VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

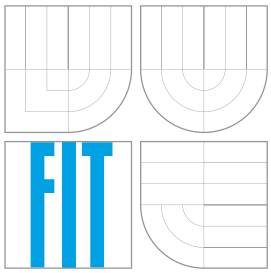
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV SENDLER

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

DATAMINING FROM JABBERU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV SENDLER
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOZEF MLÍCH
SUPERVISOR

BRNO 2011

Abstrakt

Předmětem této bakalářské práce bylo seznámení se s problematikou komunikace přes Jabber síť, která zde byla rozebrána. Konkrétním cílem bylo vytvoření jednoduchého Jabberového klienta, který by byl schopen získávat statistická data. Nashromážděná data sloužila pro pozdější analýzu a grafickou reprezentaci informací z nich získaných.

Abstract

The objective of this thesis was acquaint oneself with problems of communication via Jabber network, which was also analyzed. The specific objective was to create a simple Jabber's client which would be able to obtain statistical data. The collected data was used for analysis and graphic representation of information.

Klíčová slova

Jabber, XMPP, robot, datamining, dolování dat.

Keywords

Jabber, XMPP, robot, datamining.

Citace

Jaroslav Sendler: Datamining z jabberu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Datamining z jabberu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Sendler
17. dubna 2011

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce Ing. Jozefovi Mlíchovi za ochotu a kladný přístup při konzultacích. Dále za poskytnutí hardware na němž běžel program a sbíral data.

Obsah

Kapitola 1

Úvod

Dnes mezi velmi se rozšiřující technologie na poli síťo

Kapitola 2

XMPP

V následující kapitole jsou, pro usnadnění a jednodušší pochopení, rozebrány základní stavební kameny protokolu Extensible Messaging and Presence Protocol (XMPP). Konkrétně jsou zde popsány stávající vlastnosti implementace, architektura protokolu XMPP obecně [17, 18] a další detaily protokolu [1, 19, 12]. Ty se vztahují k požadavkům na data mining popisovaný v této práci [16, 11]. Další informace použité pro popis a pochopení XML jazyka byly čerpány z [9, 8].

Vznik samotného protokolu XMPP je datován do roku 2004 (březen), kdy na něj byl přejmenován Jabber. Původní projekt Jabber byl vytvořen roku 1998 autorem Jeremie Miller, který ho založil za účelem vytvořit svobodnou otevřenou IM službu. Uvedený projekt měl obsahovat tři základní vlastnosti, do kterých se zahrnují jednoduchost a srozumitelnost pro implementaci, jednoduchost v oblasti šíření a otevřenost podobě veřejně dostupného popisu samotného protokolu. Základní vlastnosti a výhody klientů a serverů budou podrobněji popsány níže. Roku 1999, 4.ledna byl vytvořen první server se jménem Jabber. Komunita vývojářů se chopila iniciativy a vytvořila klienty, kteří dokázali se serverem komunikovat, pro různé platformy (Linux, Macintosh, Windows). Roku 2004 byl protokol XMPP přidán mezi RFC (request of comments - žádost o komentáře) dokumenty. Základní norma popisující obecnou strukturu protokolu je RFC 3920 [17] a RFC 3921 [18], který se zaměřuje na samotný instant messaging a zobrazení stavu. Další zdokumentovaná rozšíření jsou vydávána v podobě tzv. XEP (XMPP Extension Protocol) dokumentů, které jsou známé také pod starším názvem JEP (Jabber Enhancement Proposal). Dnešní počet těchto norem se blíží k číslu 300. Každý XEP obsahuje status a stav vývoje (schválení), ve kterém se zrovna nachází.

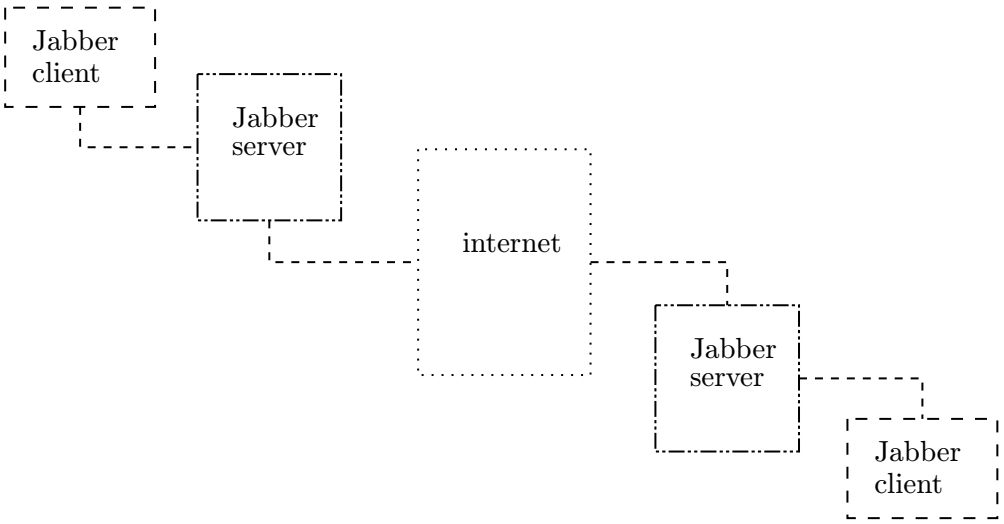
Jako bezpečnostní prvky jsou zde podporovány SASL, TLS a GPG. XMPP protokol je postaven na obecném značkovacím jazyce XML, proto vlastnosti popsané v kapitole 2.2 platí i pro tento protokol.

2.1 Architektura

Dobře navržená architektura tvoří základ pro správně fungující internetovou technologii [1, 19]. XMPP protokol využívá decentralizované klient–server složení. Tato struktura se nejvíce podobá struktuře posílání e–mailů. V tomto případě je decentralizace sítě chápána jako inteligentní nezávislost mezi vývojáři klientů a serverů. Každý z nich má možnost zaměřit se na důležité části svého vývoje. V případě server se jedná o spolehlivost a rozšiřitelnost a u klient na uživatele. Každý server pracuje samostatně, což znamená, že chod

ani výpadek jiné datové stanice žádným způsobem jeho běh neovlivní. V případě výpadku jiného serveru bude nedostupný pouze seznam kontaktů a služeb, které registrovaným uživatelům poskytoval.

Obrázek 2.1 znázorňující distribuovanou architekturu Jabberu byl převzat z [1]. Komunikace dvou Jabber klientů probíhá za účasti jejich serverů a sítě, která je spojuje. Spojení mezi nimi bývá často šifrováno.



Obrázek 2.1: Distribuovaná architektura Jabber.

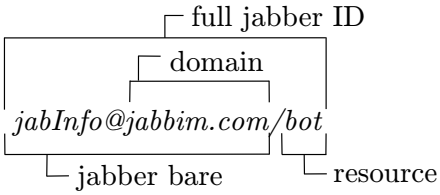
Architektura Jabber serverů využívá velké množství mezi–doménových připojení podobně jako internetový systém elektronické pošty. Komunikace klienta z jedné domény s klientem z jiné na rozdíl od e–mailového modelu nevyžaduje spolupráci třetích stran. Klient se spojí s „domácím“ serverem, který přímo naváže spojení se servem požadováného klienta. Tyto vlastnosti jsou zárukou pro bezpečný přenos zpráv, znemožňující „krádeže“ JID[1], který je popsán níže, a spamování.

Jabber ID

Jabber ID (JID) je jednoznačný virtuální identifikátor uživatele na síti. V případě založení účtu nejsou rozlišována velká a malá písmena, což znamená, že Jabber není case–sensitive. Jednoznačný Jabber identifikátor je složen ze dvou částí *Jabber bare* nebo–li čisté ID a *resource* [17]. Základní část na první pohled připomíná e–mailovou adresu *user@server*. Druhá část slouží k přesné identifikaci jednotlivých spojení. Je použita ke směrování trafiku s uživateli v případě otevření většího množství spojení pod jedním uživatelem. Společně Jabber bare a resource tvoří tzv. *full JID — user@server/resource* například *jabInfo@jabbim.cz/bot*. Jednotlivé části části uživatelského jména popsané v tomto odstavci jsou ukázány v obrázku 2.2.

Další vymoženost JabberID oproti e–mailové adrese je jeho možnost používat prakticky libovolné národní znaky u doménových jmen a uživatelských účtů [19]. Využíváním kódování UNICODE, se XMPP stává plně mezinárodní a není jako jiné protokoly omezen rozsahem ASCII tabulky. Přestože je tato vymoženost k dispozici, doposud není žádným výrazným způsobem využívána.

[1] uživatelské jméno



Obrázek 2.2: Rozebraná struktura Jabber ID.

Klient

Klient je především plně ovládatelný grafický program podporující jednoduché odesíláni zpráv. V této práci je však zastoupen robotem s konzolovým rozhraním. XMPP svou architekturou nutí, aby byl co nejjednodušší. Vlastnosti, které by měl mít jsou shrnuty do tří bodů [12]:

1. komunikace se serverem pomocí TCP soketu
2. rozparsování a následná interpretace příchozí XML zprávy „stanza“ (kapitola 2.3)
3. porozumění sadě zpráv z Jabber jádra

Server

K hlavním charakteristikám serveru oproti klientovi, jehož základní vlastností byla jednoduchost, patří stabilita a bezpečnost. Standardně běží na TCP portu 5222 [12]. Komunikace mezi servery je realizována přes port 5269. Každý server uchovává seznam zaregistrovaných uživatelů, který nevykazuje žádný jiný server. Zaregistrovaní uživatelé v daném seznamu se mohou do sítě připojovat pouze přes něj. To zajišťuje nemožnost „krádeže“ účtu. Protože XMPP komunikace probíhá přes síť, musí mít každá entita adresu, v tomto případě nazvána JabberID. XMPP spoléhá na DNS což znamená že používá jména na rozdíl od IP protokolu.

Server Jabber je považován za „deamona“, který spravuje tok dat mezi jednotlivými komponentami, které společně tvoří Jabber služby [12]. Například *Jabber Session Manager* (JSM) poskytne funkce pro IM komunikaci a práci se seznamem kontaktů. Komunikace mezi jednotlivými servery je zprostředkována za pomocí komponenty *S2S* (server to server). Při připojení klienta k serveru je komunikace řízená pomocí *C2S* (client to server). Jak již bylo řečeno Jabber síť využívá doménová jména místo špatně zapamatovatelných IP adres. Pro tento způsob identifikace je určena služba *dnsrv*, která názvy překládá.

V tabulce 2.1 jsou shrnuty informace o serverech Jabberu. První sloupec tvoří jméno, následuje programovací jazyk v němž je napsán. Většina aplikací pro servery je vydávána pod licencí GPL[2]. U všech software byla zkoumána nejaktuálnější verze. Její číslo lze naleznout ve třetím sloupci. Všechny servery lze provozovat na operačním systému Linux a Windows. Na platformě Mac OS mohou být použity všechny zde jmenované vyjma jabberd2. Pět z šesti zde představených software pro server Jabber jsou ve stále vyvíjeny, tedy kromě jabberd14. Hlavním účelem tabulky je prezentovat důležité vlastnosti serverů v oblasti podpory rozšířených statusů. Jedná se o standardy *pubsub*[3] (XEP–0060) [11] a o jeho verzi

[2] General Public License-všeobecná veřejná licence GNU
[3] Publish–Subscribe

zaměřenější více na uživatele *pep*[4] (XEP–0163) [16]. Obě tato rozšíření tvoří nezbytnou základnu pro *rozšířené statusy* a proto je jejich podpora jak u serverů tak klientů vyžadována. Podrobněji toto téma bude rozebráno v podkapitole 2.4.

Server	Jazyk	Verze	XEP–0060	XEP–0163
ejabberd	Erlang/ Top	2.1.6	ANO	ANO
Openfire	java	3.6.4	ANO	ANO
jabbred2	c	2.2.11	NE	NE
jabbred14	c, c++	1.6.1.1	ANO	NE
Prosody	lua	0.7.0	NE	ANO
Tigase	java	5.0.0	ANO	ANO

Tabulka 2.1: Přehled Jabber serverů.

Z výše uvedené tabulky je zřejmé, že aplikace pro servery, které jsou stále ve vývoji, podporují tzv. *rozšířené statusy*. Tedy kromě programu jabbred2.

2.2 XML

Jazyk XML (eXtensible Markup Language) [8], metajazyk pro deklaraci strukturovaných dat, je jádrem protokolu XMPP. Samotný jazyk vznikl rozšířením metajazyka SGML, jež slouží na deklaraci různých typů dokumentů. Základní vlastností je jednoduchá definice vlastních značek (tagů). Dokument XML se skládá z elementů, jež můžeme navzájem zanořovat. Vyznačujeme je pomocí značek - počáteční a ukončovací. Pomocí tohoto jazyka je tvořena stanza popsaná v kapitole 2.3.

Ukázka možné struktury dokumentu psaného jazykem XML je zobrazena na příkladu 2.1. Standardně je předpokládáno, že je psán v kódování UTF-8 [9], ale je–li jako v tomto případě použito jiné musí být konkrétní kódování uvedeno na jeho počátku. V opačném případě nemusí být obsah správně zobrazen. Na začátku dokumentu se také uvádí verze XML, ve které je dokument psán (1. řádek obrázku). Následuje kořenový element, jež je uzavřen na samotném konci dokumentu. 4. řádek prezentuje možnost použití prázdného elementu, který obsahuje jeden atribut s názvem zkratky fakulty. Velký význam zde mají úhlové závorky. Jsou jimi z obou stran obaleny všechny elementy.

```
1  <?xml version="1.0" encoding="iso-8859-2"?>
2  <fakulta>
3    <název>Fakulta informačních technologií</název>
4    <zkratka fakulty="FIT" />
5    <typy studia>
6      <bakalářské titul="Bc."></bakalářské>
7      <magisterské></magisterské>
8      <doktorské></doktorské>
9    </typy studia>
10 </fakulta>
```

Příklad 2.1: Ukázka základního XML dokumentu.

[4] Personal Eventing Protocol

2.3 Stanza

Základní jednotkou pro komunikaci založenou na XML je stanza. Z jednoduššího pohledu je možné se na ni dívat jako na jeden dlouhý XML soubor. Při zahájení komunikace se tento soubor „otevře“. Jeho samotné uzavření probíhá až při odhlášení od sítě, nebo–li přepnutí klienta do stavu offline. Stanzu je tedy možné vnímat jako stream, který obsahuje všechna data probíhající komunikace. Mezi elementy používané pro komunikaci klienta se serverem patří tyto tři: *message*, *presence* a *iq*. Každý zde uvedený člen má svůj jednoznačný význam. V následujících odstavcích jsou jednotlivé části stanzy blíže definovány a na reprezentativních příkladech jsou ukázány jejich základní struktury a možnosti využití v praxi.

První prvek o kterém se zmíníme je označen anglickým výrazem pro slovo zpráva–*message*. Jak již název napovídá slouží k posílání zpráv všeho druhu. Je to základní metoda pro rychlý přenos informací z místa na místo. Zprávy jsou typu „push“, což znamená že jsou odeslány a není očekávána žádná aktivita od příjemce, která by přijetí potvrdila. Jedno z možných využití se nachází v klasické komunikace po internetu, tzv. instant messaging (IM). K dalším možným použitím patří skupinový chat a oznamovací nebo upozorňující zprávy. Každá z těchto zpráv je tvořena z minimální povinné struktury. Tak jako u klasické poštovní korespondence nesmí chybět adresa odesílatele a adresa příjemce, kterému je zpráva adresována. Podle možnosti použití jsou zprávy děleny do kategorií. Jmenovitě toto rozdělení implementuje atribut *type*, který může nabývat jednu ze čtyř hodnot. Jsou rozlišovány zprávy pro komunikaci mezi dvěma entitami, skupinový chat, upozornění, chybová zpráva a v neposlední řadě zpráva bez kontextu vyžadující odpověď příjemce. Nakonec nesmí být opomenut blok zprávy, pro uživatele IM nejdůležitější, nesoucí vlastní obsah.

Základní použití struktury elementu *message* je prezentováno na příkladu 2.2. Na prvním řádku je uveden atribut, značící odesílatele. Druhý řádek opět obsahuje klientův JID, který zprávy přijímá. Následuje informace o typu zprávy a poté je uveden element *body* nesoucí samotný obsah.

```
1  <message from="uzivatel@jabbim.com"
2           to="jabinfo@jabbim.com/bot"
3           type="chat"
4     <body> Kolik je hodin? </body>
5  </message>
```

Příklad 2.2: Použití elementu *message*.

Další částí stanzy je poskytována struktura pro *request–response* (žádost–odpověď) vazbu a wokrflows, podobný metodám GET, POST a PUT z protokolu HTTP [19]. Zkráceně je označována pomocí dvou počátečních písmen *Info/Query* nebo–li IQ. Na rozdíl od elementu *message* tvoří *iq* spolehlivější přenos, optimalizovaný pro výměnu dat (binární data). K dalším rozdílům patří povinnost příjemce odpovědět na každou přijatou zprávu, nebo–li potvrdit její doručení. Skutečnost, že je na právě požadovanou zprávu odpovězeno, zajišťuje parametr *id*. Iq dotaz nebo odpověď musí obsahovat stejnou hodnotu tohoto atributu jako zpráva vytvořená žádajícím subjektem. Další povinný atribut rozděluje iq na čtyři typy. Jednotlivé žádosti na proces nebo akci jsou posílány samostatně [18]. V příloze E je uvedena rozsáhlejší struktura tohoto elementu.

Obrázek 2.3 znázorňuje základní použití elementu *iq*. Uživatel *uzivatel* posílá dotaz na získání kontakt listu (řádek 5.).

```
1  <iq from="uzivatel@jabbim.com/doma"
2      to="uzivatel@jabbim.com"
3      id="uhhfw23648"
4      type="get"
5     <query xmlns="jabber:iq:roster"/>
6  </iq>
```

Příklad 2.3: Použití elementu *iq*.

Celá struktura elementu *iq* je zobrazena v příloze E. Použití nachází v případech, které nastavují, žádají nebo informace posílají. Tato struktura je využívána pro novou registraci, posílání kontakt listu a další.

Poslední a pro tuto práci nejdůležitější prvek stanzy je *presence*. V případě, že nemá určeného příjemce, tak funguje způsobem jako broadcast. Což znamená směrování informací všem klientům, kteří jsou zaregistrování k jejímu odběru. Presence nebo–li volně přeloženo do češtiny informace o stavu (přítomnost) rozesílá dostupnost ostatních entit v síti. Jedná se tedy o nastvaení uživatelské dostupnosti taj jako na jiných real–time komunikačních a sociálních systémů.

Existuje několik základních stavů statusů, které reprezentují aktuální dosažitelnost uživatele. Tento jev je vyjádřen pomocí elementu *show*, který disponuje čtyřmi možnostmi. První oznamuje, že je uživatel k dispozici a schopen aktivní komunikace. Druhá často se vyskytující možnost naznačuje, že je subjekt krátkou dobu pryč od svého IM klienta. Tento a další dva stavy, popsané dále, jsou často změněny bez lidského zásahu (pomocí pc nebo jiného zařízení) prostřednictvím funkce známé jako „auto–away“. Poslední dva stavy charakterizují delší časové období nečinnosti. Tato oznámení o změně stavu uživatele jsou často zasílána pouze kontaktům, které se nacházejí v režimu online. Tato optimalizace přispívá ke snížení síťového provozu, jelikož presence v reálném čase při komunikaci využívá velké množství šířky pásma.

Základní použití *presence* je zobrazeno v příkladu 2.4. Kontakt *jabinfo@jabbim.com/bot* (1.řádek) posílá informace o svém stavu (řádek č.2) a svůj status (č.3).

```
1  <presence from="jabinfo@jabbim.com/bot"
2     <show> online </show>
3     <status> Jsme zde. </status>
4  </presence>
```

Příklad 2.4: Použití elementu *presence*.

Obsáhlejší struktura elementu *presence* je zobrazena v příloze E, kde je rovněž k nalezení přehled všech možných stavů.

Jak již bylo zmíněno v části o Jabber ID Jabber podporuje práci s více současně připojenými klienty k jednomu Jabber účtu. Vysvětlení funkčnosti bude prezentováno na příkladu uživatele přihlášeného na stolním počítači a z klienta v mobilním telefonu. U obou těchto připojení je použit stejný Jabber bare, ale odlišného resource, například *domov* a *mobile*. Právě tento rozdíl v tzv. „full“ adrese účtu zajišťuje jednu ze dvou možných podmínek pro správnou adresaci zpráv. Druhá možnost, která bude uplatněna při použití adresy účty pouze ve formě Jabber bare, je nastavení priority u jednotlivých programů. Priorita je číslo v rozsahu hodnot od -128 do 127, kde klient s větší prioritou má přednost před klientem s nižší. Nastane–li případ připojení více klientů se stejnou prioritou, každý server se při roze-

sílání zpráv zachová podle vlastní implementace. Některé rozešlou zprávy všem klientům, jiné naopak jen poslednímu přihlášenému.

2.4 Rozšíření

Dále se tato práce zabývá rozšířeními protokolu XMPP o další vlastnosti k jejichž popisu slouží XEP. Pro tuto práci jsou nepostradatelné „statusy“, které popisují standardy XEP–0060 [11] a XEP–0163 [16] zkráceně PEP [5]. Obě tato rozšíření umožňují strukturovaně pracovat, používat a přenášet další XEP protokoly. Jako příklady relevantní k práci jsou zde uvedeny protokoly *User Location* (kde se uživatel právě nachází) [5], User Tune (co uživatel poslouchá za hudbu) [14], User Mood () [15] a User Activity () [10]. Jsou to tedy protokoly založené na PEP, které vyžadují podporu nejen v klientech, ale i na straně serveru (zobrazuje tabulka 2.1). S touto informací úzce souvisí další protokol XEP–0115 [6], který umožňuje zjistit podporované schopnosti klienta, případně které informace je ochoten přijímat viz 2.4.

Všechna tato rozšíření by mohla být přidána přímo do statusu viz 2.4, avšak ten je primárně určen k informování o přítomnosti na IM síti. Hlavní rozdíl mezi PEP a obyčejným posílání stavu pomocí presence je v pravomoci klienta přijmout nebo odmítnou informaci, na rozdíl od presence, jež je přijata vždy.

Základ přenosu informací začíná na straně klienta, který chce všechny ve svém roster (kontakt) listu informovat o statusu. Zašle zprávu obalenou v elementu *iq* serveru. Ukázka této zprávy je prezentována na příkladu 2.5, který znázorňuje zaslání informaci o druhu hudby, kterou v danou chvíli poslouchá. Využívá k tomu rozšíření User Tune, definovaném na řádku číslo 5. Základ zprávy oznamující začátek vysílání informací o rozšířených statusech je vždy stejný. Liší se pouze řádkem 3. a obsahem elementu item v obrázku 2.5.

```
1   <iq from='uzivatel@jabbim.com' type='set' id='pub1'>
2     <pubsub xmlns='http://jabber.org/protocol/pubsub'>
3       <publish node='http://jabber.org/protocol/tune'>
4         <item>
5           <tune xmlns='http://jabber.org/protocol/tune'>
6             <artist>Daniel Landa</artist>
7             <length>255</length>
8       ...
```

Příklad 2.5: Začátku vysílání rozšířeného statusu.

V případě úspěšného přijetí *iq* zprávy serverem, každý, kdo se zaregistroval k odebírání rozšířené statusy, obdrží oznámení ve formě *message*. Oznámení bude také doručeno všem resources. Celá zpráva i všechny další náležitosti jsou uvedeny v příloze E.4.

XEP-0115: Schopnosti subjektu

Pomocí protokolu Entity Capabilities [6] je možné zjistit vlastnosti klienta. Tato informace výrazně snižuje počet a velikost komunikací a přenosů zpráv mezi uživateli. Dotazem zobrazeným na příkladu 2.6 je zjištěna schopnost jednotlivých klientů, kterou následně server využije pro správné směrování rozšířených statusů. Všechny zde zmiňované rozšíření a protokoly z kapitoly 2.4 je možné u každého klienta (seznam klientů obsahuje tabulka v

[5] Personal Eventing via Pubsub

příloze E.4) vyčíst z atributu *ver* (druhá část u atributu node), který je vypočítán ze všech podporovaných protokolů klienta, viz [6].

```
1<iq from="uzivatel@jabbim.com" id="disco1"
2    to="jabinfo@jabbim.com/bot" type="get">
3  <query xmlns="http://jabber.org/protocol/disco#info"
4      node="http://code.google.com/p/exodus#QgayPKawpkPSDYmwT/WM94uAlu0=" />
5</iq>
```

Příklad 2.6: Dotaz na podporované protokoly.

Kapitola 3

Data mining

...

Terminologie

Pojem data mining nebo–li česky dolování dat se začal ve vědeckých kruzích objevovat počátkem 90. let 20. století. První zmínka pochází z konferencí věnovaných umělé inteligenci (IJCAI'89 [1]–mezinárodní konference konaná v Detroitu, AAAI'91 [2] a AAAI'93–americké konference v Californii a Washingtonu, D.C).

Tradiční metoda získání informací z dat je realizována jejich manuální analýzou a interpretací. V praxi ji například nalezneme v odvětví zdravotnictví, vědy, marketingu (efektivita reklamních kampaní, segmentace zákazníků) a dalších. Pro tato a mnoho dalších disciplín je manuální zpracování příliš pomalé, drahé a vysoce subjektivní. Na druhou stranu velikost dat dramaticky vzrostla tudíž se manuální analýza stává zcela nepraktická. Databáze rychle rostou ve dvou kategoriích :

1. počet záznamů nebo–li objektů v databázi
2. počet polí nebo–li atributů objektů v databázi

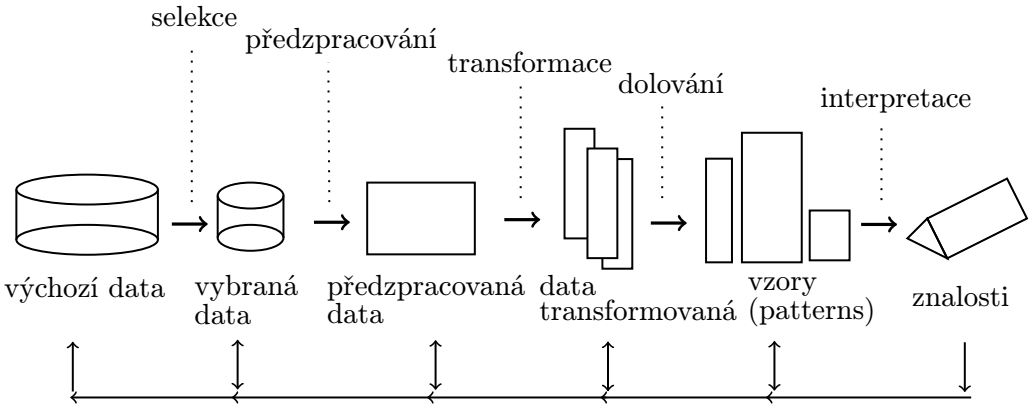
Proces data mining je pouze jedna část z odvětví nazývané dobývání znalostí z databází nebo–li KDD[3] definované níže 3.0.1. Vznik disciplíny KDD je důsledkem nepřeberného množství automaticky sbíraných dat, která je potřeba dále využívat. Podstatným znakem celého procesu je správnost reprezentace výsledků formou uživateli nejbližší. Jako příklad uveďme implikace ve tvaru rozhodovacích pravidel, asociační pravidla, rozhodovací stromy, shluky podobných dat a další. Základem KDD je praktická použitelnost metod. Očekává se zjištění nových skutečností namísto prezentování již známých informací.

Definice 3.0.1 KDD je chápáno jako interaktivní a iterativní proces tvořený kroky selekce, předzpracováním, transformace, vlastního „dolování" (data–mining viz 3.0.2) a interpretace [2].

Grafické znázornění definice 3.0.1 je popsáno schématem na obrázku 3.1, který ukazuje časový harmonogram v KDD. Schéma znázorňuje následnost jednotlivých procesů, které

[1] International Joint Conference on Artificial Intelligence
[2] Association for the Advancement of Artificial Intelligence
[3] Knowledge Discovery in Database

tvoří KDD. KDD je iterativní proces. Skutečnosti nalezené v předešlých částí zjednodušují a zpřesňují vstupy pro následující fáze. Jakmile jsou znalosti získány, jsou předvedeny uživateli. Pro přesnost může být část procesu KDD ještě upravena. Tím budou získány „přesnější a vhodnější“ výsledky.



Obrázek 3.1: Proces dobývání znalostí z databází podle knihy autora Fayyad [3].

Získávání znalostí z databází je proces složen z několika kroků vedoucích od surových dat k formě nových poznatků. Iterativní proces se skládá z následujících kroků:

- **čištění dat** – fáze, ve které jsou nepodstatné údaje odstraněny z kolekce.
- **integrace dat** – kombinování heterogenních dat z několika zdrojů do společného jediného zdroje.
- **výběr dat** – rozhodování o relevantních datech.
- **transformace dat** – také známý jako konsolidace dat. Fáze, ve které jsou vybraná data transformována do formy vhodné pro dolování.
- **data mining** – zásadní krok, ve kterém jsou aplikovány vzory na data.
- **hodnocení modelů** – vzory dat zastupují získané znalosti.
- **prezentace znalostí** – konečná fáze, zjištěné poznatky jsou reprezentovány uživateli. Tento základní krok využívá vizualizační techniky, které pomáhají uživatelům porozumět a správně interpretovat získané výsledky.

Je běžné některé z těchto kroků kombinovat dohromady. Krok čištění dat a integrace dat může být provedena společně, tak jako je uvedeno na obrázku 3.1.

V této podsekci jsou ve stručnosti vysvětleny základní nejdůležitější pojmy dále v práci využívané.

Definic výrazu data mining se v odborné literatuře nachází několik. Zde uvedená je kombinací dvou „definic“.

Definice 3.0.2 Data Mining je proces objevování znalostí, který používá různé analytické nástroje sloužící k odhalení dříve neznámých vztahů a informací z velmi rozsáhlých databází. Výsledkem je predikční model, který je podkladem pro rozhodování [13].

Mezi další četně se vyskytující pojmy v tomto odvětví patří například data, znalosti a informace. Tyto termíny jsou často mezi sebou zaměňovány, proto je níže jejich význam striktně definován.

Jedna z několika existujících definic pojmu data je uvedena v 3.0.3, která je popisuje z pohledu informačního. Data často nemají sémantiku (význam) a bývají zpracována čistě formálně.

Definice 3.0.3 Data jsou z hlediska počítačového pouze hodnoty různých datových typů [7].

Informace definované v 3.0.4, jsou zpracovaná data interpretována uživatelem.

Definice 3.0.4 Informace jsou data, která mají sémantiku (význam) [7].

Znalosti, definované v 3.0.5, patří do stejné kategorie jako informace, ale jejich interpretace bývá ještě složitější. Často jsou tvořeny shluky informací, proto bývají reprezentovány jako odvozené informace.

Definice 3.0.5 Znalosti jsou informace po jejich zařazení do souvislostí [7].

3.1 Metody dolovaní dat

Základ metod dolování dat je založen na statistice, posledních poznatcích z umělé inteligence či strojového učení. Hlavní cíle těchto netriviálních metod je společný – snaha zjištěné výsledky prezentovat srozumitelnou formou. Pro většinu používaných metod je společná vlastnost předpoklad, že objekty popsané pomocí podobných charakteristik patří do stejné skupiny (učení na základě podobnosti similarity-based learning). Objekty obsahující atributy, lze převést na body v n–rozměrném prostoru (n – počet atributů). Vychází z představy podobnosti bodů tvořící určité shluky v prostoru.

Tyto dva příklady nejsou jediné rozdíly v metodách dolování dat. Další rozdíly spočívají v:

- schopnost reprezentace shluků (např. otázka lineární separability)
- srozumitelnost nalezených znalostí pro uživatele (symbolické vs. subsymbolické metody)
- efektivnost znovupoužití nalezených znalostí
- vhodnost typů data
- a další ...

Problémy, které data mining řeší, se rozdělují do několika skupin. Výčet vybraných z nich nejčastějších:

Asociační pravidla

Asociační pravidla jsou založena na syntaxi IF-THEN. Jejich rozšíření se datuje do 90.let 20.století. Pan Agrawal je představil v souvislosti s analýzou „nákupního košíku“.

Použitelnost bude vysvětlena právě na příkladu analýzy nákupního košíku. Podstata příkladu je tvořena zákazníkem a jeho systémem nakupování. Jsou zjišťovány produkty, které jsou nakupovány současně. Hledáme nebo–li vytváříme společné vazby (asociační pravidla) mezi výrobky a určujeme jejich spolehlivost. Na základě těchto závislostí se upravuje umístění jednotlivých výrobků.

Obecně lze tedy asociační pravidla považovat za konstrukci, která z hodnot jedné transakce odvozuje možnost výskytu závislostí v jiných transakcích. Jsou hledány všechny vnitřní závislosti mezi daty.

Z pravidel vytvořených z dat zjišťujeme počet příkladů splňujících předpoklad a kolik závěr pravidla, kolik příkladů splňuje obě podmínky a počet příkladů splňující pouze druhou část pravidla. Podle knihy Berky [2] převedeme základní myšlenku asociačních pravidel *IF–THEN* do jiné terminologie:

```
Ant => Suc
```

kde *Ant* bývá interpretován jednou z možností výčtu – předpoklad, IF, levá strana pravidla nebo ancedent a *Suc* je chápan jako – závěr, ELSE, pravá strana pravidla, sukcendent. Nyní jsou uvedeny základní vlastnosti:

$n(Ant \wedge Sus) = \boldsymbol{a}$; $n(Ant \wedge \neg Sus) = \boldsymbol{b}$; $n(\neg Ant \wedge Sus) = \boldsymbol{c}$; $n(\neg Ant \wedge \neg Sus) = \boldsymbol{d}$; $n(Ant) = a+b = \boldsymbol{r}$; $n(\neg Ant) = c+d = \boldsymbol{s}$; $n(Suc) = a+c = \boldsymbol{k}$; $n(\neg Suc) = b+d = \boldsymbol{l}$; $n = a+b+c+d$;

všechna pravidla jsou shrnuta v tabulce 3.1, z nichž jsou dále počítány různé charakteristiky a následně tak hodnotit zjištěné znalosti.

	Suc	**¬Suc**	$\sum$
Ant	a	b	r
¬Ant	c	d	s
$\sum$	k	l	n

Tabulka 3.1: Kontingenční tabulka [2].

Mezi základní charakteristiky asociačních pravidel podle Agrewalova patří *podpora* a *spolehlivost*.

Klasifikace

Klasifikace bude opět vysvětlena na příkladu. Obsah databáze nebo dotazníku nám každého klienta banky zařadí do různých krizových skupin. Na základě těchto skupin pracuje „credit skóring“, jež klientovi poskytne nebo odepře například úvěr v bance. Dalším příkladem je zdravotnictví. Na základě zdravotního stavu pacienta a příznaků je zařazen do tříd, reprezentující jednotlivé nemoci.

Klasifikace rozdělí jednotlivé zkoumané elementy (podle hodnot atributů) do vhodných kategorií, které jsou předem vytvořeny navzájem podobnými objekty (tvorba profilů třídy). Při této metodě se upřednostňuje přesnost před jednoduchostí a rychlostí. Je snaha o nalezení obecných závěrů z chování dat. Zdroje klasifikovaných objektů většinou tvoří jednotlivé

řádky v databázi. Vzory dat vytváří instance. Jejich vlastnosti reprezentují atributy vyjádřené číselnou hodnotou.

Z matematického pohledu klasifikace představuje funkci o více proměnných. Atribut instance odpovídá proměnné a funkční hodnota výstupu.

Modely

Základem modelů jsou trénovací data. Příklad vybraných klasifikačních modelů:

- Rozhodovací stromy
- Neuronové sítě
- Statistické metody
- Klasifikační pravidla
- Využití vzdálenosti
- a další.

Predikce

Predikce patří mezi velmi známé procesy, které na základě získaných znalostí předpovídají následující vývoj. Chronologicky seřazená data a vývoj jejich hodnot v minulosti tvoří základ pro určení hodnot budoucích. Předpokládá se, že data v minulosti budou chovat stejně nebo alespoň podobně i v budoucnu. Využití naleznou v předpovědi počasí (z naměřených meteorologických hodnot se určují budoucí předpokládané teploty), při vývoji cen na burze a dalších.

Shlukování

Metoda zaměřená na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků. Tato část bude podrobně rozebrána v následující kapitole 3.2.

3.2 Shlukování

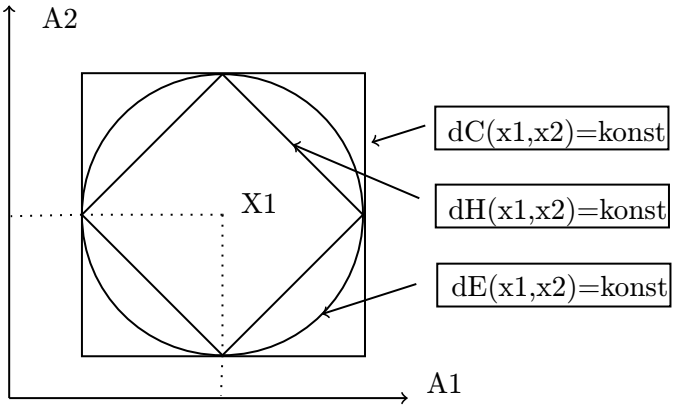
V této podkapitole je shlukování rozděleno na několik metod shlukové analýzy podle [4]. U každé z nich jsou popsány její základní vlastnosti a uvedeny nejrozšířenější algoritmy. Poslední metoda *metoda rozkladu* 3.2 je rozebrána podrobněji z důvodu jejího praktického využití v této práci.

Shlukování je zaměřeno na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků.

Výpočet vzdáleností mezi objekty patří, pro většinu níže uvedených metod a algoritmů, mezi nedílnou součást. Tato vzdálenost lze vyjádřit různými mírami například *Hammingovou vzdáleností* (dH), *Euklidovskou vzdáleností* (dE) a *Čebyševovou vzdáleností* (dC). Rozdíl mezi těmito typy určující vzdálenosti, graficky vyjadřuje obrázek č. 3.2.

Metody hierarchické

...



Obrázek 3.2: Srovnání výpočtu vzdáleností od bodu x_1 [2].

Metody založené na mřížce

Metody založené na mřížce kvantují datový prostor do konečného počtu pravoúhlých buněk, které jsou uspořádány do víceúrovňové mřížkové struktury. Zmíněná struktura tvoří základ pro shlukové operace. Hlavní výhoda tohoto přístupu je rychlost zpracování, které většinou nebere ohled na počet datových objektů. Čas zpracování závisí pouze na počtu buněk v každé dimenzi kvantovaného prostoru.

Mezi zástupce metod založených na mřížce patří metoda STING[4], který pracuje se statickými informacemi uloženými v buňkách mřížky. Algoritmus je rozdělen do dvou částí. První si klade za cíl rekurzivně rozdělit datový prostor na pravoúhlé buňky. Druhá fáze testuje spojitost mezi sousedy relevantních buněk.

Mezi další metody založené na mřížce patří CLIQUE[5] a WaveCluster[6]. WaveCluster využívá vlnkové transformace k rozdělení prostoru dat. Tato transformace zdůrazňuje shluky v prostoru a objekty jim vzdálené potlačuje.

Metody založené na modelu

Metody založené na modelu se pokouší přiřadit data k určitému matematickému modelu na základě společných optimalizovaných vlastností. Většina procesů je založena na předpokladu generování dat pomocí standardních statistik.

Mezi zástupné metody této shlukovací analýzy se řadí Expectation–Maximization (EM), SOON[7] a částicové filtry (particle filtres). Algoritmus SOON je založen na neuronové síti. Je to metoda vycházející z algoritmu SOM[8]. Metoda EM je rozšířením algoritmus *k–means* o němž se budeme podrobně zabývat v následujících částí 3.2.

Metody založené na hustotě

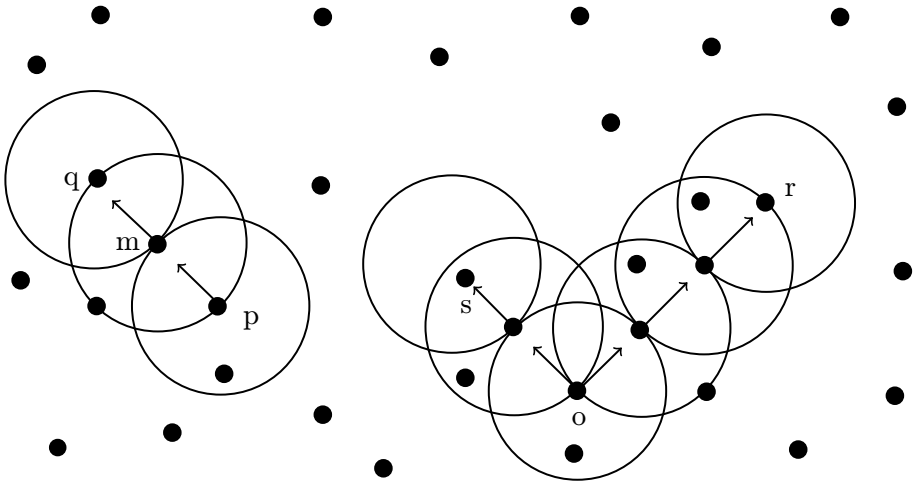
Vychází z m–rozměrného prostoru do něhož, jsou zobrazeny objekty ve formě bodů. Místa v prostoru s větší koncentrací objektů ve srovnáním s ostatními oblastmi jsou nazývány

[4] STatistical INformation Grid
[5] CLustering In QUEst
[6] Clustering Using Wavelet Transformation
[7] Self Organizing Oscillator Network
[8] Self–Organizing Map

shluky. Jsou to ta místa, která hledáme. Výchozí předpoklad je existence okolí jednotlivých bodů (sousedství). Tato metoda se dokáže vypořádat s hodnotami velmi vzdálenými označovanými jako šum.

Jako příklad uveďme metody DBSCAN[9], OPTICS[10] a DENCLUE[11]. Metoda DBSCAN je založena na hustotě objektů v prostoru. U jednotlivých objektů je zkoumáno jejich okolí. Algoritmus je ovlivňován dvěma parametry ε (velikost shluku) a $MinPts$ (minimální počet objektů v daném shluku), které spolu úzce souvisí (viz [4]). Bod splňující obě podmínky je označen za jádro. Pomocí jader se rozšiřuje množina objektů spojených na základě hustoty. Obsahuje–li jádro x_1 ve svém okolí další jádro x_2 znamená to, že jádro x_1 je přímo dosažitelné z jádra x_2. Tímto způsobem se vytváří výsledné *shluky*. V opačném případě, body nesplňující dvě zmíněné podmínky, jsou označeny jako *šum*.

Příklad postupu algortimu DBSCAN zobrazuje obrázek č. 3.3. Kde ε–sousedství tvoří kruh a $MinPts$ má hondotu 3.



Obrázek 3.3: Algoritmus DBSCAN [4].

Shlukování velkých dat

Všechny zde doposud zmiňované metody poskytují dobré výsledky pouze s malým počtem dimenzí. S narůstajícím počtem atributů roste počet nerelevantních dimenzí určených pro shlukování. S tímto také přibývá zvětšená produkce zašumění a znesnadnění nalezení relevantních shluků. Data jsou roztroušena do mnoho dimenzí a tím odpadá možnost použití vzdálenostních funkcí.

Zmíněné problémy shlukování velkých dat řeší dvě techniky *metoda transformace rysů* a *metoda výběru atributů*.

Pro efektivní shlukování je možné použít například algoritmus CLIQUE[12], PROCLUS[13], ENCLUS[14] a další.

[9] Density–Based Spatial Clustering of Applications with Noise
[10] Ordering Points To Identify the Clustering Strukture
[11] DENsity–base CLUstering
[12] CLustering In QUEst
[13] PROjested CLUStering
[14] ENntropy–based CLUStering

Metody rozkladu

...

k–Means

...popis a za pis algoritmu

k–Medoids

...

3.3 Programy

Fitminer

Rapid miner

Weka

Kapitola 4

Implementace

4.1 Databáze

Návrh databáze

4.2 Architektura

–robot plus databaze –rapidminer–spusteni davkove/vlastni algoritmus v PHP –webova implementace prezentuje vysledky

4.3 Robot

–class diagram, pomoci nej popsat strukturu, mozne rozsireni

gloox

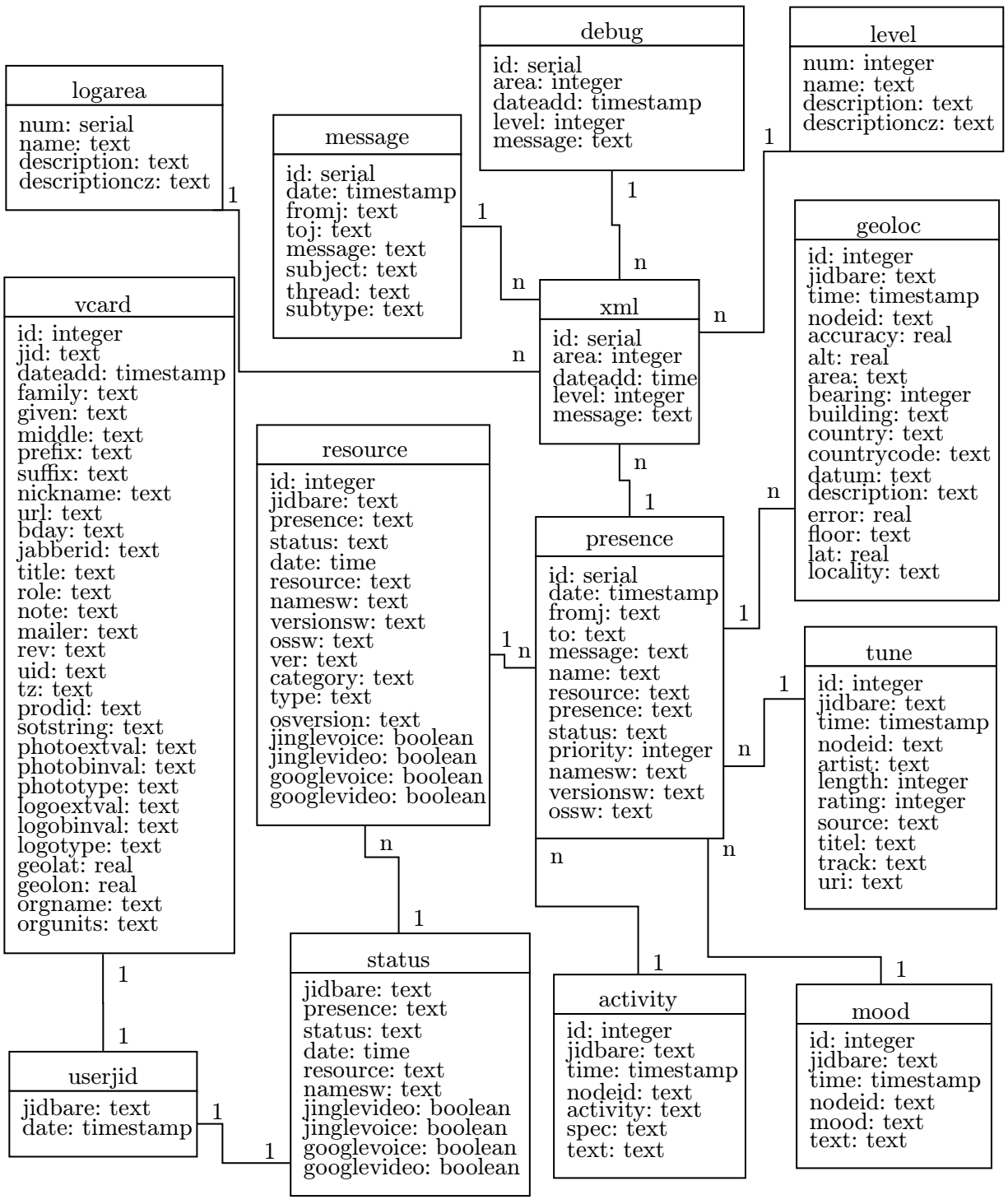
Gloox je stabilní Jabber/XMPP knihovna vydávána pod licencí GNU GPL. Je určena pro vývoj klienta a komponent. Jelikož je psána v ANSCI C++ mezi podporované platformy patří Linux, Windows, Mac OS X, Symbian/Nokia S60, FreeBSD a další systémy stavějící na ANSI C++ kompilátor.

Pomocí knihovny gloox je psán bot v této práci. Byla vybrána na základě požadavku psaní programu v jazyce C/C++ a operačním systému Linux. V porovnání s jinými knihovnami pro jazyk C nebo C++ disponuje lepší podporou a dokumentací. Gloox plně podporuje standart XMPP Core [17] a z větší části i standard XMPP IM [18]. Dodatečně je plně podporováno kolem 30 XEP standardů například XEP-0054 vcard-temp a další.

Návrh bota

Transformace

temporalni databaze



Obrázek 4.1: Struktura databáze

Kapitola 5

Vyhodnocení výsledků

–charakter dat, jaka data jsem nasbiral –

Kapitola 6

Závěr

Literatura

[1] Adams, D.: *Programming jabber*. Sebastopol: O'Reilly, první vydání, 2002, 455 s., iSBN 05-960-0202-5.

[2] Berka, P.: *Dobývání znalostí z databází*. Praha: Academia, první vydání, 2003, 366 s., iSBN 80-200-1062-9.

[3] Fayyad, U. M.; Smyth, P.: *Advances in knowledge discovery and data mining*. California: MIT Press, první vydání, 1996, 611 s., iSBN 02-625-6097-6.

[4] Han, J.; Kamber, M.: *Data mining : concepts and techniques*. San Francisco: Morgan Kaufmann Publisher, druhé vydání, 2006, 770 s., iSBN 15-586-0901-6.

[5] Hildebrand, J.; Saint-Andre, P.: XEP-0080: User Location. [online], 15-09-2009, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0080.html

[6] Hildebrand, J.; Saint-Andre, P.; Tronçon, R.; aj.: XEP-0115: Entity Capabilities. [online], 02-26-2008, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0115.html

[7] Hruška, T.: *Informační systémy : IIS/PIS*. Brno: Fakulta informačních technologií, 2008, 14733 s.

[8] Kolektiv autorů: Extensible Markup Language (XML) 1.0. [online], 26-11-2008, [cit. 17. dubna 2011].
URL http://www.w3.org/TR/2008/REC-xml-20081126/

[9] Kosek, J.: *XML pro každého : podrobný průvodce*. Praha: Grada, první vydání, 2000, 163 s., iSBN 80-716-9860-1.

[10] Meijer, R.; Saint-Andre, P.: XEP-0108: User Activity. [online], 29-10-2008, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0108.html

[11] Millard, P.; Saint-Andre, P.; Meijer, R.: XEP-0060: Publish-Subscribe. [online], 12-07-2010, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0060.html

[12] Moore, D.; Wright, W.: *Jabber developer's handbook*. Indianapolis: Sams Publishing, první vydání, 2004, 487 s., iSBN 06-723-2536-5.

[13] Nemrava, M.; Pospíšil, J.: Dolování dat a jeho aplikace. [online], 2006, [cit. 17. dubna 2011].
URL http://www.spatial.cs.umn.edu/paper_ps/dmchap.pdf

[14] Saint-Andre, P.: XEP-0118: User Tune. [online], 30-01-2008, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0118.html

[15] Saint-Andre, P.; Meijer, R.: XEP-0107: User Mood. [online], 29-10-2008, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0107.html

[16] Saint-Andre, P.; Smith, K.: XEP-0163: Personal Eventing Protocol. [online], 12-07-2010, [cit. 17. dubna 2011].
URL http://xmpp.org/extensions/xep-0163.html

[17] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Core. [online], 10-2004, [cit. 17. dubna 2011].
URL http://tools.ietf.org/html/rfc3920

[18] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. [online], 10-2004, [cit. 17. dubna 2011].
URL http://tools.ietf.org/html/rfc3921

[19] Saint-André, P.; Smith, K.; Troncon, R.: *XMPP : the definitive guide : building real-time applications with jabber technologies*. Sebastopol: O'Reilly, první vydání, 2009, 287 s., iSBN 978-059-6521-264.

Příloha A

Obsah CD

Příloha B

Manual

Příloha C

Konfigrační soubor

Příloha D

Slovník výrazů

DNS — Domain Name System

GPG — dkshckdsjvlsdjvodsvjdfokj

IM služby — dkshckdsjvlsdjvodsvjdfokj

IP — Internet Protocol

JEP — dkshckdsjvlsdjvodsvjdfokj

JID — dkshckdsjvlsdjvodsvjdfokj

SASL — dkshckdsjvlsdjvodsvjdfokj

TCP — dkshckdsjvlsdjvodsvjdfokj

TLS — dkshckdsjvlsdjvodsvjdfokj

WWW — dkshckdsjvlsdjvodsvjdfokj

XEP — dkshckdsjvlsdjvodsvjdfokj

XML — dkshckdsjvlsdjvodsvjdfokj

XMPP — dkshckdsjvlsdjvodsvjdfokj

e-mail — dkshckdsjvlsdjvodsvjdfokj

jabber — dkshckdsjvlsdjvodsvjdfokj

klient — dkshckdsjvlsdjvodsvjdfokj

presence — dkshckdsjvlsdjvodsvjdfokj

server — dkshckdsjvlsdjvodsvjdfokj

stanza — dkshckdsjvlsdjvodsvjdfokj

vCard — dkshckdsjvlsdjvodsvjdfokj

Příloha E

Stanza - základní schéma

E.1 iq

```
<iq from=""
    to=""
    type="[get,set,result,error]"
    id="">
  Namespace
</iq>
```

Příklad E.1: Popis elementu *iq*.

jabber:client	jabber:server	jabber:iq:auth	jabber:iq:register
jabber:iq:roster	jabber:x:offline	jabber:iq:agent	jabber:iq:agents
jabber:x:delay	jabber:iq:version	jabber:iq:time	vcard-temp
jabber:iq:private	jabber:iq:search	jabber:iq:oob	jabber:x:oob
jabber:iq:admin	jabber:iq:filter	jabber:iq:auth:0k	jabber:iq:browse
jabber:x:event	jabber:iq:conference	jabber:x:signed	jabber:x:encrypted
jabber:iq:gateway	jabber:iq:last	jabber:x:envelope	jabber:x:expire
jabber:xdb:ginsert	jabber:xdb:nslist	texthttp://www.w3.org/1999/xhtml	

Tabulka E.1: Přehled Namespace elementu <*iq/*>.

E.2 Message

```
<message from=""
         to=""
         type="[normal,chat,groupchat, headline, error]"
         id=""
  <body>  </body>
  <x xmlns="jabber:x:event">
     <[Offline, Delivered, Displayed, Composing]/>
  <subject>  </subject>
  <thread>   </thread>
  <error>    </error>
  <x>        </x>
</message>
```

Obrázek E.1: Popis elementu <*message/*>.

E.3 Presence

```
<presence from=""
          to=""
          type="[available, unavailable, probe, subscribe,
                 unsubscribe, subscribed, unsubscribed, error]"
          id=""
  <show>
    [away, chat, dnd, normal, xa]
  </show>
  <status>     </status>
  <priority>   </priority>
  <error>      </error>
</presence>
```

Obrázek E.2: Popis elementu <*presence/*>.

E.4 Přehled průběhu rozšíření

```
<iq from="uzivatel@jabbim.com" type="set" id="pub1">
   <pubsub xmlns="http://jabber.org/protocol/pubsub">
      <publish node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
               <track>5</track>
            </tune>
         </item>
      </publish>
   </pubsub>
</iq>
```

Obrázek E.3: Informování server o právě přehrávající písničce.

```
<message from="uzivatel@jabbim.com" type="set"
         to="jabinfo@jabbim.com/bot" id="pub1">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
               <track>5</track>
            </tune>
         </item>
      </items>
   </event>
</message>
```

Obrázek E.4: Server informuje uživatele podporující rozšíření o stavu *uzivatel@jabbim.com*.

```
<message from="uzivatel@jabbim.com" type="set"
         to="uzivatel@jabbim.com/doma" id="pub2">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
               <track>5</track>
            </tune>
         </item>
      </items>
   </event>
</message>
```

Obrázek E.5: Server přepošle informace o přehrávané písničce všem otevřeným spojením uživatele *uzivatel@jabbim.com*.

```
<iq from="uzivatel@jabbim.com/prace" type="set" id="pub1">
   <pubsub xmlns="http://jabber.org/protocol/pubsub">
      <publish node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune"/>
         </item>
      </publish>
   </pubsub>
</iq>
```

Obrázek E.6: Uživatel ukončil „vysílaní“ rozšířených zpráv o svém stavu.

```
<message from="uzivatel@jabbim.com"
         to="jabinfo@jabbim.com/bot">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune"/>
         </item>
      </items>
   </event>
</message>
```

Obrázek E.7: Server informuje klienty o ukončení šíření rozšířeného statusu uživatele *uzivatel@jabbim.com*.

Příloha F

Přehled klientů a jejich rozšíření

..... sdufiysdiufisudfyn df ..
. zrofj http://xmpp.org/xmpp-software/clients/ ..sd.fsdf

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
Adium										
Agile Messenger										
AQQ										
Ayttm										
beejive										
Beem										
BitlBee										
Bombus										
BuddyMob										
Chatopus										
Citron										
Claros Chat										
climm										
Coccinella										
Crosstalk										
Digsby										
eM Client										
emite										
Empathy										
Exodus										
Finch										
Gajim										
Galaxium										
glu										
GNU Freetalk										
Gossip										
iChat										
iJab										
IM+										
imov Messenger										
irssi-xmpp										
Jabbear										
Jabber Mix Client										
jabber.el										
Jabbim										
Jabbim for Android										
Jabiru										
JAJC										
Jappix										

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
JBuddy Messenger										
Jeti										
Jitsi (SIP Communicator)										
JWChat										
Kadu										
Kopete										
Lampiro										
m-im										
mcabber										
mChat										
Miranda IM										
Monal IM										
OctroTalk										
OneTeam										
OneTeam for iPhone										
Oyo										
Pandion										
Poezio										
Pidgin										
Prodromus										
Psi										
Psi+										
Quiet Internet Pager (QIP)										
qutIM										
saje										
SamePlace										
Sim-IM										
Slimster										
SoapBox Communicator										
Spark										
SparkWeb										
Synapse										
Talkonaut										
Tigase Messenger										
Tigase Minichat										
Tkabber										
Tlen										
Trillian										
TrophyIM										
V&V Messenger										
Vacuum-IM										
Vayusphere										
WTW										
Xabber										
xmppchat										
Yambi										
Yaxim										

Tabulka F.1: Přehled podporovaných rozšíření u jednotlivých klientů.