné cez webovú stránku. Každému klientovi sa dá prispôbiť úroveň prezentácie, a to použitím / vynechaním technológií ako napr. javascript, ajax (zamýšľané do budúcnosti) či možnosť prezentácie v grafickom / textovom prostredí.

Pre porovnanie uvádzam komunikačnú náročnosť pri sťahovaní znalostnej bázy a dát priemerného pacienta. TAB

Kapitola 6

Budúcnosť projektu

Projekt MUDR Proxy obsahuje veľké množstvo potenciálnych vylepšení a rozšírení funkčnosti, ktoré sa naskytli počas vývoja pilotnej aplikácie pri konzultáciách s Miroslavom Nagyom. Pravdepodobné je aj pokračovanie projektu na pôde EuroMISE centra alebo v rámci diplomovej práce. Návrhy na zlepšenie či odstránenie stávajúcich problémov aplikácie sú diskutované v nasledujúcich sekciách.

6.1 Proxy server

- Komunikáciu s pilotným MUDR serverom prostredníctvom HTTP protokolu zabezpečuje web server Apache, je nutné ju otestovať aj s inými riešeniami HTTP serverov a zlepšiť výkonové vlastnosti.
- Základom projektu je knižnica wxWidgets. Tá má množstvo rozšírení, ktoré by sa dali do budúcna využiť (ako napr. automatická HTTP komunikácia, XML parser). Táto knižnica sa však rýchlo vyvíja a je otáznne, či bude pripravovaná úplne nová verzia (3.0) kompatibilná s použitou (2.6).
- Zlepšenie komunikácie medzi proxy servermi, napr. automatické predávanie MUDR či ostatných proxy serverov, znalostnej bázy alebo pacientových dát (spolu s informáciou z ktorých MUDR serverov boli posťahované, plus časová známka a iné).
- Podpora pre komunikáciu s MUDRCom (momentálne uvažovaná iba read-only), čo by však znamenalo značné začlenenie logiky pre komu-

nikáciu pomocou MUDR XML schémy, prepočítavanie id uzlov a iné závažné problémy.

- Zavedenie dôkladnejšej autentifikácie, v tejto verzii sa používa na identifikáciu s MUDR servermi testovací účet.
- Podporu pre medzinárodné komunikačné štandardy, napr. HL7.
- Zmenu v komunikačnom protokole s MUDR (hlavne doimplementovanie podpory pre časové razítka u pacientových dát).

6.2 CGI skript

CGI skript ako jednoduchá aplikácia nemá veľa dôvodov na zmeny, ak áno, je vitálne implementovať iba tie, ktoré nespôsobia značné spomalenie chodu programu.

- Analýza možnosti presunutia časti logiky spracovávania pacientových dát z proxy alebo web stránky, ako napr. export požadovaných uzlov pacienta.
- Rozšírenie administrátorskej funkčnosti proxy.

6.3 Webová stránka

Webová stránka má v momentálnej verzii MUDR Proxy najväčšie možnosti jednoduchého rozšírenia.

- Riešenie autentifikácie, aby mal lekár prístup iba k tým MUDR serverom, na ktorý má účet a na nich iba k tým dátam, ktoré má právo prezeráť, či editovať.
- Rozšírenie možností zobrazovania pacientových dát s ohľadom na dizajn, časové razítka (timestamps) alebo rozsah hodnôt.
- Pridanie viacerých užívateľských nastavení, vrátane prispôbenia dizajnu stránky.
- Rozširovanie podporovaných tenkých klientov.
- Širšie možnosti administrácie MUDR alebo MUDR Proxy serverov a užívateľov.

Kapitola 7

Záver

S výsledkami projektu je autor spokojný, pretože sa v uspokojivej miere podarilo dosiahnuť požadovanú funkčnosť - integrovať pacientove dáta z viacerých MUDR serverov a poskytnúť ich tenkým užívateľským klientom v prehľadnej a prívetivej forme prispôbenej danému klientovi. Zároveň je projekt dobre pripravený na ďalšie rozširovanie a pokračovanie vo vývoji.

Dodatok A

Použité knižnice

Počas vývoja aplikácie bolo potrebné a zároveň výhodné použiť špecializované knižnice. Podmienkou pre použitie bola podpora Windows a Unix, v neposlednom rade kvalita a tiež licencia, ktorá by mi dovoľovala použiť ju v tomto projekte ako bakalárskej práci, a takisto aby nekolidovala s licenciou knižnice wxWidgets, na ktorej je postavená celá proxy.

wxWidgets - multiplatformová knižnica, hlavne pre tvorbu programov s grafickým prostredím v rôznych programovacích jazykoch. Okrem veľkej sady nástrojov na tvorbu grafiky sa vyznačuje veľmi dobrou implementáciou rôznych dôležitých častí aplikácie od vlákien, udalostí a sieťovej komunikácie až po parsovanie príkazového riadka. Takisto zabezpečuje prenositeľnosť kódu medzi platformami a rozhranie pre komunikáciu s databázami, či automatické a bezstarostné programovanie s/bez Unicode podpory.

Zdroj: <http://wxwidgets.org/>

Licencia: wxWindows Licence

(<http://wxwidgets.org/about/newlicen.htm>)

Xerces-C++ Kvôli komunikácii s MUDR API servermi vo forme XML je potrebný kvalitný XML parser. Jeden z najlepších voľne dostupných sa ukázal Xerces-C++ parser od Apache, hlavne kvôli jeho robustnosti a výbornej použiteľnosti.

Zdroj: <http://xml.apache.org/xerces-c/>

Licencia: Apache Software License, Version 2.0

(<http://www.apache.org/licenses/LICENSE-2.0.html>)

log4cplus Logovací nástroj, široká škála nastavení (podpora externého kon-

figuračného súboru), asi jediný dobre použiteľný pre C++.

Zdroj: <http://log4cplus.sourceforge.net/>

Licencia: Apache Software License, Version 1.1

(<http://www.apache.org/licenses/LICENSE-1.1>)

boost Kvalitná C++ knižnica plná rôznych uľahčení pre programátorov.

V projekte je použitá pre serializáciu STL kontajnerov typu map a vector do XML.

Zdroj: <http://www.boost.org/>

Licencia: Boost Software License

(http://www.boost.org/more/license_info.html)

patch pre wx Keďže wxWidgets majú tú nevýhodu, že sú úzko späté s grafickým prostredím (ale zlepšuje sa to), v základnej zostave konzolovej aplikácie sa nenachádzajú ani také dôležité veci ako event systém či main loop. Tento patch to napráva.

Zdroj: Lukasz Michalski

(https://sourceforge.net/tracker/?func=detail&atid=309863&aid=1113088&group_id=1113088)

Mico Jednoduchá distribúcia CORBY postačujúca pre projekt.

Zdroj: <http://www.mico.org/>

Licencia: GNU

(<http://www.gnu.org/copyleft/>)

iconv Knižnica na prekonvertovanie textu medzi množstvom znakových kódov.

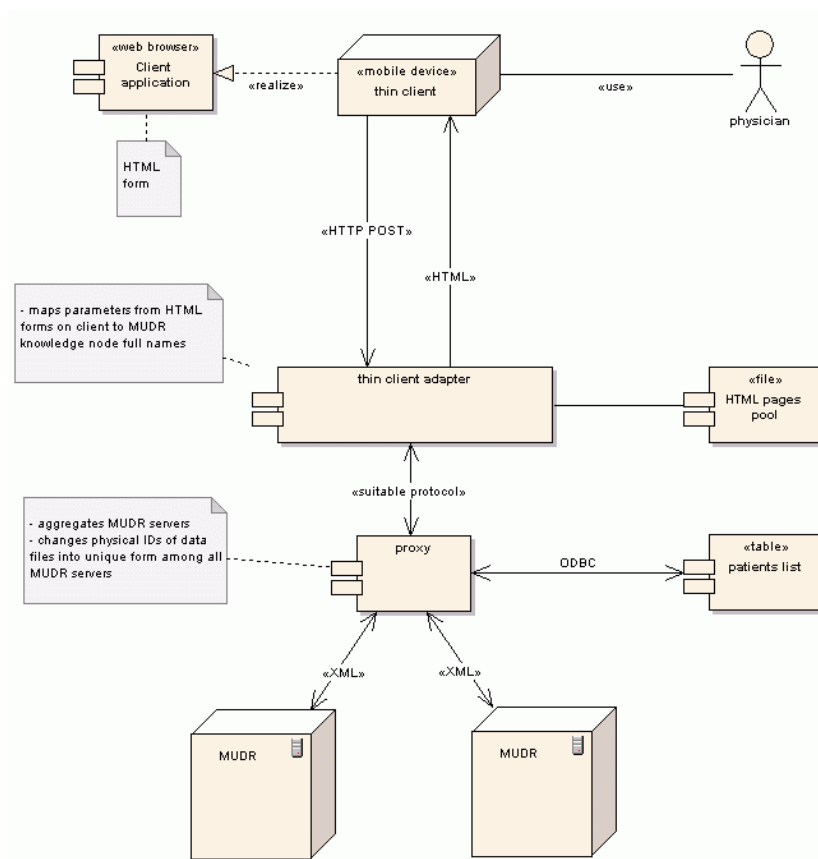
Zdroj: <http://www.gnu.org/software/libiconv/>

Licencia: GNU

(<http://www.gnu.org/copyleft/>)

Dodatok B

Schéma prvotného návrhu



Obr. B.1: Prvotný návrh

Literatúra

- [1] Josef Špidlen: *Databazová reprezentace medicínských informací a lékařských doporučení*, dipl. práce, KSI MFF UK, 2002.
- [2] *The SynEx Project*
<http://www.chime.ucl.ac.uk/work-areas/ehrs/SynEx/>
- [3] *I4C-TripleC - Integration and Communication for the Continuity of Cardiac Care*
<http://www.euromise.cz/new/i4c-triplec.php>
- [4] *The Good European Health Record*
<http://www.chime.ucl.ac.uk/work-areas/ehrs/GEHR/>
- [5] *ISO/TC215 Health Informatics*
<http://www.iso.org/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeDetailP>
- [6] *CEN/TC251*
<http://www.centc251.org/>
- [7] *International Classification of Diseases*
<http://www.who.int/classifications/icd/en/>
- [8] *Health Level 7*
<http://en.wikipedia.org/wiki/HL7>
- [9] *DASTA*
<http://www.mzcr.cz/index.php?clanek=482>
- [10] *MEDEA*
http://www.stapro.cz/pr_medea_cz.asp
- [11] *Medical Process Assistant*
<http://www.i.cz/zdravotnictvi/mpa.html>

- [12] *Nemocniční informační systém HERMES*
http://www.logis.cz/hermes_info.htm
- [13] *Komplexný nemocničný informačný systém*
http://www.lcs.sk/charakter_knis.php
- [14] Erik Borš, František Brantál, Lukáš Brožovský, Miroslav Nagy, Miroslav Vranka: *Multimedia Distributed Record Client - Programátorská dokumentácia*
- [15] *wxWidgets Online Manual*
<http://www.wxwidgets.org/>
- [16] *log4cplus API Documentation*
<http://log4cplus.sourceforge.net/docs/html/index.html>
- [17] *Apache HTTP Server Project*
<http://httpd.apache.org/>
- [18] *Xerces-C++ Documentation*
<http://xml.apache.org/xerces-c/apiDocs/index.html>
- [19] *libiconv documentation*
<http://www.gnu.org/software/libiconv/>
- [20] *Mico*
<http://www.mico.org>
- [21] *Thread pool pattern*
http://en.wikipedia.org/wiki/Thread_pool_pattern
- [22] *Thread pools and work queues*
<http://www-128.ibm.com/developerworks/java/library/j-jtp0730.html>
- [23] *boost documentation*
<http://www.boost.org/libs/libraries.htm>
- [24] *MySQL 5.0 Reference Manual*
<http://dev.mysql.com/doc/refman/5.0/en/>
- [25] *PHP Manual*
<http://www.php.net/manual/en/>

- [26] *XML Tutorial*
<http://www.w3schools.com/xml/>
- [27] *XSL Tutorial*
<http://www.w3schools.com/xsl/>