БИБИС

Данијела Боберић
Душан Сурла

Преузимање библиографских записа по Z39.50 стандарду

Иновациони центар за електронске библиотеке и архиве
Департман за математику и информатику
Природно–математички факултет
Универзитет у Новом Саду

Данијела Боберић
Душан Сурла

Преузимање библиографских записа по Z39.50 стандарду

Нови Сад, 2007.

Назив монографије
Преузимање библиографских записа по стандарду Z39.50

Аутори
мр Данијела Боберић, истраживач приправник, Иновациони центар за електронске библиотеке и архиве, Департман за математику и информатику, Природно-математички факултет, Нови Сад

др Душан Сурла, редовни професор, Иновациони центар за електронске библиотеке и архиве, Департман за математику и информатику, Природно-математички факултет, Нови Сад

Рецензенти

др Милош Рацковић, редовни професор, Природно-математички факултет, Нови Сад

др Зора Коњовић, редовни професор, Факултет техничких наука, Нови Сад

др Бранко Милосављевић, доцент, Факултет техничких наука, Нови Сад

Издавач
Природно-математички факултет, Нови Сад

Главни и одговорни уредник
Проф. др Мирослав Весковић, декан Природно-математичког факултета у Новом Саду

Научно-наставно веће Природно-математичког факултета у Новом Саду, на седници 22. 10. 2007. године, одобрило је штампање и употребу ове монографије.

Корице
Данијела Тешендић

Штампа
АМБ Економик

Тираж
150

Садржај

ПРЕДГОВОР **5**

САЖЕТАК **7**

ABSTRACT **9**

УВОД **11**

1.1 Z39.50 СТАНДАРД У ПРАКСИ 11
1.1.1 Клијентске апликације *12*
1.1.2 Серверске апликације *17*
1.2 БИБЛИОТЕЧКИ ИНФОРМАЦИОНИ СИСТЕМ БИСИС 20
1.2.1 Преузимање библиографских записа *20*
1.2.2 Примена XML технологије у развоју система БИСИС *21*

СТАНДАРД Z39.50 ..... **25**

2.1 Z39.50 СЕРВИСИ 26
2.2 ФОРМАТ ЗАПИСА 30
2.3 УПИТНИ ЈЕЗИЦИ 36
2.4 СКУПОВИ АТРИБУТА 36
2.4.1 Скуп атрибута Bib-1 ..... *37*
2.5 ПРОФИЛИ 39
2.6 ОСТАЛИ СТАНДАРДИ 40

УПИТНИ ЈЕЗИЦИ **43**

3.1 Z39.50 УПИТ ТИПА -1 43
3.1.1 Синтакса упита *43*
3.1.2 XML шема упита *45*
3.2 Z39.50 УПИТ ТИПА-100 48
3.2.1 Синтакса упита *48*
3.2.2 XML шема упита *49*
3.3 Z39.50 УПИТ ТИПА-102 52
3.3.1 Синтакса упита *52*
3.3.2 XML шема упита *54*

МОДЕЛИРАЊЕ СИСТЕМА **57**

4.1 ДИЈАГРАМ СЛУЧАЈЕВА КОРИШЋЕЊА 58
4.2 ДИЈАГРАМ ПАКЕТА 63
4.3 ДИЈАГРАМ КЛАСА 64
4.4 ДИЈАГРАМ СЕКВЕНЦИ 67
4.4.1 Дијаграм секвенци Search *67*
4.4.2 Дијаграм секвенци createQuery *68*
4.4.3 Дијаграм секвенци createOperand *71*
4.4.4 Дијаграм секвенци reviewRecords *72*

ИМПЛЕМЕНТАЦИЈА СИСТЕМА **75**

5.1 Архитектура алата JAFER ... 75
5.2 Едитор за преузимање библиографске грађе ... 78

ОПИС КОРИШЋЕЊА СИСТЕМА ... 87

6.1 Конфигурисање параметара библиотеке ... 88
6.2 Претраживање по вредностима *Use* атрибута ... 89
6.3 Претраживање употребом логичких оператора ... 94
6.4 Напредно претраживање ... 96

ПРИЛОГ 1 ... 99

ВРЕДНОСТИ ТИПОВА АТРИБУТА *BIB-1* ... 99

ЛИТЕРАТУРА ... 111

Предговор

У овој монографији приказан је део добијених резултата на пројекту *Апстрактни модели и примене у рачунарским наукама*, број 144017А, који финансира Министарство науке и заштите животне средине Републике Србије. Носилац истраживања овог пројекта је Природно-математички факултет у Новом Саду.

Предмет истраживања у овој монографији је моделирање и имплементација система за преузимање библиографских записа. Основна улога имплементираног система је да омогући преузимање библиографских записа из удаљених база података путем протокола Z39.50. Архитектура система је вишеслојна. Слој клијентске стране реализован је у Java окружењу (Swing компоненте). Слој пословне логике је базиран на XML технологијама.

Монографија садржи следећа поглавља:

1. Увод
2. Стандард Z39.50
3. Упитни језици
4. Моделирање система
5. Имплементација система
6. Опис коришћења система

У првом поглављу извршена је анализа постојећих клијентских и серверских апликација које врше претраживање и преузимање библиографских записа по стандарду Z39.50. Такође, је

У другом поглављу дат је опис стандарда Z39.50 који се користи при претраживању библиографских записа. Такође су описани и неки основни концепти дефинисани у стандарду.

У трећем поглављу су описани упитни језици који се могу користити при претраживању ако се користи протокол Z39.50. Поред овога, за сваки описани упит, поред његове синтаксе, је приказана и XML шема тог упита која се може користити у даљој обради постављеног упита.

Спецификација апликације која представља практични део рада је дата у четвртом поглављу. Систем је моделиран помоћу CASE алата који подржавају објектно моделирање (дијаграми случајева коришћења, дијаграм пакета, дијаграм класа и дијаграми секвенци).

Детаљи везани за имплементацију апликације дати су у петом поглављу. Имплементација је реализована на основу спецификације дате у четвртом

поглављу у Java окружењу. Дат је кратак опис јавно доступног софтерског пакета JAFER, који је коришћен за имплементацију едитора.

Опис коришћења апликације и примери упита и резултата претраживања дати су у шестом поглављу.

Поред наведених поглавља дат је и апстракт на српском и енглеском језику.

Захваљујемо се рецензентима који су детаљно прегледали текст монографије и својим корисним предлозима утицала на квалитет и крајњи изглед монографије.

Нови Сад, 2007. Аутори

Сажетак

Циљ. Циљ истраживања је моделирање и имплементација XML едитора за претраживање и преузимање библиографских записа. Помоћу овог едитора омогућено је претраживање и преузимање библиографских записа из удаљених база података путем протокола Z39.50. Моделирање информационих захтева овог едитора урађено је у обједињеном језику моделирања (UML 2.0). Клијентске апликација је реализована у Java окружењу, а пословна логика је базиран на XML технологијама.

Методологија. Коришћена је објектно оријентисана методологија за пројектовање софтверских компоненти као и CASE алати. Едитор је реализован као самостална софтверска апликација, али је архитектура софтвера таква да омогућава интеграцију едитора у различите библиотечке информационе системе.

Резултат истраживања. Резултат је апликација едитора за преузимање библиографских записа по стандарду Z39.50. Едитор подржава формирање упита типа-1 који је дефинисан стандардом Z39.50. Имплементација едитора је заснована на XML технологијама чиме је омогућен једноставан прелазак на друге типове упита који су дефинисани стандардом. Апликација је верификована претраживањем и преузимањем библиографских записа из Конгресне библиотеке у Вашингтону и Универзитетске библиотеке у Оксфорду.

Ограничења резултата истраживања. Едитор подржава само упит типа-1 који је моделиран XML шема језиком. Додавање других типова упита захтевало би њихово моделирање помоћу XML шеме као и одговарује измене екранских форми.

Практична примена. Едитор је интегрисан у библиотечки софтверски систем БИСИС. На овај начин омогућено је преузимање библиографских записа из светских библиотека које користе протокол Z39.50. Преузети запис се складишти у објектну структуру која се даље обрађује у едитору за обраду библиографске грађе система БИСИС.

*Оригиналност резултата истраживања***.** Оригиналност рада је у архитектури софтверског система која је заснована на XML документима и независна је од стандарда којим је дефинисан упитни језик и софтверског система у који се систем интегрише. XML документ који садржи податке о креираном упиту представља улазну информацију у едитору. Према томе, увођење новог типа упита састоји су у креирању одговарајућег XML

документа који би се као улазна информација проследио едитору. Након извршавања упита добија се скуп резултата који се може преузети у виду XML документ и као такав обрађивати у различитим софтверским системима.

Abstract

Purpose. The main aim of this research was modelling and implementation of XML editor for searching and retrieving bibliographic records. The editor provides search and retrieval of bibliographic records residing in remote databases by means of Z39.50 protocol. Modelling of the functional requirements of this system is done by using Unified Modelling Language (UML 2.0). Client application was developed in Java environment and business logic was based on XML technologies.

Methodology. The object – oriented methodology was used for software components design, as well as appropriate CASE tools. The XML editor is implemented as a stand alone application, but architecture of the system is designed in order to provide its integration into various library information systems.

Findings. The result of the research is the application for retrieval of bibliographic records by Z39.50 standard. The editor supports creating *type-1* query of the Z39.50 standard. Implementation of the editor is based on XML technologies thus allowing easy and simple implementation of other types of queries defined by standard. Application is verified by successful search and retrieval of the bibliographic records from Library of Congress in Washington, and Oxford University Library.

Research limitations. Editor is devised to support only *type-1* query which is modelled by XML Schema language. Implementation of the other types of queries demands their modelling by XML Schema language, and some changes of the user interface.

Practical implications. Editor is integrated into library software system BISIS. In that way, the retrieval of bibliographic records from the libraries all over the world which use Z39.50 protocol is supported. The record retrieved is stored in an object structure which is then submitted to the editor for processing bibliographic material of system BISIS.

*The originality of research findings***.** The originality of this work lies in the architecture of the software system which is based on XML documents and independent of the standard which describes query language. Also, the system can be integrated into different librarian software systems. XML document containing information on formed query is the input to the editor. Consequently, implementation of the new query type is reduced only to the creation of the appropriate XML document which will be submitted to the editor as an input. Once the submitted query is executed, the result set is

obtained which also could be presented as XML document and processed by various software systems.

Поглавље 1

Увод

Резултати приказани у овој монографији добијени су у оквиру развоја библиотечког софтверског система БИСИС, и то оног дела који се односи на примену XML (*Extensible Markup Language*) технологија за размену библиографских записа у електронској форми. Постојање интероператибилних система који омогућавају проналажење и размену квалитетних информација електронским путем постало је основа пословања. Ови системи омогућавају да библиотекари утроше мање времена на сам процес каталогизације књига, али такође на овај начин могу да провере квалитет својих библиографских записа са разним светским институцијама које се баве каталогизацијом, као што је на пример Конгресна библиотека у Вашингтону.

Да би се омогућило претраживање различитих база података доступних преко Интернета и преузимање библиографских записа неопходно је постојање комуникационог протокола који је независтан од окружења (базе података, платформе, програмског језика). Један од таквих протокола, који је првенствено намењен за комуникацију између библиотечких информационих система описан је стандардом Z39.50 [Z3950].

Z39.50 (*ANSI/NISO Z39.50-2003, Information Retrieval(Z39.50): Application Service Definition and Protocol Specification)* је стандард организације NISO (*National Information Standards Organization*). Прва верзија стандарда појавила се 1988. године, а 1992. и 1995. године изашле су верзије 2 и 3. Тренутно је актуелна верзија број три. Z39.50 је стандард за проналажење и преузимање података у дистрибуираним рачунарским системима. Детаљнији опис овог стандарда дат је у другом поглављу а кратак преглед софтверских решења који у свом раду примењују овај стандард дат је у наредном одељку. На крају овог поглавља дат је преглед актуелних истраживања у оквиру система БИСИС.

1.1 Z39.50 стандард у пракси

Стандард Z39.50 је широко распрострањен унутар библиотечких система и променио је начин функционисања ових система. На сајту Конгресне библиотеке може се наћи списак преко 900 библиотека чији фондови се могу претраживати помоћу Z39.50 протокола који је дефинисан стандардом Z39.50 [Servers]. Већина библиотека нуди претрагу преко

одређене Интернет странице, међутим постоји и велики број самосталних апликација које се могу користити при претраживању заснованом на Z39.50 протоколу.

Да би се омогућило коришћење овог стандарда у пракси, потребно је да постоје две стране у комуникацији. Са једне стране постоји апликација која кориснику омогућава да на једноставан начин постави упит и изабере библиотеку чији библиотечки фонд жели да претражује. Оваква апликација се у терминологији Z39.50 стандарда назива Z-клијент. Са друге страни налази се апликација која треба да одговори на захтев Z-клијента, односно да изврши постављени упит и проследи одговарајући скуп погодака. Оваква апликација се у терминологији Z39.50 стандарда назива Z-сервер.

У наредним одељцима дате су основне карактеристике постојећих Z-клијената и Z-сервера, као и одређене статистике везане за распрострањеност Z39.50 стандарда у свету.

1.1.1 Клијентске апликације

Анализом постојећих софтверских решења Z-клијената, уочене су одређене функционалности које би таква Z39.50 клијентска апликација требала да испуни.

Минимални захтеви су:

- додавање и ажурирање библиотека које се претражују;
- подржавање свих типова атрибута из скупа *bib-1*, детаљи везани за овај скуп дати су у одељку 2.4.1;
- употреба логичких оператора;
- приказ записа у различитим форматима;
- приказ формираног упита;
- подршка за UTF-8 кодни распоред.

Напредне могућности су:

- претраживање више библиотека истовремено;
- сортирање добијених записа по одређеним критеријумима;
- чување креираних упита и могућност поновног коришћења;
- постојање шифарника за поједине критеријуме претраге (одредница, аутори).

У наставку су приказане неке Z39.50 клијентске апликације које су доступне или као *open source* софтвери или као демо верзије одговарајућег комерцијалног софтвера. Циљ овог прегледа постојећих софтверских решења, у домену клијентске стране Z39.50 протокола, је да се уоче њихове предности, али и мане како би то допринело квалитетнијем развоју апликације за размену библиографских записа која је описана у овој монографији.

MarcEdit [MarcEdit] је апликација која представља едитор за рад са записима. Омогућава креирање новог записа и његову модификацију, али такође обезбеђује и преузимање записа са неког Z-сервера. За имлементацију Z39.50 клијента искоришћен је алат YAZ++, који је писан у програмском језику C++. Програм представља некомерцијални софтвер који ради под Windows оперативним системом.

Апликација пружа могућност да клијент сам постави параметре конекције за Z-сервер који жели да претражује. Под параметрима конекције подразумевају се адреса сервера и назив базе, евентуално корисничко име ако је потребно за аутентификацију. Могућности претраге су веома скромне. Апликација подржава само упите типа-1. Могуће је претраживати по наслову, аутору, кључној речи, одредници и по још пар атрибута. Не постоји могућност употребе логичких оператора, нити постављања вредности за све атрибуте из скуп *bib -1*. Добијени резултати се приказују у MARC 21 пуном формату (full format) [MARC21].

Willow (**W**ashington **I**nformation **L**ooker-upper **L**ayered **O**ver **W**indows) [Willow] представља аплет писан у програмском језику Јава. Програм ради под Windows оперативним системом, али постоји истоимена верзија програма за UNIX оперативни систем. Ово је некомерционални програм, који обезбеђује једноставан графички интерфејс, који може служити за претрагу база података које садрже податке о библиографским записима. Развијен је на Универзитету у Вашингтону где се користио до 1999. године, када је и прекинут даљи рад на овом пројекту.

Програм пружа могућност претраге по одређеним критеријумима који зависе од избора библиотеке. При коришћењу програма корисник може изабрати више критеријума за претрагу, али су ти критеријуми повезани логичким оператором AND. Не постоји могућност коришћења осталих логичких оператора подржаних Z39.50 стандардом. Такође, за одређене библиотеке постоје шифарници за поједине критеријуме претраге. Такав критеријум је на пример предметна одредница, и тада корисник може изабрати неку од понуђених вредности за тај критеријум или самостално унети вредност.

Након завршене претраге, у случају да има погодака појављују се основни подаци о запису, као што су име аутора, назив дела и година издања. Селектовањем једног од приказаних записа у доњој половини прозора појављују се и остали подаци везани за тај запис. Могуће је приказати и записе у пуном формату, али за тај део посла је задужен Web браузер. Добијени запис се може снимити, послати електронском поштом или одштампати.

ZSearcher [ZSearcher] је Z39.50 клијент који омогућава Интернет претрагу и преузимање библиографских података из библиотека које су доступне преко Z-сервера. Ради под *Windows* оперативним системом и обезбеђује истовремену претрагу различитих сервера и то са великом ефикасношћу при преузимању записа. Софтвер је комерцијалан, али је доступан као демо верзија. Софтвер не пружа подршку за UTF-8 кодни распоред.

Апликација обезбеђује следеће операције:

- креирање конекције,
- претрагу записа и
- преузимање записа и њихов приказ.

Приликом конекције, Z-сервер је могуће изабрати из листе постојећих сервера или креирати нови сервер ако он не постоји у листи понуђених. За сваки сервер постоји податак о његовој адреси и порту, назив базе којој се приступа, као и потребни подаци за аутентификацију.

Ова апликација подржава скуп атрибута *bib-1*, тако да је за сваку појединачну библиотеку могуће дефинисати атрибуте из овог скупа које библиотека подржава. Међутим приликом претраге кориснику су доступни и остали атрибути, тако да је дефинисање ових атрибута само информативно. Такође, могу се дефинисати и остали атрибути као што су: *Position*, *Relation*, *Structure*, *Truncation, Completeness*, али је недостатак оваквог начина дефинисања то што се они дефинишу приликом конекције на одређену базу и затим се те вредности користе у свим даљим претрагама. Да би се промениле ове вредности потребно је затворити конекцију, изабрати нове вредности и поново успоставити конекцију.

Постоје три врсте претраге: једноставна, са логичким операторима и напредна претрага. Код једноставне претраге корисник апликације треба само да изабере по ком префиксу жели да претражује и да унесе текст упита. Слично је и када је у питању претрага са логичким операторима, само је сада могуће појединачне једноставне упите груписати помоћу логичких везника као што су AND, OR и NOT. Број једноставних упита који се комбинују са логичким операторима је неограничен. Напредна претрага је слична претрази са логичким операторима, али је овде могуће

комбиновати само два једноставна упита уз коришћење оператора AND и NOT. Једна од погодности ове апликације је да постоји историја свих упита који су употребљени, па их је лако могуће поново искористити.

По завршетку претраге добијени записи се приказују у MARC 21 пуном формату (*full format*). Записе је могуће снимити као обичан текст или као XML документ. Такође, постоји могућност и сортирања записа по аутору, наслову или години издања.

Surpass Cpoycat [CopyCat] је апликација која ради под *Windows* оперативним системом и омогућава претраживање и добављање записа у MARC 21 формату помоћу протокола Z39.50. Пружа могућност истовремене вишеструке претраге различитих библиотека. У систему је дефинисано преко 100 библиотека које су доступне преко Z39.50 протокола. Програм је комерцијалан, међутим не пружа подршку за UTF-8 кодни распоред.

Кориснику овог система је омогућено да додаје нове библиотеке, такође може вршити промену параметара за неку библиотеку или је може избрисати из листе. Такође, од пунућених библиотеке могу се издвојити оне које припадају одређеној држави или су на пример одређеног типа. Приликом дефинисања параметара за појединачну библиотеку дефинишу се и *Use* атрибути из скупа *bib-1*, међутим постоји ограничен број места за унос атрибута и не могу се дефинисати сви атрибути из скупа *bib-1*.

Систем подржава употребу логичких оператора, али је број операнада ограничен на пет. Након извршене претраге појављује се прозор са резултатима. Прозор је подељен на два дела у првом делу су дати основни подаци о запису, а селектом једног од записа у другом делу прозора се појављује запис са свим подацима. Запис се може приказати у MARC 21 пуном формату или у облику каталошког листића.

EndNote [EndNote] представља алат за претрагу и преузимање библиографских записа. Он омогућава складиштење и обраду преузетих записа, али такође омогућава и креирање нових записа. Овде ће бити речи само о улози софтвера *EndNote* у претраживању.

EndNote је комерцијални софтвер који поред рада у Windows оперативном систему подржава и ради са MAC OS оперативним системом. Програм пружа и подршку за UTF-8 кодни распоред.

Приликом претраге кориснику се на почетку нуди да изабере базу, односно сервер на који жели да се конектује. Уколико не постоји тражена база у листи понуђених, корисник може креирати нову конекцију и при томе унети све параметре потребне за конекцију.

После успешне конекције на базу, корисник може изабрати неки од понуђених критеријума претраге. Такође, могуће је користити и логичке операторе AND, OR и NOT. Софтвер пружа још и једну додатну могућност која би могла да сузи претрагу, а то је специфицирање да ли на пример унети израз означава почетак неке друге речи и слично. Ови критеријуми донекле подсећају на остале атрибуте из скупа *bib-1*, мада софтвер нуди могућност дефинисања свих атрибута из датог скупа, али је тада потребно познавати синтаксу у којој се уносе ти атрибути, јер се уносе заједно са изразом за претрагу.

Софтвер пружа и могућност снимања постављеног упита, тако да је могуће у следећој претрази само учитати неки од снимљених упита.

BookWhere [BookWhere] је Z39.50 клијентска апликација за претраживање и добављање библиографских записа који су доступни преко Интернета. Ова апликација нуди могућност истовремене претраге више библиотека. *BookWhere* нуди подршку за UTF-8, па је могуће претраживати записе на било ком језику.

Приликом покретања апликације отвара се нова сесија и кориснику се нуди могућност избора библиотека које жели да претражи. Корисник може додати нову библиотеку и подесити одговарајуће параметре конекције уколико се тражена библиотека не налази у листи. За сваку библиотеку је могуће специфицирати атрибуте скупа *bib-1* које та библиотека подржава.

Кориснику су на располагању три врсте претраге: *Simple*, *Power* и *Batch*. *Simple,* односно једноставна претрага омогућава претраживање по понуђеним критеријумима који су међусобно повезани логичким оператором AND, и не могу се користити остали оператори. *Power* претрага представља напредно претраживање у коме је могуће користити остале логичке операторе, али се као подрезумевани оператор користи AND. Приликом креирања упита, упит се приказује на екрану и да би се променио оператор потребно је у упиту селектовати оператор који се жели променити. Овакав начин избора оператора, за корисника може бити донекле компликован. За разлику од једноставне претраге, код ове врсте претраге је за сваки изабрани *Use* атрибут могуће дефинисати и остале атрибуте из скупа *bib-1*.

Batch претрага нуди могућност да се за један изабрани атрибут, на пример ISBN унесе више вредности. Систем ће кориснику вратити поруку колико погодака има за сваку од унетих вредности.

У горњем делу прозора који се отвара након претраживања, дати су само основни подаци о пронађеном запису, док се селектовањем једног од тих

записа у доњем делу прозора појављује целокупан садржај записа. Пронађене записе је могуће сортирати по аутору, наслову, години издања и бази у којој је запис пронађен. Такође жељени запис се може експортовати, снимити у фајл систему, у неком од понуђених формата.

На основу детаљне анализе постојећих клијентских апликација у радовима [Boberić06, Boberić07a, Boberić07b, Boberić07c] дат је опис моделирања и имплементације новог XML едитора за преузимање библиографских записа.

1.1.2 Серверске апликације[1]

У свету постоји тренутно преко хиљаду регистрованих сервера који подржавају Z39.50 протокол и доступни су без икакве наплате за коришћење система. Табеле приказане у овом одељку преузете су из [Statistic]. У табели 1.1 је дат преглед земаља које омогућавају претраживање и добављање библиографских записа путем Z39.50 протокола, и као што се из табеле може видети на првом месту је Данска са 94 регистрована Z-сервера. Међутим, ово рангирање извршено је на основу домена који се налази у оквиру адресе Z-сервера, далеко већи број сервера (205) припада универзитетској мрежи библиотека широм света.

	Земље	Број Z-сервера (проценат)
1	Данска	94 (10%)
2	Русија	62 (7%)
3	Велика Британија	57 (6%)
4	Канада	48 (5%)
5	Америка	45 (5%)
6	Аустралија	33 (4%)
7	Шпанија	24 (3%)
8	Италија	12 (1%)

Табела 1.1. Земље које подржавају Z39.50 протокол

Z39.50 стандардом је дефинисан скуп сервиса који се користе приликом комуникације. При томе постоје обавезни сервиси које сваки сервер мора да подржи, а постоје и они који су опциони и чија реализација зависи од имлементације. У табели 1.2 дат је приказ у којој мери су сервиси и неки од

[1] Сви термини везани за стандард Z39.50 који се помињу у овом одељку детаљно су објашњени у трећем поглављу.

параметара сервиса имплементирани у постојећим Z- серверима. Оно што је и логично за очекивати је да су сервиси *Search* и *Present* имплементирани у свим серверима, јер они представљају сервисе који су обавезни. Сервиси наведени у табели 1.2 детаљно су описани у трећем поглављу.

	Сервиси	**Број Z-сервера (проценат)**
1	Search	925 (100%)
2	Present	925 (100%)
3	Scan	732 (79%)
4	namedResultSets параметар сервиса Search	651 (70%)
5	Delete	479 (52%)
6	Sort	367 (40%)
7	concurrentOperations параметар сервиса Init	289 (31%)
8	Extended-Services	286 (31%)
9	Resource Control	52 (6%)
10	Access Control	50 (5%)
11	Resource-report	47 (5%)
12	Trigger-Resource-Control	40 (4%)
13	Segment • level 1 Segmentation • level 2 Segmentation	 34 (4%) 21 (2%)

Табела 1.2. Преглед имплементираних сервиса

Приликом имплементирања Z-сервера искоришћени су различити софтверски алати. Преглед значајнијих алата са бројем сервера у којима су имплементирани дат је у табели 1.3.

Z39.50 представља клијент/сервер протокол и у претходном одељку су представљени неки од најчешће коришћених клијентских апликација приликом комуникације путем Z39.50 протокола. У овом одељку су приказани неки од најчешће коришћених серверских апликација.

	Назив алата	Број Z- сервера (проценат)
1	z39-innopac	246 (27%)
2	Zebra	210 (23%)
3	Voyager	112 (12%)
4	SIRSI CORP	105 (11%)
5	Geac Advance Z39.50 SERVER	28 (3%)
6	Horizon Information Portal Z39.50 Server	22 (2%)
7	RUSLAN Server	19 (2%)
8	DBC пер	12 (1%)
9	Library.Z	8 (1%)

Табела 1.3. Преглед алата искоришћени у имплементацији Z-сервера

Z39-innopac [INNOPAC] је производ компаније *Innovative Interfaces*, која је једна од водећих компанија у области пројектовања библиотечких информационих система. **Z39-innopac** је софтвер независтан од платформе и окружења и писан је у Јава програмском језику. **Z39-innopac** сервер подржава и верзију 2 и верзију 3 протокола Z39.50. Подржава скуп атрибута *bib* -1 и пружа могућност појединачним библиотекама да изврше мапирања атрибута на одговарајуће индексе у бази података на њима својствен начин.

Zebra [ZEBRA] је алат који је реализовала софтверска компанија *Index Data.* Овај алат спада у групу текст сервера и између осталог омогућава претраживање и добављање записа путем Z39.50 протокола. Алат је писан у језику C++ и постоје *open source* верзије за оперативне системе Windows и Unix.

Од сервиса дефинисани Z39.50 стандардом, **Zebra** подржава следеће сервисе: *Initialization*, *Search*, *Present*, *Segmentation*, *Delete*, *Scan*, *Sort*, *Close* и садржи *Extended Service* за додавање или замену постојећег записа који је у XML-у. Подржава упит типа-1 и типа-101, и подржава скуп атрибута *bib*-1. Конкретни детаљи о вредностима атрибута, које су подржане а које нису,

дати су у раду [ZEBRA]. Резултати претраживања се могу приказати у различитим нотацијама, као на пример GRS-1(Generic Record Syntax), SUTRS (Simple Unstructured Text Record Syntax), XML, ISO2709 (MARC).

Voyager LMS-Z39.50 Server [VOYAGER] је софтвер компаније Endeavor Information System и дизајниран је да задовољи првенствено потребе академских и истраживачких установа. Алат је писан у C++ програмском језику и омогућава комуникацију путем Z39.50 протокола. **Voyager** Z39.50 сервер подржава верзију 3 протокола Z39.50. Овај систем је уграђен у web портал Конгресне библиотеке у Вашингтону.

Подржан је упит типа-1 са одређеним подскупом скупа *bib*-1, детаљи о атрибутима и осталим параметрима налазе се на адреси [VOYAGER]. Сервер подржава следеће сервисе: *Initialization*, *Search* и *Present***.**

SIRSI CORP [SIRSI] је софтвер компаније *SirsiDynix*, која се првенствено бави развојем библиотечких информационих система. Поред обавезних сервиса које сваки Z39.50-сервер мора имплементирати овај сервер подржава и неке додатне сервисе (*Delete*, *Access-control, Resource-report, Resource-control, Trigger-resource-control*). Подржава скуп атрибута *bib* -1 и упите типа 1 и 101.

1.2 Библиотечки информациони систем БИСИС

Библиотечки софтверски систем БИСИС развија се од 1993. године. Тренутно је актуелна трећа верзија система. Систем је базиран на YUMARC формату [Vulić], који је настао из UNIMARC-а, COMARC-а и из захтева пројекта Библиотечка мрежа система научних и технолошких информација Србије. Развој и пројектовање овог система описан је у монографији [BISIS04].

У верзији 3.0 развијен је сопствени текст сервер за индексирање и претраживање библиографских записа у UNIMARC формату. Главне карактеристике овог сервера су: специјализованост која резултује бољим перформансама, трослојна архитектура, коришћење *Java* платформе и независност од коришћеног релационог система за управљање базом података. Подршка за *Unicode* стандард доследно је спроведена у целокупном систему БИСИС верзија 3.0.

1.2.1 Преузимање библиографских записа

У верзији 3.0. постоји могућност преузимања библиотечких записа, али је то преузимање ограничено само на библиотеке које користе систем БИСИС. За потребе размене библиотечких записа унутар система БИСИС имплементиран је протокол који је базиран на размени порука у виду XML

документа што је детаљно описано у радовима [Zarić03, Zarić04a, Zarić04b, Surla].

За имплементацију протокола искоришћена је ebXML спецификација, која у суштини јесте веб сервис, али управљан порукама. Оваква идеја је омогућила да се на страни сервера реализује једноставан интерфејс који обавља једну просту функцију пријема XML форматиране поруке, док се анализа саме поруке и њена обрада може реализовати у произвољним модулима апликације. Како су у овом случају и саме поруке које се размењују XML документи, то омогућава врло брзу и квалитену валидацију коректности размењених порука, јер је могуће прописати и формат самих порука путем XML шема.

1.2.2 Примена XML технологије у развоју система БИСИС

Примена XML технологија у развоју библиотечког софтверског система БИСИС обухвата следеће:

- Опис библиграфских формата и библиографских записа помоћу XML шема језика
- Контролу квалитета XML библиографских записа
- Употребу XML native технологија
- Генерисање каталошких листића на основу XML библиографских записа
- Имплементацију XML едитора за библиографске формате и библиграфске записе

Опис библиграфских формата и библиографских записа помоћу XML шема језика разматран је са два аспекта. Први је формирање XML шеме тако да посебно описује све појединачне елементе формата а други формирање XML шеме погодну за имплементацију библиотечког софтвера.
У радовима [Zeremski04, Zeremski01a, Zeremski01b] показано је да се помоћу XML шема језика могу моделирати сви концепти UNIMARC формата. Овако формирана XML шеме могла би се користити за валидацији изабраног дела UNIMARC формата за обраду библиографске грађе. Такође је показано да се помоћу XML шема језика могу моделирати сви концепти UNIMARC библиографских записа, као и сви елементи тих концепата. XML шема бибиографских записа описана је у радовима [Zeremski01c, Zeremski02, Zeremski01d]. Овако формирана XML шема могла би се користити за валидацију библиографских записа формираних

по UNIMARC формату као и за контролу квалитета библиографских записа.

Међутим, у радовима [Dimić07a, Milosavljević07] дат је предлог XML шема које описују концепте библиографских формата (YUMARC, UNIMARC, MARC21) као што су поља, потпоља, индикатори, шифарници, док појаве тих концепата садрже конкретне податке о формату, односно спецификацију поља, потпоља, индикатора и друго. Поред тога, описане су и XML шеме библиографских записа формираних по тим форматима. Овакве шеме омогућавају да се у софтверским системима ради само са изабраним појавама формата и записа за обраду библиографске грађе.

Контрола квалитета XML библиографских записа. Монографија [Budimir04] посвећена је контроли квалитета библиографских записа. Дефинисана је и систематизована контрола за утврђивање квалитета библиографских записа формираних по UNIMARC формату. Концепти контрола записа описани су помоћу XML шема језика. Контроле које се не могу описати XML шема језиком описане су помоћу XSLT спецификација. Имплементиран је систем за контролу квалитета XML библиографских записа у Java окружењу.

Употреба XML native технологија. Циљ истраживања приказаних у радовима [Škrbić04, Škrbić05] је испитивање могућности употребе XML native технологија у развоју библиотечког информационог система заснованог на UNIMARC формату. У складу са тим, урађено је моделирање и имплементација софтверског система за анализу и верификацију употребе XML native технологија за обраду библиографске грађе.

Генерисање каталошких листића на основу XML библиографских записа. У радовима [Rađenović06a, Rađenović06b] приказано је моделирање и имплементација софтверског пакета за формирање библиотечких каталошких листића. Формирање ових листића базирано је на софтверском пакету FreeMarker и XML документима библиографских записа. Архитектура система је таква да омогућује доступност каталошких листића из свих сегмената библиотечког софтверског система БИСИС и могућност ажурирања каталошких листића без поновног компајлирања изворног кода.

Имплементацију XML едитора за библиографске формате и библиграфске записе. Детаљан опис функционалности едитора за обраду библиографске грађе у актуелној верзији система БИСИС дато је у [BISIS03]. Поред едитора за обраду библиографске грађе у оквиру развоја система БИСИС вршено је и истраживање које се односи на моделирање и имплементацију едитора за UNIMARC формат. У тези [Mijić03] дат је

предлог моделирања и имплементације XML едитора за опис UNIMARC формата. У монографији [Škrbić05] приказано је моделирање и имплементација XML едитора који илуструје начин употребе XML native технологија у развоју библиотечког информационог система заснованог на UNIMARC стандарду. Као наставак ових истраживања, у овој монографији описан је едитор за претраживање и преузимање библиографских записа по протоколу Z39.50. Едитор је заснован на XML технологијама и упитни језици који се користе приликом претраживања моделирани су XML шема језиком.

Поглавље 2

Стандард Z39.50

Z39.50 (*ANSI/NISO Z39.50-2003, Information Retrieval (Z39.50): Application Service Definition and Protocol Specification)* је стандард организације NISO (*National Information Standards Organization*). Прихваћен је од стране ISO (*International Standards Organization*) као ISO 23950:1998, *Information and documentation- Information retrieval (Z39.50) - Application service definition and protocol specification*. Актуелна верзија стандарда је верзија 3 и датира из 1995. године. Стандард је осмишљен као општи стандард који се бави проблемима проналажења и преузимања података из удаљених база података, мада је његова најчешћа употреба у системима као што су библиотеке, универзитети и централни каталози. Агенција која се бави формалним дефинисањем, унапређивањем и одржавањем стандарда Z39.50 је *International Standard Z39.50 Maintainance Agency* у оквиру Конгресне библиотеке.

Стандард описује сервисе који се користе приликом претраживања и добављања информација и даје спецификацију одговарајућег протокола који се користи при клијент/сервер комуникацији и који подржава дате сервисе.

Сервиси описују активност између две апликације: апликације која иницира активност (клијента) и апликације која шаље одговор (сервера), мада постоје и сервиси у којима је иницијатор акције серверска страна протокола. Сервер је повезан са једном или више база података. Сервиси су подељени на процедуре које извршава клијент и процедуре које извршава сервер.

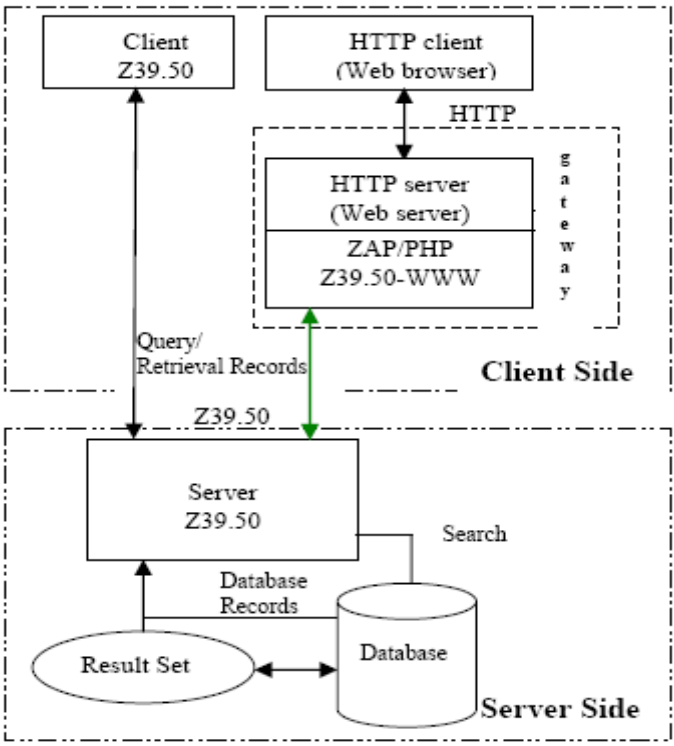
При спецификацији протокола дефинишу се правила за размену информација као и основни захтеви који се морају задовољити приликом имплементације протокола. Стандард не описује начин имплементације сервиса нити протокола у оквиру појединачног система.

Стандард је базиран на OSI моделу (*Open Systems Interconnection Basic Reference Model*) дефинисаног током 80-тих година. У оквиру OSI модела Z39.50 представља протокол на апликационом нивоу и од нижих слојева OSI модела Z39.50 захтева поуздан потпуно двосмеран бинарни транспортни протокол, као што је TCP. За спецификацију садржаја пакета података користи се апстрактна синтаксна нотација – *Abstract Syntax*

Notation One (ASN.1), а за серијализацију ASN.1 структура користе се *Basic Encoding Rules (*BER*)*

2.1 Z39.50 сервиси

На слици 2.1 која је преузета из [Pict1] приказана је архитектура система који учествују у комуникацији путем протокола Z39.50. На клијентској страни налази се кориснички интерфејс који омогућава крајњем кориснику система да постави упит и да манипулише са преузетим подацима. Z-клијент представља клијентску страну протокола и он може директно комуницирати са серверском страном. Међутим комуникацију је могуће остварити и преко система који се називају капије (*gateway*). Капија је програм који има два интерфејса. Овде су од значаја капије чији је бар један интерфејс Z39.50 клијент или сервер. У том случају крајњи корисник преко одређеног браузера комуницира са капијом која у себи поред web сервера садржи и Z-клијент. На слици је приказан софтвер ZAP/PHP који представља конкретну имплементацију Z-клијента и преко њега се врши комуникација са Z-сервером.



Слика 2.1 Клијент-сервер архитектура

На серверској страни између осталог налази се база података и комуникација између базе и система није дефинисана стандардом и то зависи од самог система. На серверској страни налази се Z-сервер и његов задатак је да реализује серверску страну протокола Z39.50. На серверској страни је потребно извршити и мапирање локалне базе података тако да ти подаци буду доступни Z-серверу. То значи да када Z-сервер прими упит од Z-клијента, тај упит се мора трансформисати тако да се може применити на локалну базу података. Такође пронађени подаци се морају прилагодити формату који је погодан за даљу обраду од стране Z-сервера. На слици 2.1 су ове трансформације приказане разменом порука између Z-сервера и базе података и компонентом *Result Set* која представља скуп податак пронађених у бази.

Између Z-клијента и Z-сервера успоставља се конекција, односно Z-асоцијација. Комуникација између Z-клијента и Z-сервера се одиграва разменом порука. Садржај и редослед порука дефинисан је сервисима који су груписани у структурне блокове - *facilities.* Порука може бити захтев (*request*) или одговор (*response*). Сервиси могу бити потврдни, непотврдни и условно-потврдни.

Потврдни сервис је такав сервис који има захтев и одговор. Код непотврдних сервиса постоји само захтев који могу да упуте или клијент или сервер, и не постоји порука о одговору. Условно-потврдни сервис се може посматрати и као потврдни и као непотврдни сервис у зависности да ли се у захтеву специфицира да је потребно проследити и одговор.

Z39.50 стандард дефинише 11 структурних блокова или механизама (*facilities*):

1. Механизам за иницијализацију (*Initialization Facility*)
   - *Init* сервис;
2. Механизам за претраживање (*Search Facility*)
   - *Search* сервис;
3. Механизам за добављање информација(*Retrieval Facility*)
   - *Present* сервис,
   - *Segment* сервис;
4. Механизам за брисање скупа погодака (*Result-set-delete Facility*)
   - *Delete* сервис;
5. Механизам за прегледање (*Browse* Facility)
   - *Scan* сервис;
6. Механизам за сортирање (*Sort Facility)*
   - *Sort* сервис,
   - *Duplicate Detection* сервис;
7. Механизам за контролу приступа (*Access Control Facilty)*

- *Access Control* сервис;

8. Механизам за контролу ресурса (*Accounting /Resource Control Facility*)
   - *Resource Control* сервис,
   - *Triggger-resource-control* сервис,
   - *Resource-report* сервис;
9. Механизам за објашњења (*Explain Facility*)
10. Допунски механизам (*Extended Services Facility*)
    - *Extended-services* сервис;
11. Механизам за прекид конекције (*Termination Facility)*
    - *Close* сервис.

Init сервис омогућава клијенту да успостави Z-асоцијацију. У *Init* захтеву клијент предлаже вредности за иницијалне параметре. У *Init* одговору сервер даје предлог својих вредности за иницијалне параметре, и те се вредности могу разликовати од оних које је предложио сервер. Ако сервер потврдно одговори на клијентов предлог Z-асоцијација се успоставља. Уколико клијент не жели да прихвати параметре које је понудио сервер, он може одустати од даље комуникације. Неки од параметара који се могу поставити су на пример: верзија Z39.50 протокола која се користи у комуникацији, максимална величина слога који ће сервер вратити клијенту, подаци о аутентификацији, подаци о имплементацији сервера/клијента.

Search сервис омогућава клијенту да дефинише упит који ће се применити на бази и да прими информације о резултату извршавања упита. Као одговор на клијентов захтев сервер креира скуп резултата који задовољава упит.

Present сервис омогућава клијенту да затражи приказ слогова који су добијени претходном претрагом. Тражени слогови (записи) су груписани у оквиру скупа резултата (*result set*) и имају релативну позицију у оквиру тог скупа. На овај начин је обезбеђено да клијент може тражити један конкретан запис или да дефинише опсег из ког жели преузети записе.

Уколико слог који је захтеван преко *Present* сервиса превазилази величину слога која је договорена у фази иницијализације тада сервер може вратити слог распоређен у више сегмената. Слање оваквих сегмената обезбеђено је *Segment* сервисом.

Delete сервис пружа могућност клијенту да од сервера захтева брисање одређеног скупа резултата или брисање свих скупова резултата који су формирани током Z-асоцијације. Овај сервис може бити веома користан у случају када сервер има ограничен број скупова које може чувати у меморији. Када број скупова резултата пређе дозвољену границу сервер ће

по неком свом алгоритму обрисати неке од скупова, тако да уколико неки од тих обрисаних скупова поново затребају клијенту они више неће бити доступни. Због тога је боље решење да клијент сам избрише скупове резултата који су му непотребни.

Scan сервис представља варијанту претраживања. Омогућава претраживање у оквиру уређене индексиране листе термова (тзв. индекс). Клијент треба да наведе префикс, на пример префикс може бити наслов дела, и његову вредност, а сервер може вратити број погодака, пронађене терме или назив базе у којој се терм јавља и томе слично. На пример, ако корисник жели да погледа у којим све варијантама се јавља реч „Моцарт“ у наслову, он може искористити овај сервис и добиће листу термова као што су: „Моцарт и Бетовен“, „Моцарт у Прагу“, “Моцартове синфоније“ и томе слично. Резултат ове операције су само индексирани изрази, тако да би се добио целокупан запис, потребно је позвати сервис *Search*. Показано је да је боље прво позвати сервис *Scan*, па затим извршити претраживање јер се на тај начин смањује могућност да се добију неочекивани резултати.

Sort сервис омогућава клијенту да од сервера захтева сортирање скупа резултата. Клијент дефинише правила за сортирање и слогови унутар скупа резултата треба да буду уређени према тим правилима.

Duplicate Detection сервис дозвољава клијенту да од сервера затражи да анализира један или више скупова резултата у циљу проналажења слогова који су дупликати. Клијент дефинише правила за препознавање дупликата.

Access-control сервис омогућава серверу да провери клијента. Под провером се подразумева да сервер, уколико је то договорено у фази иницијализације, у сваком тренутку може послати клијенту захтев у коме тражи од клијента да се идентификује. Уколико клијент одговори на начин који је прихватљив серверу комуникација се наставља, у супротном сервер може одбити да изврши операцију која је била у току пре него што је дошло до слања *Access-control* захтева или може затворити Z-асоцијацију.

Resource-control сервис служи за извештавање о стању ресурса. Овај сервис увек иницира сервер. На пример, сервер може обавестити клијента да ће доћи до прекорачења ресурса приликом извршавања операције и од клијента може захтевати потврду о наставку започете операције.

Trigger-resource-control сервис позива клијент као део неке друге активне операције. Овај сервис омогућава клијенту да затражи од сервера извршавање *Resource-control* сервиса или прекидање неке операције. Сервер није обавезан да одговори на овакав клијентов захтев због тога и не

постоји *Trigger-resource-control* одговор, већ само *Trigger-resource-control* захтев.

Resource-report service омогућава клијенту да затражи Resource-report који ће се односити на одређену завршену операцију или целокупну Z-асоцијацију. *Resource-report service* се разликује од *Trigger-resource-control* сервиса по томе што је *Trigger-resource-control* непотврдан сервис и захтева извештај који се односи на тренутно активну операцију, док је *Resource-report* потврдни сервис и извештај који се захтева се односи на операцију која је завршена.

Explain facility не садржи ни један сервис и омогућава клијенту да види детаље о имплементацији сервера: опште могућности (опис, контакт информације, радно време, ограничења, цену коришћења), које су базе података на располагању, детаље о атрибутима и скуповима атрибута, синтаксу слогова. Све ове информације чувају се на серверу у посебној бази којој клијенти могу приступити преко Z39.50 протокола.

Extended Services сервис служи за иницирање посебних додатних задатака који се извршавају ван сесије Z39.50 и чији се напредак може пратити кроз Z39.50 сервисе. Неки од ових специфичних сервиса су: чување скупа резултата, подешавање периодичних упита, експортовање докумената, ажурирање базе података.

Close сервис омогућава клијенту или серверу да оконча све операције које су у току и да иницира затварање Z-асоцијације.

Стандардом су следећи сервиси дефинисани као обавезни приликом имплементације протокола Z39.50: *Init*, *Search Present* и *Close* сервис, док су остали сервиси опциони и зависе од имплементације.

2.2 Формат записа

Приликом слања захтева да се преузме одређени запис, Z-клијент може специфицирати синтаксу у којој жели да преузме запис. У ту сврху у оквиру Z39.50 користи се концепт синтаксе записа - *Record Syntaxes*. На овај начин клијент може од сервера захтевати да се резултати врате у тачно одређеном формату специфицираном одређеном синтаксом. Уколико је сервер у стању извршити овакав захтев, резултати ће бити враћени у захтеваном формату. Међутим, уколико сервер не подржава захтевану синтаксу, могуће је да се резултати врате и у некој другој синтакси коју сервер одреди. Ово такође може довести до смањења интероперабилности различитих софтверских имплементација.

Z39.50 стандардом подржане су различите синтаксе које су наведене на адреси [RECS]. У наставку је дат списак синтакси које се најчешће користе приликом преузимања библиографских записа.

- MARC (**MA**chine-**R**eadable **C**ataloging)
  - ❖ Unimarc
  - ❖ Marc21
  - ❖ Остале врсте MARC синтакси (Canmarc, Ukmarc, Danmarc, Normarc...)
- GRS-1 (General Record Structure)
- SUTRS (Simple Unstructured Text Record Syntax)
- OPAC

MARC - Представља формат за складиштење библиографских записа, који је развијен у оквиру Конгресне библиотеке у периоду 1965-1966. Неке друге националне библиотеке су следиле пример Конгресне библиотеке и креирале сличне стандарде прилагођене својим потребама. Тако се у Великој Британији развио UKMarc [UKMARC], у Америци USMarc [USMARC], у Канади Canmarc [CANMARC], у Данској Danmarc [DANMARC], Unimarc (Universal Machine Readable Cataloguing) [UNIMARC] у другим европским земљама. На основу формата USMarc и Canmarc 1999. године настао је формат Marc21 [MARC21]. Сви наведени MARC формати се размењују у виду ISO2709 формата за размену [ISO2709] док приказ може зависити од имплементације клијента. Листинг 2.1 представља приказ једног библиографског записа у Marc21 пуном формату (*full format*) који је преузет из Конгресне библиотеке.

```
00812n  2200157z  4504
037  $f BIBLIOGRAPHY OF MAINE GEOLOGY
035  $a ESDD0062
040  $a MAINE GEOLOGICAL SURVEY
245  $a BIBLIOGRAPHY OF MAINE GEOLOGY
270  $p MAINE GEOLOGICAL SURVEY
     $a MAINE GEOLOGICAL SURVEY,STATE HOUSE STATION 22
     $b AUGUSTA
     $d USA
     $k (207) 289-2801
270  $p ROBERT MARVINNEY
     $p MAINE GEOLOGICAL SURVEY
     $a MAINE GEOLOGICAL SURVEY,STATE HOUSE
        STATION 22
     $b AUGUSTA
     $d USA
$k (207) 289-2801
506  $a NONE
513  $b 1692-PRESENT
520 $a This data base is a computer based bibliography  of
       marine. It allows searching by topic and geografic
       location, similar to GEOREF. It is currently under
       development to replace the printed Bibliography of
       Marine Geology.
540  $a UNDER DEVELOPMENT
710  $a MAINE GEOLOGICAL SURVEY
```

Листинг 2.1. Пример библиографског запис у Marc21 пуном формату

GRS-1 - Ова синтакса има структуру стабла које хијерархијски представља запис. Сваком чвору у стаблу додељен је уређени пар *(t,v)* који припада одређеном скупу (*tagSet*) где *t* представља тип скупа, а *v* је вредност из тог скупа [Tag]. Z39.50 стандардом дефинисана су два таква скупа: *Tagset-G* и *Tagset-M*. Ради идентификације скупова додељене су им нумеричке вредности, тако да скупу *Tagset-G* одговара вредност 2, а скупу *Tagset-M* вредност 1. Такође разне организације су за своје потребе развиле и нове скупове као што је на пример *GILS Tagset* [GILS] који има нумеричку вредност 3. *Tagset-G* даје информације о ресурсу, а *Tagset-M* садржи информације о запису који описује дати ресурс. Тако на пример, ако је изабран *Tagset-G* и вредност 20 која означава језик дела тада ће се у чвору (2,20) чувати информација о језику на ком је књига написана. Са друге стране ако је изабран *Tagset-M* и из тог скупа је изабрана вредност 22 која означава језик записа, тада ће се у чвору (1,22) налазити информација на ком језику је извршена каталогизација. Пример једног записа у GRS-1 формату дат је на листингу 2.2, где је на пример чвором (2,1) дефинисан наслов дела (BIBLIOGRAPHY OF *MAINE GEOLOGY*).

```
(1,1) OID: GILS-schema
(2,1) BIBLIOGRAPHY OF MAINE GEOLOGY
(4,52) MAINE GEOLOGICAL SURVEY
(3,Local-Subject-Index) GEOLOGY; MAINE
(2,6)
    (1,19) This data base is a computer based bibliography of marine geology.  I
t allows
searching by topic and geographic location, similar to GEOREF.  It is currently

under development to replace the printed Bibliography of Marine Geology.
    (3,Format) BIBLIOGRAPHIES
    (3,Comments) Software developed by the Maine Geological Survey; documentatio
n developed by
the Maine Geological Survey staff and written in ANSI COBOL.
(4,71)
    (3,Geographic-Coverage) 7 1/2' OR 15' QUADS; CITIES; US STATE; COUNTIES
    (3,Coverage-Description) MAINE
    (4,91)
        (4,9) -71
        (4,10) -67
        (4,11) 48
        (4,12) 43
(4,93)
    (4,16) 1692-PRESENT
(4,70)
    (4,90)
        (2,10) MAINE GEOLOGICAL SURVEY
        (4,2) MAINE GEOLOGICAL SURVEY, STATE HOUSE STATION 22
        (4,3) AUGUSTA
        (3,State) ME
        (3,Zip-Code) 04333
        (2,16) USA
        (2,14) (207) 289-2801
    (4,7) BIBLIOGRAPHY OF MAINE GEOLOGY
    (4,8)
        (3,Data-Set-Type) AUTOMATED
        (3,Access-Method) INTERACTIVE
        (3,Number-of-Records) 3,500
        (3,Bytes-Per-Record) 768
        (3,Computer-Type) BURROUGHS B20
        (3,Computer-Location) AUGUSTA, ME.
(4,53)
    (3,Documentation) NONE
(4,54)
    (3,Status) UNDER DEVELOPMENT
(4,94)
    (2,7) ROBERT MARVINNEY
    (2,10) MAINE GEOLOGICAL SURVEY
    (4,2) MAINE GEOLOGICAL SURVEY, STATE HOUSE STATION 22
    (4,3) AUGUSTA
    (3,State) ME
    (3,Zip-Code) 04333
    (2,16) USA
    (2,14) (207) 289-2801
(4,1) ESDD0062
(4,19) MAINE GEOLOGICAL SURVEY
(1,16) 198709
```

Листинг 2.2. Пример библиографског запис у GRS-1 формату

SUTRS - Ова синтакса омогућава преузимање запис путем протокола Z39.50 у виду ASCII текста. Намена ове синтаксе је да прикаже запис у формату који ће бити читљив крајњем кориснику а да при томе корисник не мора да врши додатне трансформације записа како би га прилагодио за приказивање. Текст се састоји од низа карактера где после сваког 72 карактера следи ознака за нови ред. На листингу 3 је приказан исти запис који је представљен и у листингу 1, али у облику SUTRS синтаксе.

```
gils:
  title: BIBLIOGRAPHY OF MAINE GEOLOGY
  originator: MAINE GEOLOGICAL SURVEY
  Local-Subject-Index: GEOLOGY; MAINE
  abstract: This data base is a computer based bibliography of marine
    geology. It allows searching by topic and geographic location, similar
    to GEOREF. It is currently under development to replace the printed
    Bibliography of Marine Geology.
    Format: BIBLIOGRAPHIES
    Comments: Software developed by the Maine Geological Survey;
      documentation developed by the Maine Geological Survey staff and
      written in ANSI COBOL.
  spatialDomain:
    Geographic-Coverage: 7 1/2' OR 15' QUADS; CITIES; US STATE; COUNTIES
    Coverage-Description: MAINE
    boundingCoordinates:
      westBoundingCoordinate: -71
      eastBoundingCoordinate: -67
      northBoundingCoordinate: 48
      southBoundingCoordinate: 43
  timePeriod:
    timePeriodTextual: 1692-PRESENT
  availability:
    distributor:
      organization: MAINE GEOLOGICAL SURVEY
      streetAddress: MAINE GEOLOGICAL SURVEY, STATE HOUSE STATION 22
      city: AUGUSTA
      State: ME
      Zip-Code: 04333
      country: USA
      phoneNumber: (207) 289-2801
    resourceDescription: BIBLIOGRAPHY OF MAINE GEOLOGY
    technicalPrerequisites:
      Data-Set-Type: AUTOMATED
      Access-Method: INTERACTIVE
      Number-of-Records: 3,500
      Bytes-Per-Record: 768
      Computer-Type: BURROUGHS B20
      Computer-Location: AUGUSTA, ME.
  accessConstraints:
    Documentation: NONE
  useConstraints:
    Status: UNDER DEVELOPMENT
  pointOfContact:
    name: ROBERT MARVINNEY
    organization: MAINE GEOLOGICAL SURVEY
    streetAddress: MAINE GEOLOGICAL SURVEY, STATE HOUSE STATION 22
    city: AUGUSTA
    State: ME
    Zip-Code: 04333
    country: USA
    phoneNumber: (207) 289-2801
  controlIdentifier: ESDD0062
  recordSource: MAINE GEOLOGICAL SURVEY
  dateOfLastModification: 198709
```

Листинг 2.3. Пример библиографског запис у SUTRS формату

OPAC - Ова синтакса је осмишљена у циљу симулације каталошког листића. Поред основних података који се односе на сам библиографски запис синтакса укључује и додатна поља која се односе на податке о локацији, сигнатури и податке везане за циркулацију (*Data holdings*).

```
01268ckm  22003377a 4500
001 11701275
005 20021110133511.0
007 kh|||
007 cr |||
008 970811s1955    xx nnn             kn
035    $9 (DLC)   97514049
035    $a (DLC)11701275
906    $a 7 $b cbc $c orignew $d u $e ncip $f 19 $g y-printpho
955    $a qw05
010    $a    97514049
037    $a LC-USZ62-119279 $b DLC $c (b&w film copy neg.)
040    $a DLC $c DLC $e gihc
050 00 $a LOT 7399  $b <item>
245 00 $a [Act I, Scene 2 of "Hamlet," showing (l to r), Claudius, Gertrude, Ham
let and Polonius, Mayakovsky Theater, Moscow, Russia] $h [graphic].
260    $c [1955]
300    $a 1 photographic print.
500    $a No. 1.
600 10 $a Shakespeare, William, $d 1564-1616. $t Hamlet.
650  7 $a Theatrical productions $z Russia (Federation) $z Moscow $y 1950-1960.
$2 lctgm
655  7 $a Photographic prints $y 1950-1960. $2 gmgpc
852    $a Library of Congress $b Prints and Photographs Division $e Washington,
D.C. 20540 USA $n dcu
856 41 $3 b&w film copy neg. $d cph $f 3c19279 $u http://hdl.loc.gov/loc.pnp/cph
.3c19279
953    $a TE01
985    $a pp/cph
991    $b c-P&P $h LOT 7399  $i <item> $v obj. $w MUMS VM File
991    $b c-P&P $h LC-USZ62-119279 (b&w film copy neg.) $w MUMS VM File

Data holdings 0
typeOfRecord: u
encodingLevel: u
receiptAcqStatus: 0
generalRetention: |
completeness: 0
dateOfReport: 000000
nucCode: c-P&P
localLocation: Prints & Photographs Reading Room (Madison, LM337)
callNumber: LOT 7399 <item>
enumAndChron:  0 obj.
volume 0
 enumeration: obj.
 enumAndChron: obj.
circulation 0
 availableNow: 1
 itemId: 11800121
 renewable: 0
 onHold: 0
 enumAndChron: obj.
Data holdings 1
typeOfRecord: u
encodingLevel: u
receiptAcqStatus: 0
generalRetention: |
completeness: 0
dateOfReport: 000000
nucCode: c-P&P
localLocation: Prints & Photographs Reading Room (Madison, LM337)
callNumber: LC-USZ62-119279 (b&w film copy neg.)
circulation 0
 availableNow: 1
 itemId: 11800122
 renewable: 0
 onHold: 0
```

Листинг 2.4. Пример библиографског записа у OPAC формату

Листинг 2.4 представља библиографски запис дат у виду OPAC формата, где је део који се односи на запис дат у виду Marc21 пуног формата, али постоје и додатне информације које се односе на библиографску јединицу.

Ако у систему постоји више примерака за једну књигу онда се за сваки примерак креира посебан део који садржи информације везане за тај конкретни примерак. Тако се на листингу 2.4 може уочити да за дати наслов постоје два примерка, јер постоје два елемента *Data holding*. У оквиру овог елемента налази се информација о томе да ли је ово серијска публикација и ако јесте постоје подаци о волуменима, затим датум креирања записа, сигнатура, податак о локацији, податак да ли је примерак доступан или не и томе слични подаци.

2.3 Упитни језици

Један од основних задатака који је дефинисан овим стандардом је претрага база података. Стандард дефинише шест врста упита:

1. Тип-0 се може користити ако постоји претходни договор ван оквира стандарда.
2. Тип-1 представљен је обрнутом пољском нотацијом и стандардом Z39.50 је дефинисан као обавезан тип упита**.**
3. Тип-2 је упит дефинисан у стандарду ISO8777.
4. Тип-100 је упит познат под називом *Common Command Language* и није дефинисан у оквиру стандарда Z39.50.
5. Тип-101 представља проширење упита Тип-1. Овај упит је сличан Типу-1, само што не зависи од верзије протокола Z39.50.
6. Тип-102 (*Ranked List Query*) треба да буде дефинисан у наредним верзијама овог стандарда

Поред основних сервиса које клијент/сервер треба да задовољи, стандард условљава имплементаторе Z39.50 протокола да морају да подрже упит типа-1. Међутим, ово не значи да морају бити подржани сви оператори или све могућности за операнде. У овом одељку биће описана основна правила за формирање упита типа-1. Детаљи везани за овај тип упита и остале упите дефинисане овим стандардом биће описани у трећем поглављу.

Упит типа-1 се састоји од више појединачних термова који се траже, и за сваки терм се дефинишу одређени атрибути**.** У упиту типа-1 више термова може бити повезано логичким операторима AND, OR и NOT. Терми и оператори су изражени у обрнутој пољској нотацији (постфиксни записи).

2.4 Скупови атрибута

У упиту атрибути асоцирани с термом који се тражи припадају одређеном скупу атрибута. Дефиниција скупа атрибута је регистрована, тј. додељен јој

је јединствен, глобално препознатљив идентификатор скупа атрибута. Сваки атрибута представља уређени пар, где је први елемент пара тип атрибута, а други елеменат конкретна вредност за тај тип атрибута. За сваки тип атрибута постоје предефинисане вредности које атрибут може имати. Улога јединственог идентификатора скупа атрибута је да омогући разликовање оваквих уређених парова који могу имати исте вредности али припадати различитим скуповима атрибута.

За различите системе развијени су и различити скупови атрибута. Тако на пример, постоје следећи скупови атрибута: STAS-1 (Scientific and Technical Attribute Set), који је намењен за претраживање техничких и научних информација, CIMI-1 који је намењен за претраживање музејске грађе, bib-1 који специфицира разне атрибуте који су корисни за претраживање библиотечких система и многи други који су дати на адреси [AttributeSets]. У наставку је детаљније објашњен скуп атрибута bib -1.

Za мапирање атрибута ка логичком дизајну базе података задужен је сервер који имплементира Z39.50 протокол. Један од проблема код употребе скупова атрибута је да различити имплементатори Z-сервера различито интерпретирају дате атрибуте. На овај начин умањује се интероперабилност, која је и основна предност овог стандарда.

2.4.1 Скуп атрибута Bib-1

Скуп атрибута *bib-1* [Bib1] развијен је од стране библиографске заједнице и користи се при формирању упита типа-1. Bib-1 скуп атрибута састоји се од шест типова атрибута:

- *Use Attributes*
- *Relation Attributes*
- *Truncation Attributes*
- *Structure Attributes*
- *Completeness Attributes*
- *Position Attributes*

Use атрибут - Дефинише улазне критеријуме претраге (наслов, аутор, одредница, итд). Да би се избегла различита тумачења атрибута Конгресна библиотека је дефинисала правила за мапирање Use атрибута на одговарајућа поља Marc 21 формата. У Прилогу 1 у табели 1, дат је преглед Use атрибута и списак поља односно потпоља по којима се претражује када се изабере одређени атрибут.

Ако се на пример за Use атрибут изабер вредност 4, то значи да је изабрана вредност која означава наслов и та вредност се приликом претраживања тражи у пољима: 130, 21X -24X, 440, 490, 730, 740, 830, 840 и у потпољу $t у следећим пољима: 400, 410, 410, 600, 610, 611, 700, 710, 711, 800, 810, 811.

Ознака X приликом означавања поља, као што је то у случају 21X -24X значи да ће се претраживати по свим пољима (из задатог опсега) која на почетку имају задате две цифре а трећа цифра је опциона. На конкретном примеру то би значило да ако корисник жели да пронађе књигу са насловом „*На Дрини ћуприја*“ он би изабрао вредност 4 за Use атрибут и добиће записе које у неком од наведених поља садрже тражени израз.

Relation атрибут - Дефинише релације између критеријума претраге (мање, веће, једнако, итд). У табели 2 (Прилог 1) дат је списак свих атрибута типа *Relation* који дефинишу везу између атрибута типа *Use* и одговарајућег терма. На пример, ако се жели истаћи да се траже књиге код којих је година издања мања од израза који је наведен у одговарајућем Use атрибуту користи се вредност 1 за *Relation* атрибут и она означава релацију мање.

Truncation атрибут - Дефинише који део вредности критеријума претраге ће бити коришћен током претраге (почетак речи, крај речи, итд). Списак вредности типа атрибута *Truncation* дат је у табели 3 (Прилог 1). Улога овог атрибута може се упоредити са џокер знацима који се користе у стандардним претраживањима. Овај тип атрибута дефинише да ли ће један или више карактера из терма који се тражи бити изостављени.

Structure атрибут - Дефинише формат терма који се тражи, као на пример, реч, фраза, датум и слично. У табели 4 (Прилог 1) приказане су вредности атрибута Structure. Уколико се, на пример израз „Лелејска гора“, жели да се тумачи као фраза а не као листа речи које могу бити у произвољно редоследу тада ће се изабрати вредност 1 за *Structure* атрибут која означава фразу.

Position атрибут - Дефинише где се у оквиру поља налази тражени термин. Атрибутом *Position* дефинише се локација терма који се тражи у оквиру поља или потпоља у којима се терм јавља. У табели 5 (Прилог 1) приказане су вредности атрибута *Position.*

Completeness атрибут - Дефинише да ли термин за претрагу мора или не мора бити једина вредност у датом пољу/потпољу. У табели 6 (Прилог 1) дате су вредности атрибута *Completness*. Овај тип атрибута дефинише да ли терм који се тражи представља целокупан садржај поља, односно потпоља или не. Атрибут Completeness специфицира да ли се и неке друге речи осим тражених налазе у пољу/потпољу.

Временом се јавила појачана потреба за коришћења Z39.50 стандарда, и појавила се тенденција да се уместо дефинисања потпуно нових скупова атрибута, врши додавање нових *Use* атрибута у *bib*-1 скуп. Управо ово оптерећивање *Use* атрибута, као и све веће преклапање *bib*-1 скупа

атрибута са оним скуповима који су ипак формирани за посебне потребе, утицало је на потребе реструктурирања система. Нова архитектура атрибута би се састојала од основних атрибута (*core Attribute Set*) који су заједнички свим апликацијама и које би користиле све имплементације Z39.50 протокола, без обзира на област примене, док би се додатне функционалности, карактеристичне за одређене регионе и/или области примене обезбеђивале кроз додатне скупове атрибута који се не би преклапали са овим основним.

2.5 Профили

Концепт профила осмишљен је за потребе претраживања различите врсте докумената. Њима се дефинишу правила за коришћење стандарда Z39.50. Тако на пример, дефинише по којим префиксима ће се претраживати, који скуп атрибута ће бити коришћен, затим у ком формату ће се записи преузимати, који сервиси ће бити подржани и слично.

Различити профили су дефинисани за различите области примене укључујући профиле за геопросторне намене, профиле специфичне за државну управу, профиле за библиотечке системе и друге. Ови профили се врло често формирају у интересним групама, а агенција за стандардизацију и одржавање Z39.50 се појављује у улози саветодавног и надзорног тела које такве профиле уводи у ширу употребу, тј. публикује их као усаглашене профиле.

Неки од дефинисаних профила су:

- GEO - Амерички национални профил за рад са географским подацима.
- CIMI (***C****omputer* ***I****nterchange of* ***M****useum* ***I****nformation*) - поред преноса текста дефинише се и начин преузимања слика.
- Z+SQL - представља надоградњу стандарда Z39.50 са елементима SQL-a.
- Национални профили (Фински Z39.50 профил, Дански Z39.50 профил, Тексашки профил) - ови профили специјализовани су за потребе библиотека одређене земље.
- Bath - Ово је најчешће коришћени профил у библиотекарству. У њему је обухваћено неколико других профила и на тај начин је покушано да се отклоне недостаци претходних профила (различите интерпретације профила од стране различитих имплементатора, застарелост, неекономичност, недостатак интероперабилности). Детаљи везани за овај профил дати су на адреси [Bath].

2.6 Остали стандарди

Z39.50 представља доста сложен и компликован стандард те је из тих разлога основана организација ZING (Z39.50 ***I**nternational **N**ext **G**eneration*) која је предложила поједностављивање стандарда. На тај начин би концепти Z39.50 били боље искоришћени и имлементација сервиса би била поједностављена**.** Предвиђена је употреб модерних технологија као што су XML који би се користио за приказ и енкодинг података, транспорт података би се вршио преко HTTP-а, а интеракција између клијента и сервера би се одвијала преко SOAP порука.

У оквиру ZING иницјативе развило се више различитих стратегија које су засноване на Z39.50: SRW/U, ZOOM, ZeeRex, ez39.50.

SRW/U (*Search & Retrieve Web Service/ Search & Retrieve URL Service*) - је заснован на Z39.50 стандарду и web технологијама. Комбинује више Z39.50 сервиса у виду операција дефинисаних у оквиру WSDL документа. Комплетан листинг WSDL фајла SRW сервиса дат је на адреси [SRWWSDL]. За пренос података и енкодинг користи се XML. Претраживање и преузимање података може се обављати било путем SOAP протокола (SRW), било путем приступа стандардним URL (SRU). Ови сервиси омогућавају кориснику да претражује удаљене базе. Корисник шаље *searchRetrieve* упит који у себи садржи критеријум претраге, и од сервера добија *searchRetrieve* одговор који садржи информације о броју нађених резултата, и евентуално и саме записе у XML формату који је корисник захтевао. Такође је дефинисан посебан упитни језик *Common Query Language* (CQL) који омогућава писање људски читљивих, интуитивних упита. Овај језик подржава веома једноставне упите, али поседује и експресивност сложенијих језика, која обезбеђује могућност постављања произвољно сложених упита.

ZOOM (*The **Z**39.50 **O**bject-**O**rientation **M**odel*) - Представља апстрактни објектно-оријентисани API за подскуп сервиса дефинисаних Z39.50 стандардом. Детаљи везани за API дати су на адреси [ZOOM]. Ово је абстрактан API али постоје имплементације за неке програмске језике као што су C++, Јава, .Net, Perl, Phyton, Vbasic.

ZeeRex (*Z39.50 **E**xplain, **E**xplained and **R**e-**E**ngineered in **X**ML*) - је имплементација *Explain Facility* из стандарда Z39.50, чија улога је да клијенту пружи потребне техничке податке о серверској страни протокола Z39.50, као што су на пример: адреса на којој се налази сервер, назив базе података, који типови синтаксе записа су подржани и томе слично. Потребне информације о серверској стране су упаковане у XML документ и као такве се прослеђују клијенту. Обично се овај софтвер користи у

комбинацији са SRW/U протоколом. Детаљан опис овог система дат је на адреси [ZeeRex] .

ez39.50 (*Simple Implementation of Z39.50 over SOAP using XER*) - Представља начин да се протокол Z39.50 који је описан ASN-1 синтаксом опише помоћу XML нотације и да се тако формиране поруке преносе путем SOAP-а. За потребе транспорта порука и њихову серијализацију искоришћен је XER (*XML Encoding Rules*) [XER], који обезбеђује механизам помоћу кога је могуће обезбедити подршку за Z39.50 путем било ког интернет протокола, а да при томе није неопходно мењати тренутно постојећи ASN-1 стандард.

Поглавље 3

Упитни језици

У другом поглављу је речено да је Z39.50 стандардом дефинисано више различитих типова упита, па су у овом поглављу објашњени неки од типова упита. У пракси се највише користи упит типа-1, јер је он обавезан приликом имплементације Z39.50 протокола. Неки од Z39.50 сервера такође подржавају упит типа-101, који представља уопштење упита типа-1. Упит типа-101 је независтан од верзије имплементације Z39.50 протокола, тренуто су актуелне верзија 2 и верзија 3, док упит типа-101 је стриктно везан за верзију 3. У досадашњем истраживању на овом раду нису пронађени Z39.50 сервери који пружају могућност претраживања помоћу осталих типова упита.

3.1 Z39.50 упит типа -1

Z39.50 стандардом дефинисан је одређен скуп обавезних захтева које сваки имплементатор овог протокола мора да подржи. Међу тим обавезним захтевима налази се и имплементирање упита типа-1.

3.1.1 Синтакса упита

Синтакса упита дата је помоћу Бекус-Наурове форме и има следећу структуру:

RPN-Query ::= Argument | *Argument + Argument Operator*

Argument ::= Operand | *RPN-Query*

Operand ::= AttributeList + Term | *ResultSetId* | *Restriction*

Restriction ::= ResultSetId + AttributeList

Operator ::= AND | *OR* | *AND-NOT* | *Prox*

Упит се може састојати од једног операнда или више операнада повезаних логичким операторима. Операнди у суштини представљају скуп слогова који припадају бази података и за операнд постоји више могућности:

1) *AttributeList + Term* - у овом случају операнд представља скуп који се добија применом скупа одређених атрибута и одговарајуће вредности терма за те атрибуте.

2) *ResultSetId* - овај операнд означава именовани скуп резултата који је добијен у некој ранијој претрази.
3) *Restriction* - овај операнд представља скуп резултата који су добијени тако што је на неки именовани скуп примењена рестрикција са одређеним атрибутима. На пример, ако имамо неки именовани скуп резултата *скуп1* у коме се реч Хамлет појављује у:
   - наслову,
   - наслову и у аутору,
   - наслову, аутору и одредници.

   тада ће скуп резултата које добијемо после примене рестрикције са атрибутом *аутор* обухватати само слогове који припадају случају б) и в). Овај операнд је дефинисан само за верзију 3 протокола Z39.50.

Оператори који се користе при формирању упита су бинарни оператори AND, OR, AND-NOT и Prox. Оператори AND и OR су стандардни логички оператори, док оператор AND-NOT означава везу између два операнда при чему се врши негација другог операнда. Према томе ако постоје два операнда A и B тада A AND-NOT B означава све слогове који задовољавају операнд A и не задовољавају операнд B. Оператор Prox даје информацију о удаљености два операнда, на пример да ли су у истом параграфу и томе слично.

Ако претпоставимо да имамо два операнда O_1 и O_2 и ако респективно са S_1 односно са S_2 означимо скупове који они представљају тада:

а) O_1 AND O_2 - представља пресек скупова S_1 и S_2

б) O_1 OR O_2 - представља унију скупова S_1 и S_2

в) O_1 AND-NOT O_2 - представља све слогове који припадају скупу S_1 и не припадају скупу S_2

г) O_1 Prox O_2 -

- ако су оба операнда типа 1) тада је ово подскуп од скупа који се добија применом O_1 AND O_2 и важи да је O_1 ProxTest O_2 тачно
- у осталим случајевима сервер мора обезбедити додатни скуп резултата или ће доћи до грешке.

Prox оператор служи да ближе одреди везе између два операнда и параметри које *ProxTest* може имати су следећи:

- *Distance*: Означава удаљеност између два операнда. На пример, ако је вредност параметра *Unit* параграф и ако је вредност параметра *Distance*

нула, то означава да се оба операнда морају наћи у истом параграфу. Вредност за овај параметар не може бити негативна.

- *Relation*: Означава релацију између два операнда. Вредност за овај параметар могу бити мање, мање или једнако, једнако, веће, веће или једнако или неједнако.
- *Unit*: Вредности које може имати овај параметар су на пример карактер, реч, реченица, параграф, одељак, поглавље, документ или нешто што је дефинисано ван ових могућности.
- *Ordered flag*: Ако је постављен то значи да се леви операнд мора налазити пре десног операнда.
- *Exclusion flag*: Ако је овај параметар постављен то значи да треба да се примени негација *ProxTest*.

Употреба овог оператора биће разумљивија на конкретном примеру. Претпоставимо да *A* и *B* респективно означавају операнде "*аутор=Андрић*" и "*аутор= Нушић*" и нека су параметри за *ProxTest* постављени на следећи начин:

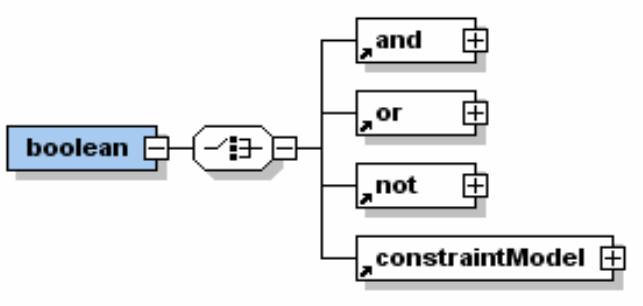
- *Distance* је нула,
- *Relation* је једнако,
- *Unit* је параграф,
- *Ordered flag* је искључен,
- *Exclusion flag* је искључен.

Тада, скуп резултата чине слогови у коме се оба имена аутора појављују у истом параграфу. Ако у истом примеру укључимо *Exclusion flag* резултат чине слогови у коме се оба имена аутора никада не појављују у истом параграфу. Ако укључимо Ordered flag a Exclusion flag је искључен онда резултат чине слогови у коме се име аутора Андрић појављује у истом параграфу, али пре имена аутора Нушић.

3.1.2 XML шема упита

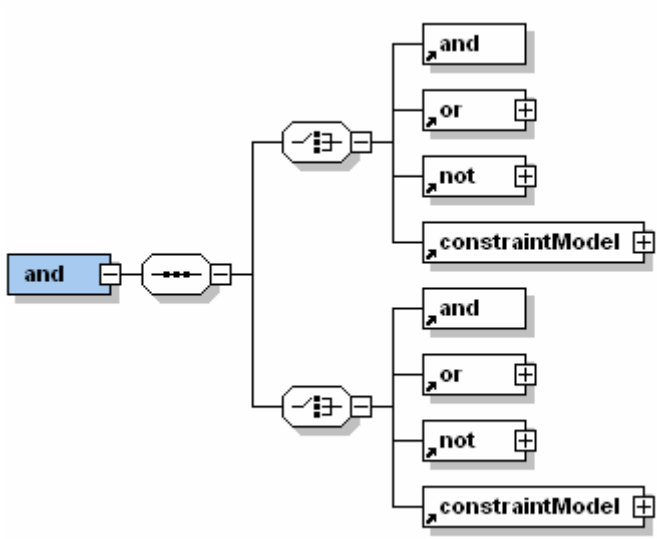
У наставку је приказана XML шема упита типа-1. Ова шема је преузета из система JAFER.

На слици 3.1 приказан је коренски елемент **boolean** са поделементима који представљају операнде.

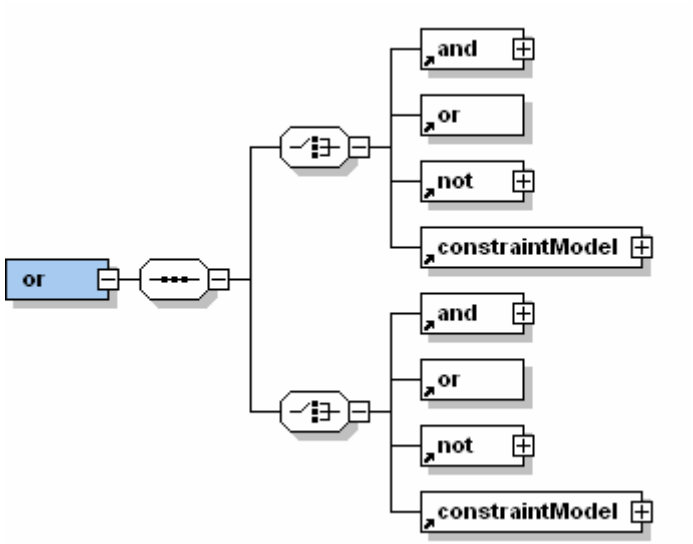


Слика 3.1 Коренски елемент *boolean*

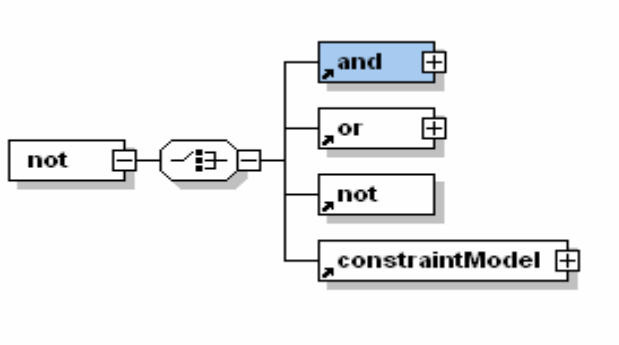
Елементом **and,** приказан на слици 3.2, дефинисан је сложени израз који се састоји од два операнда повезана оператором AND. Операнд може бити поново неки сложени израз или само терм са атрибутима. На сличан начин моделирани су и оператори OR и NOT, који су приказани на сликама 3.3 и 3.4, респективно.



Слика 3.2 Елемент *and*

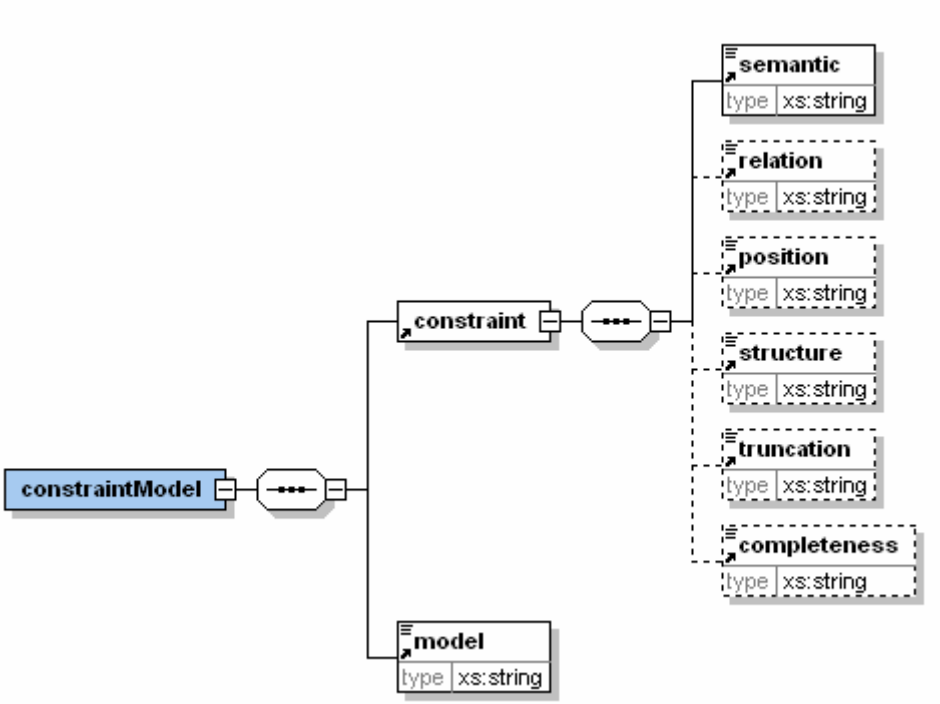


Слика 3.3 Елемент *or*



Слика 3.4 Елемент *not*

На слици 3.5 приказан је елемент **constraintModel**. Овим елементом дефинисана јевредност терма који се претражује и листа атрибута. Терм је дефинисан елементом **model** који је простог тип *string*. Елемент **constraint** састоји се од секвенце поделемената **semantic**, **relation**, **position**, **structure**, **truncation** и **completeness**. Ови поделементи представљају атрибуте из скупа *bib*-1. Тако на пример елемент **semantic** садржи вредност Use атрибута**.**



Слика 3.5 Елемент *constraintModel*

Детаљнијом анализом ове шеме уочено је да шема не покрива упит типа-1 у потпуности. Операнд може бити само терм са листом атрибута, остале могућности за операнд нису подржане. Такође, шема не подржава креирање упита који би садржао оператор *Prox*.

3.2 Z39.50 упит типа-100

Овај упит је познат под називом *Common Command Language* [CCL] и дефинисан је у оквиру стандарда ANSI/NISO Z39.58 *Common Command Language for Online Interactive*. Првобитни циљ овог упитног језика био је да се омогући унос упит путем команде линије на основу чега је и добио име.

3.2.1 Синтакса упита

Синтакса овог типа упита дата је у наставку.

CCL-Find ::= CCL-Find Op Elements | Elements

Op ::= "and" | "or" | "not"

Elements ::= '(' CCL-Find ')' |
Set |
Terms |

Qualifiers Relation Terms |
Qualifiers Relation '(' CCL-Find ')' |
Qualifiers '=' string '-' string

Set ::= 'set' = string

Terms ::= Terms Prox Term | *Term*

Term ::= Term string | *string*

Qualifiers ::= Qualifiers ',' string | *string*

Relation ::= '=' | '>=' | '<=' | '<>' | '>' | '<'

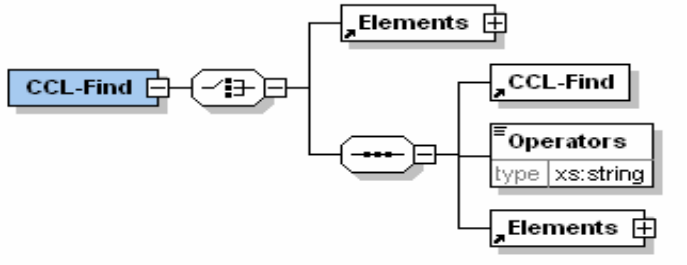
Prox ::= '%' | '!'

Слично као и упит типа-1 и овај упит се може састојати од више елемената повезаних логичким операторима. Оператори могу бити AND, OR и NOT. Поред ових оператора, и у овом типу упита се појављује и оператор Prox који детаљније објашњава везу између два терма која се претражују. Може се дефинисати редослед речи које се претражују или удаљеност једног израза од другог.

Операнд може бити раније добијен скуп резултата који има своје име. Затим, може бити низ стрингова, без икаквих префикса, што је новина у односу на упит типа-1. Код оваквог типа операнда, корисник система за претраживање наводи само изразе по којима жели да претражује, али не наводи никакве префиксе. Систем који имплементира овај тип упита дефинише неке основне индексе и тада се претрага врши по тим индексима. Следећа могућност за операнд је квалификовано име које је одређеном релацијом повезано са термом или новим упитом. Квалификовано име представља неки предефинисани префикс, као што је на пример име аутора, наслов и друго. Код овог типа упита могуће је као операнд поставити и одређени опсег вредности, али само за одговарајућа квалификована имена, што није било могуће у упиту типа-1. На пример, можемо поставити упит који ће пронаћи све књиге из периода од 1990 до 2000. године.

3.2.2 XML шема упита

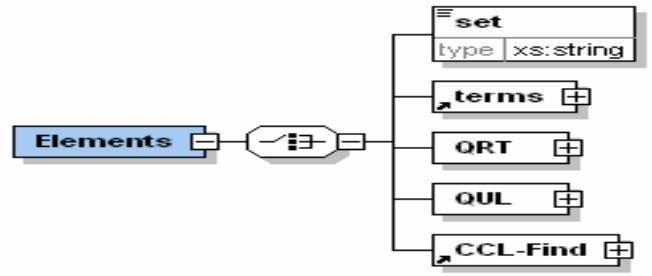
У овом одељку је дат предлог XML шеме којом је моделиран упит типа-100. Коренски елемент је елемент **CCL-Find** и приказан је на слици 3.6.



Слика 3.6 Елемент *CCL-Find*

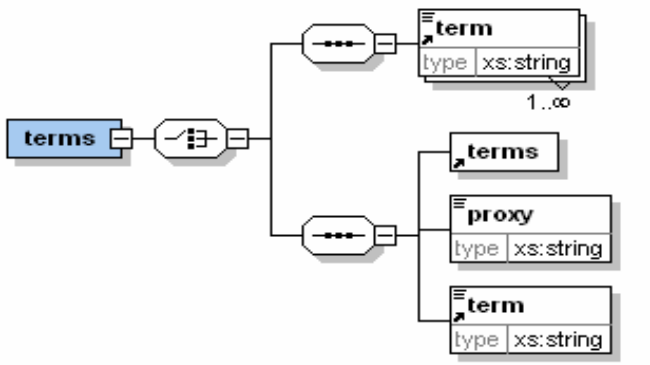
Елемент **Operators** представља логичке операторе који се могу користити у упиту. Он је простог типа и његове вредности могу бити AND, OR или NOT.

На слици 3.7 приказан је елемент **Elements** и он представља операнде који се могу користити у упиту. Елемент **set** је простог типа и садржи информацију о имену претходно добијеног скупа резултата.



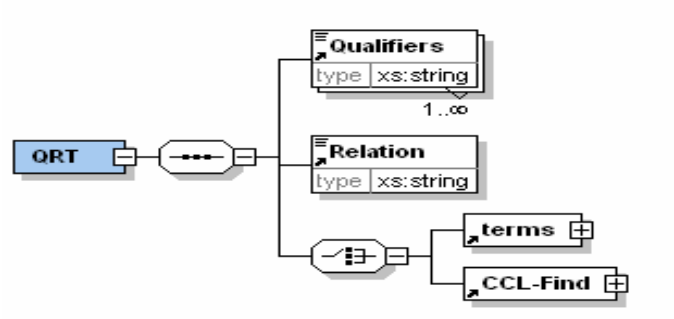
Слика 3.7 Елемент *Elements*

Елемент **terms** је комплексног типа и приказан је на слици 3.8. У елементу **term** памте се конкретне вредности израза који се претражују. Елемент **proxy** је простог типа и представља оператор који ближе означава везу између два терма. Овај елемент може имати две вредности: % која означава да редослед појављивања израза који се траже није битан, и ! која означавада да је редослед појављивања израза специфициран.



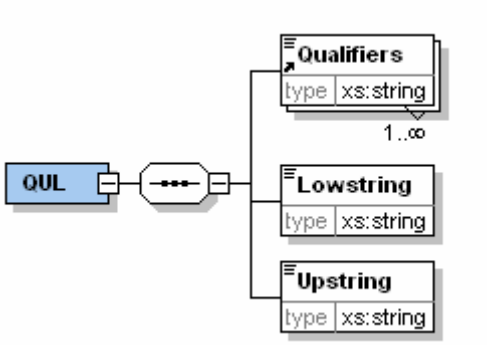
Слика 3.8. Елемент *terms*

На слици 3.9 приказан је елемент **QRT**. Елемент **Qualifiers** представља квалификовано име које ће се користити у претраживању, на пример AU што означава аутора. Елемент **Relation** је простог типа и његове вредности могу бити =, >=, <=, <>, > и <. Он означава одговарајућу релацију између израза који се тражи и квалификатора.



Слика 3.9. Елемент *QRT*

Елементом **QUL**, који је приказан на слици 3.10, је обезбеђено да операнд у упиту може представљати квалификатор чија вредност може бити из одређеног опсега. Елементом **Lowstring** се дефинише доња граница опсега, док се елементом **Uperstring** дефинише горња граница опсега вредности.



Слика 3.10 Елемент *QUL*

3.3 Z39.50 упит типа-102

Други назив под којим је овај упит познат је *Ranked Query List*, јер је заснован на рангирању добијених резултата. Ово је омогућено тако што се сваком операнду додели нека тежина у смислу важности тог операнда. Овај упит тренутно није део Z39.50 стандарда и постоје само предлози како би он требао да изгледа. Један од тих предлога [RLQ] биће описан и у овом раду.

3.3.1 Синтакса упита

Синтакса упита дата је помоћу *Abstract Syntax Notation One* (ASN1). Овде ће бити приказан само део синтаксе док је детаљан опис синтаксе дат у документу [RLQANS1].

```
RankedQuery ::= SEQUENCE {
 needList          [1] IMPLICIT SEQUENCE OF NeedStatement
NeedStatement ::= SEQUENCE {
restrictSet        [1] RestrictSet OPTIONAL,
feedbackInfo      [2] FeedbackInfo OPTIONAL,
rQuery             [3] IMPLICIT OperandPlusWeight OPTIONAL,
weight             [4] IMPLICIT IntUnit OPTIONAL }
OperandPlusWeight ::= SEQUENCE {
operand            [1] CHOICE {
                       attrTerm      AttributesPlusTerm,
                       sOperand    [1] StructuredOperand},
weight             [2] IMPLICIT IntUnit OPTIONAL,
.....
```

Листинг 3.1. Део ASN1 синтаксе упита-102

Упит се састоји од више исказа, где се за сваки овај исказ може дефинисати скуп рестрикција, затим упит и тежина овог једног исказа. Под

рестрикцијама се подразумева да се могу навести имена база које се желе претражити, али се исто тако могу навести и имена база података која не треба да учествују у претраживању. За сваки исказ дефинише се тежина која има вредности између 0 и 1, и њен задатак је да одреди важност овог исказа.

Упит у оквиру једног исказа се састоји од операнада. Операнд може бити листа атрибута или да представља структуирани операнд. У случају да операнд представља листу атрибута тада се као и код упита типа-1 наводи тип атрибута и његова вредност, само што се у овом случају уведени нови типови атрибута. Атрибути који су уведени су следећи:

- ***locationInRecord*** **-** овај атрибут садржи информацију о делу библиографског записа који ће се претраживати, на пример поље или потпоље. Овај атрибут има сличну улогу као и *Use attribute* у упиту Типа-1, само што у овом случају серверска страна не мора да води рачуна о мапирању префикса на конкретна поља записа.
- ***semanticClass*** - атрибут пружа информацију о значењу операнда. На пример да ли је то корпоративно тело, датум или можда монетарна јединица.
- ***contentAuthority*** - атрибут служи да специфицира опсег вредности које операнд може имати. Опсег вредности може бити дефинисан помоћу референце на одређени стандард, на пример "NISO Z39.53-1994 - Шифарник за језике" или одређен договором између учесника у комуникацији.
- ***contentFormat*** - атрибут специфицира енкодинг који треба користити приликом обраде операнда. На пример, овим атрибутом може се специфицирати да је садржај операнда писан на арапском језику.
- ***Relation*** - овај атрибут је еквивалентан истоименом атрибуту из скупа *bib*-1, али овај атрибут може имати само вредности *мање од*, *мање или једнако*, *једнако*, *веће или једнако*, *веће и различито*.

Према томе за сваки операнд се наводи листа атрибута по којима треба вршити претраживање и терм који се претражује.

Када операнд представља структуирани операнд, тада се он може састојати од више нових операнада који су повезани одређеним операторима. Код овог типа упита нису дефинисани стандардни логички оператори AND, OR и NOT већ су дефинисани нови оператори на пример rqAND, rqOR, rqANDNOT чија вредност може бити децималан број између 0 и 1. Ако су два операнда повезана оператором rqAND тада

вредност 0 означава да ни један операнд не мора бити задовољен а 1 да оба операнда морају бити у потпуности задовољена.

Битне разлике које се могу уочити између упита типа-1 и упита типа-102 су следеће:

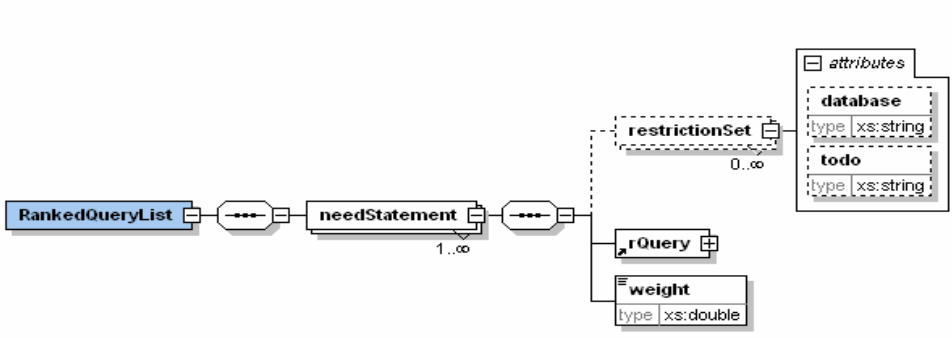
- Тип-1 подржава само бинарне операторе, док тип-102 подржава комбинације са више од два операнда.
- Резултати добијени упитом типа-102 су рангирани на основу нивоа заовољавања постављених критеријума , док са упитом типа-1 то није случај
- У оквиру типа-102 могу се дефинисати правила, која објашњавају серверу како да интерпретира постављени упит.
- Тип-102 обезбеђује могућност дефинисања одређених рестрикција у оквиру добијеног скупа резултата. На пример, могуће је приказати само први 20 резултата, а не цео скуп.

3.3.2 XML шема упита

На слици 3.11 приказан је коренски елемент **RankedQueryList**, који се састоји од секвенце елемената **needStatement.** Елемент **needStatement** представља појединачан израз који се тражи. Сваки елемент **needStatement** састоји се од секвенце елемената **restrictionSet, rQuery** и **weight**.

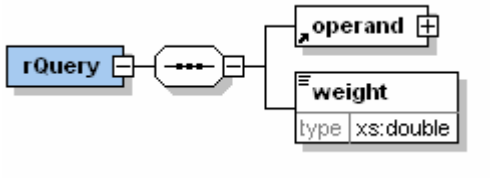
Елемент **restrictionSet** је опциони елемент и садржи атрибуте **database** и **todo**, којима је респективно дефинисано име базе и да ли та база треба да буде укључена или искључене из претраге.

Елемент **weight** је простог типа и садржи информацију о важности целокупног елемента **needStatement**. Вредност овог елемента може бити децималан број између 1 и 0.

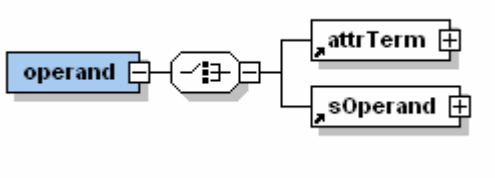


Слика 3.11 Елемент *RankedQueryList*

У елементу **rQuery**, приказаном на слици 3.12, памте се информације о самом упиту. Сваки упит представљен је операндом, ово је приказано елементом **operand**, и одговарајућом важношћу датог упита (елемент **weight**).



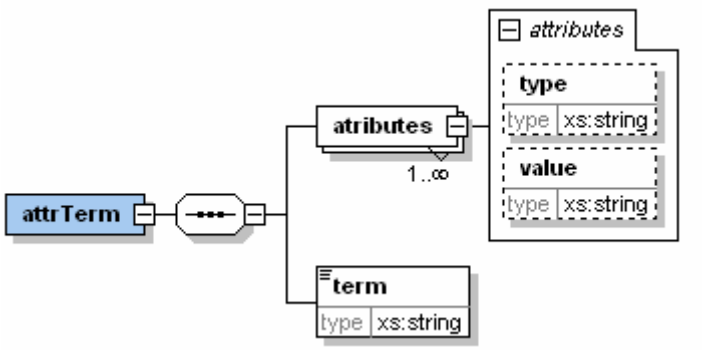
Слика 3.12 Елемент *rQuery*



Слика 3.13 Елемент *operand*

Елемент **operand** приказан је на слици 3.13. У овом типу упита дефинисана су две врсте операнда и оне су у XML шеми моделиране елементима **attrTerm** и **sOperand**.

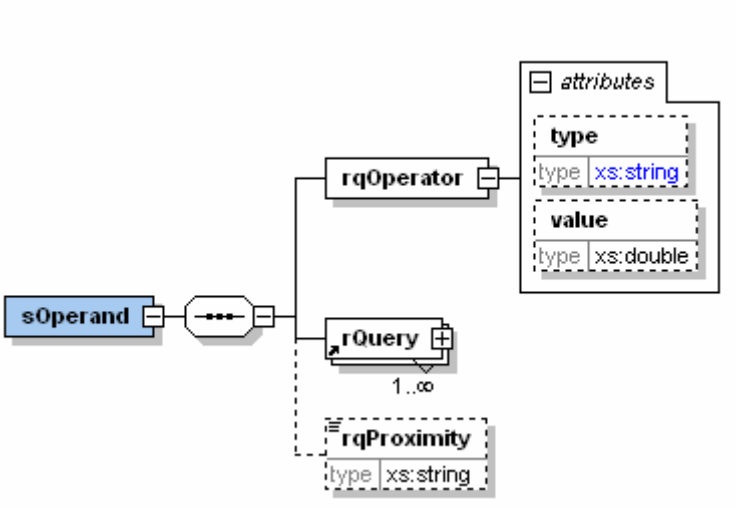
Елемент **attrTerm** је приказан на слици 3.14. Овај елемент представља прост тип операнда јер се састоји од израза који се тражи, представљен елементом **term**, и скупа атрибута, представљени елементом **attributes**. За сваки елемент **attributes** дефинисани су атрибути у којима се памти информација о типу и вредности атрибута који се користи у претраживању.



Слика 3.14 Елемент *attrTerm*

Елемент **sOperand** представља структуирани операнд и приказан је на слици 3.15. Овај елемент садржи елемент **rQuery,** елемент **rqOperator** и елемент **rqProximity**. Елементом **rqOperator** представља операторе који се користе у овом упиту, али као што је у опису упита речено вредност ових оператора је децималан број из опсега 0, 1 и то је моделирано атрибутом **value**. Оператори повезују операнаде дефинисане елементом **rQuery**.

Елементом **rqProximity** описана је веза између операнада у смислу удаљености и редоследа појављивања израза.



Слика 3.15 Елемент *sOperand*

Поглавље 4

Моделирање система

У овом поглављу приказана је спецификација информационих захтева едитора за претраживање и преузимање библиографских записа, дат је опис случајева коришћења, архитектура система и дијаграми секвенци. Као језик за моделирање је коришћен UML (*Unified Modeling Language*) [UML2.0].

UML, обједињени језик моделирања, је графички језик за визуелизацију, специфицирање, конструисање и документовање информација о софтверским системима.

Спецификација **UML 2.0** садржи значајна побољшаља у односу на претходне верзије [UML 2.0]. Спецификација се састоји од четири дела:

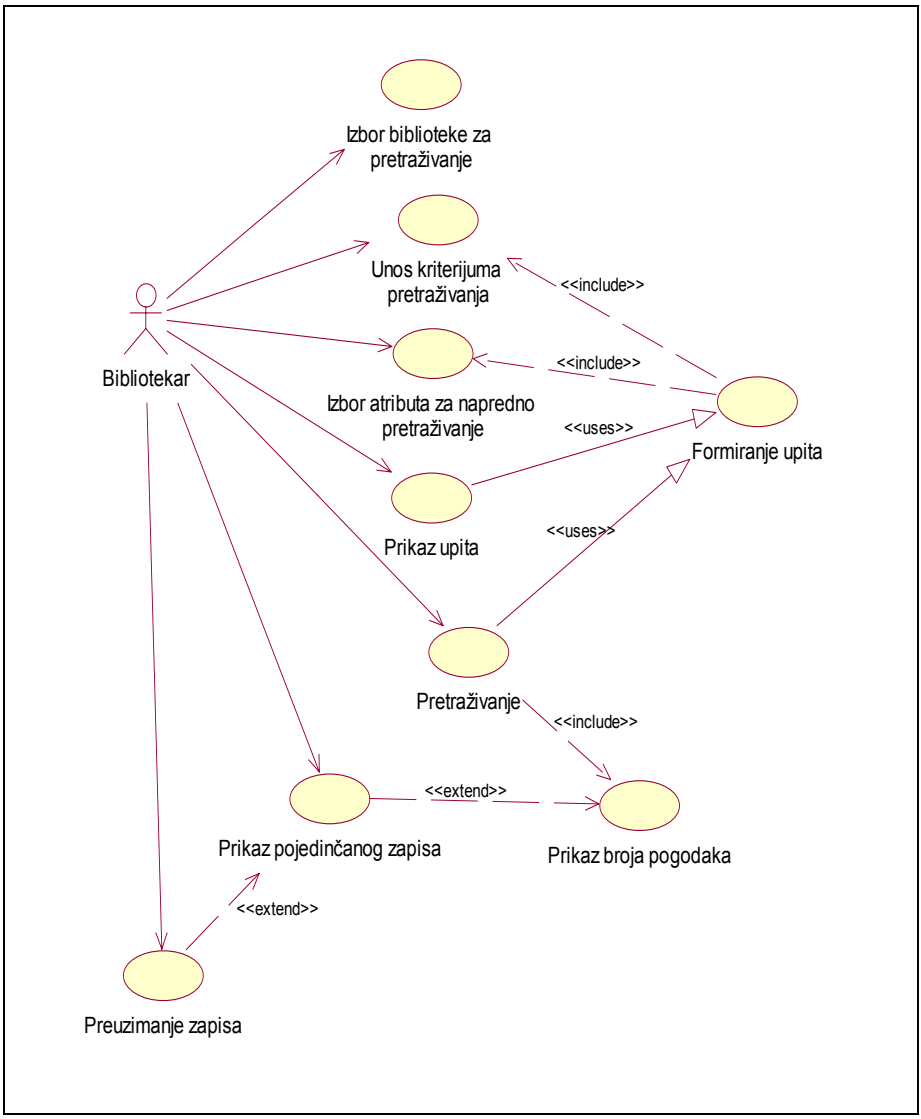
- **UML 2.0 Superstructure:** Представља надоградњу претходне верзије UML-а. Документ је усвојен октобра 2004. и дефинише шест структурних дијаграма, три дијаграма понашања, четири дијаграма интеракције и основне елементе нотације. Једна од основних новина која је у овој верзији UML-а уведена је да се сада за описивање одређених дијаграма такође користи UML нотација, што до сада није био случај и дијаграми су се описивали на неформалан начин.

Преостала три документа UML 2.0 спецификације би требало да ускоро буду усвојена и то су:

- **UML 2.0 Infrastructure**: Дефинише основне класе које чине основу не само за UML 2.0 Superstructure већ и за MOF 2.0 (Meta Object Facility).
- **UML 2.0 Object Constraint Language (OCL):** Омогућава формалан опис ограничења над објектима
- **UML 2.0 Diagram Interchange:** Спецификација која проширује UML метамодел додатним пакетом за графички оријентисане податке, дозвољавајући моделима да буду складиштени или размењени и приказани без икаквог губитка информација.

4.1 Дијаграм случајева коришћења

На слици 4.1 дат је дијаграм случајева коришћења едитора за претраживање и преузимање библиографских записа. У наставку овог одељка дат је текстуални опис наведених случајева коришћења.



Слика 4.1. Дијаграм случајева коришћења

Случај коришћења: Избор библиотеке за претраживање

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Нема

Опис: Библиотекар мора из понуђене листе изабрати библиотеку коју жели да претражује. Уколико у листи не постоји библиотека коју библиотекар тражи тада је он може додати у листу. За сваку библиотеку потребно је додати име библиотеке, име базе коју претражује и адресу (хост и порт) сервера на коме се база коју претражује налази. На сличан начин, библиотекар може променити параметре за неку од већ унетих библиотека или може библиотеку избрисати из листе.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Селектована је библиотека која се жели претражити.

Случај коришћења: Унос критеријума претраживања

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Нема

Опис: Библиотекар из листе понуђених *Use* атрибута бира један по ком жели да претражује. За изабрани атрибут уноси и вредност тог атрибута, односно сам израз по ком жели да претражује.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Унет је критеријум за претраживање

Случај коришћења: Избор атрибута за напредно претраживање

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Извршен је случај коришћења *Унос критеријума претраживања*.

Опис: Као могућност напредног претраживања библиотекару је на располагању преосталих пет атрибута из скупа *bib-1*. Библиотекар може за сваки атрибут да изабере неку од понуђених вредности. У случају да за неки атрибут не изабере вредност, приликом формирања упита узеће се подразумевана вредност тог атрибута.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Изабрани су атрибути за напредно претраживање.

Случај коришћења: Формирање упита

Учесници: Нема

Услови који морају бити задовољени пре извршавања: Извршени су случајеви коришћења *Унос критеријума претраживања* и *Избор атрибута за напредно претраживање*.

Опис: Након што су унети сви потребни параметри за претраживање приступа се формирању упита. Од унетих параметара формира се одговарајући XML елемент. Пошто упит може имати више израза повезаних са логичким операторима AND, OR и NOT овај случај коришћења ће се извршавати више пута, све док се у потпуности не формира упит. Кориснику је такође дата и могућност да избрише или ажурира неки од већ креираних операнада.

Изузеци:
[Грешка при формирању упита] Није унета вредност *Use* атрибута, потребно је унети коректан израз.

Услови који морају бити задовољени после извршавања: Формиран је XML документ упита.

Случај коришћења: Приказ упита

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Извршен је случај коришћења *Формирање упита*.

Опис: При креирању новог операнда графички се приказује креирани упит. Брисање или измена неког операнда се такође одражава и на приказ упита.

Изузеци: Нема
Услови који морају бити задовољени после извршавања: Упит је приказан на екранској форми.

Случај коришћења: Претраживање

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Извршен је случај коришћења *Формирање упита*.

Опис: Успоставља се Z39.50 сесија са одговарајућим сервером на ком се налази изабрана база података и прослеђује се упит који је формирао корисник. При овоме позивају се одговарајуће операција дефинисане по Z39.50 стандарду.

Изузеци:

[Грешка при успостављању сесије] До овога може доћи уколико адреса сервера није коректна, или корисник нема приступа датом серверу
[Сервер не подржава одређене критеријуме претраге] Може се десити да корисник изабере параметре које изабрани сервер не подржава и у том случају ће сервер јавити одговарајућу грешку

Услови који морају бити задовољени после извршавања: Успостављена је конекција са сервером и корисник је проследио упит серверу.

Случај коришћења: Приказ броја погодака

Учесници: Нема

Услови који морају бити задовољени пре извршавања: Извршен је случај коришћења *Претраживање*.

Опис: Након извршене претраге базе података приказује се број пронађених записа који одговарају критеријумима претраге. Уколико није пронађен ни један запис број погодака је нула.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Корисник је информисан о резултату претраге.

Случај коришћења: Преузимање записа

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Постоје записи који задовољавају критеријуме претраге.

Опис: Запис који одговара клијентовим захтевима може се преузети, односно снимити у фајл систему. Запис ће бити сачуван као XML документ.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Запис је успешно преузет.

Случај коришћења: Приказ појединачног записа

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Извршен је случај коришћења *Приказ броја погодака* и постоје записи који задовољавају критеријуме претраге.

Опис: Након извршене претраге, корисник може погледати записе који су пронађени. Записи су приказани у MARC 21 пуном формату, и приказују се један по један на екранској форми.

Изузеци: Нема

Услови који морају бити задовољени после извршавања: Запис је приказан у MARC 21 пуном формату.

4.2 Дијаграм пакета

Током моделирања овог система уочене су одређене самосталне целине, па су из тих разлога те целине организоване као пакети. Дијаграм пакета едитора за преузимање библиографских записа дат је на слици 4.2.

Пакет *marc21* садржи класе које омогућавају да се преузети записи серијализују у одговарајућу објектну структуру и да се прикажу у MARC 21 пуном формату. Преузети запис је представљен XML документом и класе из овог пакета служе да запис складиште у објектну структуру, а из те структуре се запис може лако трансформисати у било који библиотечки формат. Овај пакет преузет је из другог софтверског система који је објашњен у раду [Dimić07a].

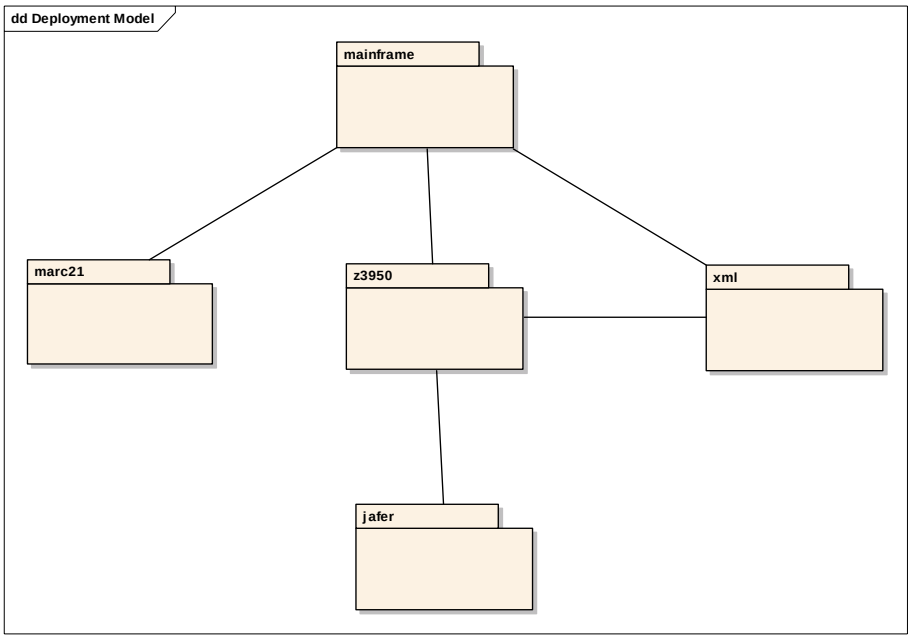
У пакету *jafer* су саджане класе које чине алат JAFER (***J**ava **A**ccess **F**or **E**lectronic **R**esources)*. Овај алат је писан у програмском језику Јава и развијен је на Универзитету у Оксфорду. Основна намена овог алата је да омогући једноставну имплементацију Z39.50 клијента. Овај пакет садржи класе у којима су имплементиране операције дефинисане Z39.50 стандардом.

Пакет *xml* садржи класе које су намењене за рад са XML документима. Класе из овог пакета омогућавају да се учитају параметри о изабраној библиотеци, који се чувају у XML документау. Такође, класе из овог пакета учествују у формирању упит у виду XML документа.

Пакет *z3950* садржи класе којима се реализује позив одговарајућих операција из пакета *jafer*. Класе из овог пакета представљају пословну

логику система и повезују екранске форме едитора и операције које имплементирају протокол Z39.50.

Класе које реализују кориснички интерфејс и омогућавају комуникацију са корисником смештене се у пакету *mainframe*.



Слика 4.2 Дијаграм пакета едитора

4.3 Дијаграм класа

На слици 4.3 дат је дијаграм класа едитора који ће се користити за претраживање и преузимање библиографских записа. Основна класа која се покреће приликом стартовања програма је класа *MainFrame*. Ова класа позива инстанцу класе *EditPanel* која је задужена за приказ почетне екранске форме. Ова класа је повезана са класама *ButtonPanel*, *AdvancePanel* и *ViewPanel* класом. Ове класе су задужене за исцртавање одређених компоненти на екранској форми које обезбеђују функционалност едитора.

Класа *QueryTree* задужена је за приказ формираног упита у виду стабла. Приликом креирања операнда динамички се креирај и нови чвор у стаблу који представља ново формирани операнд.

Класа *Client* садржи операције за претрагу и приказ добијених записа. При реализацији ових операција ова класа позива методе које припадају класама из пакета *jafer*. Ова класа прослеђује формирани XML документ упита класама из пакета *jafer*, које добијени XML документ трансформишу у облик који је дефинисан стандардом Z39.50. Ова класа прослеђује и

захтев за добављање библиографског записа који је добијен у претрази. Једном речју, ова класа представља посредника између екранских форми едитора и протокола Z39.50.

Класа задужена за рад са XML документим је класа *XMLUtil*. Ова класа поседује операције за формирање упита у облику XML документа на основу параметара унешених путем одговарајуће екранске форме. Формирани XML документ представља инстанцу шеме описане у поглављу 3.1.2. Такође, задужена је и за постављање одговарајућих параметара који се читају из конфигурационог XML фајла, а који се односе на параметре везане за библиотеку која нуди претраживање и преузимање библиографских записа путем протокола Z39.50.

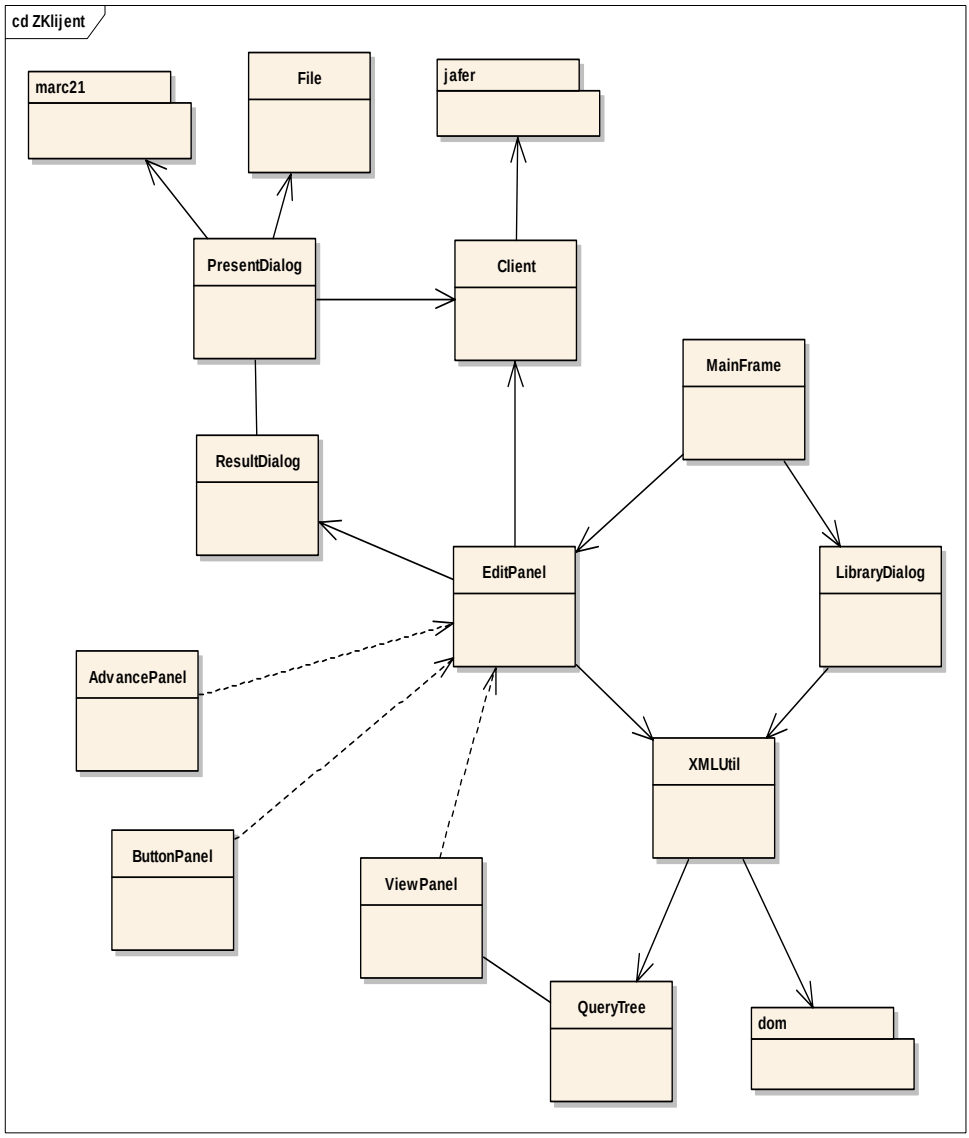
Класа *ResultDialog* је задужена за приказ нове екранске форме која служи да прикаже број погодака добијених после претраге.

Да би се приказао неки од записа добијени претрагом, потребно је позвати операције класе *PresentDialog*, која је задужена за графички приказ пронађених записа.

Класа *LibraryDialog* обезбеђује функционлност додавања и ажурирања библиотека које се претражују путем протокола Z39.50.

Пакет *dom* представља стандардни *Java* пакет за рад са XML документима.

Преузети записи се снимају у фајл систем и зато је за рад са документима искоришћена уграђена Јавина класа *File*. Међутим предвиђено је да се у наредним реализацијама овог едитора преузети записи учитавају у систем који ће служити за обраду преузетих записа.



Слика 4.3 Дијаграм класа едитора

4.4 Дијаграм секвенци

На основу приказаног дијаграма случајева коришћења едитора за претраживање и преузимање библиографских записа креирани су дијаграми секвенци који описују одређене сценарије случајева коришћења.

4.4.1 Дијаграм секвенци *Search*

На слици 4.4 приказан је дијаграм секвенци *Search*, којим је описан целокупан сценарио почевши од покретања апликације па до дела када корисник притисне дугме за претраживање и након тога добије резултате претраге.

Корисник покреће апликацију позивом операције *main*. Ова порука иницира приказ почетне екранске форме едитора и врши се учитавање параметара из конфигурационог XML фајла, где се чувају подаци о расположивим библиотекама за претраживање.

Објекат класе *MainFrame* упућује поруку *chooseServer()* објекту класе *EditPanel*, чиме се врши избор библиотеке, односно сервера, која ће се претраживати. Такође, врши се додавање нове библиотеке уколико не постоји у постојећем списку, или се врши ажурирање параметара за неку од изабраних библиотека.

Параметри које је корисник унео путем екранске форме се трансформишу у XML документ упита који ће бити погодан за даљу обраду. Ове активности приказане су на дијаграму *createQuery.*

Након тога, упит је приказан на екрану и ако је корисник задовољан упитом може га послати, односно започети претрагу базе. Упит који је представљен XML документом заједно са параметрима библиотеке која се претражује се прво шаље објекту класе *Client*, који ће тај упит проследити објекту класе *ZClient*. На дијаграму су ове активности представљене слањем порука *serach(XML, library)* и *submitQuery(XML)*, респективно. Објекат класе *ZClient* ће прво успоставити конекцију са објектом класе *Z3950Server* слањем поруке *init(),* и након потврде о успешној конекцији на сервер, послати поруку *transformXMLtoRPQ(XML)* којом ће се извршити трансформација упит из XML документа у формат дефинисан стандардом Z39.50. Након тога послаће поруку *search*(*RNPQuery)* објекту класе *Z3950Server*, у којој су поред упита прослеђени сви потребни параметри који су дефинисани Z39.50 стандардом. Објекат класе *Z3950Server* није део саме апликације и представља стварни сервер на коме се налази удаљена база података. Уколико је сервер одбио успостављање конекције са клијентом корисник добија информацију о томе и може пробати да поново креира упит и успостави конекцију. Такође, уколико корисник није

задовољан постављеним упитом, може одустати од претраге и поставити нови упит.

Након завршене претраге објекат класе *EditPane*l шаље поруку *showNumbersOfHits* објекту класе *ResultDialog* и на тај начин се корисник информише о резултату претраге. Ако су пронађени записи који одговарају постављеном упиту, могуће је прегледати пронађене записе, што је описано дијаграмом секвенци *reviewRecords*. У супротном, корисник може поставити нове критеријуме претраге за нови упит.

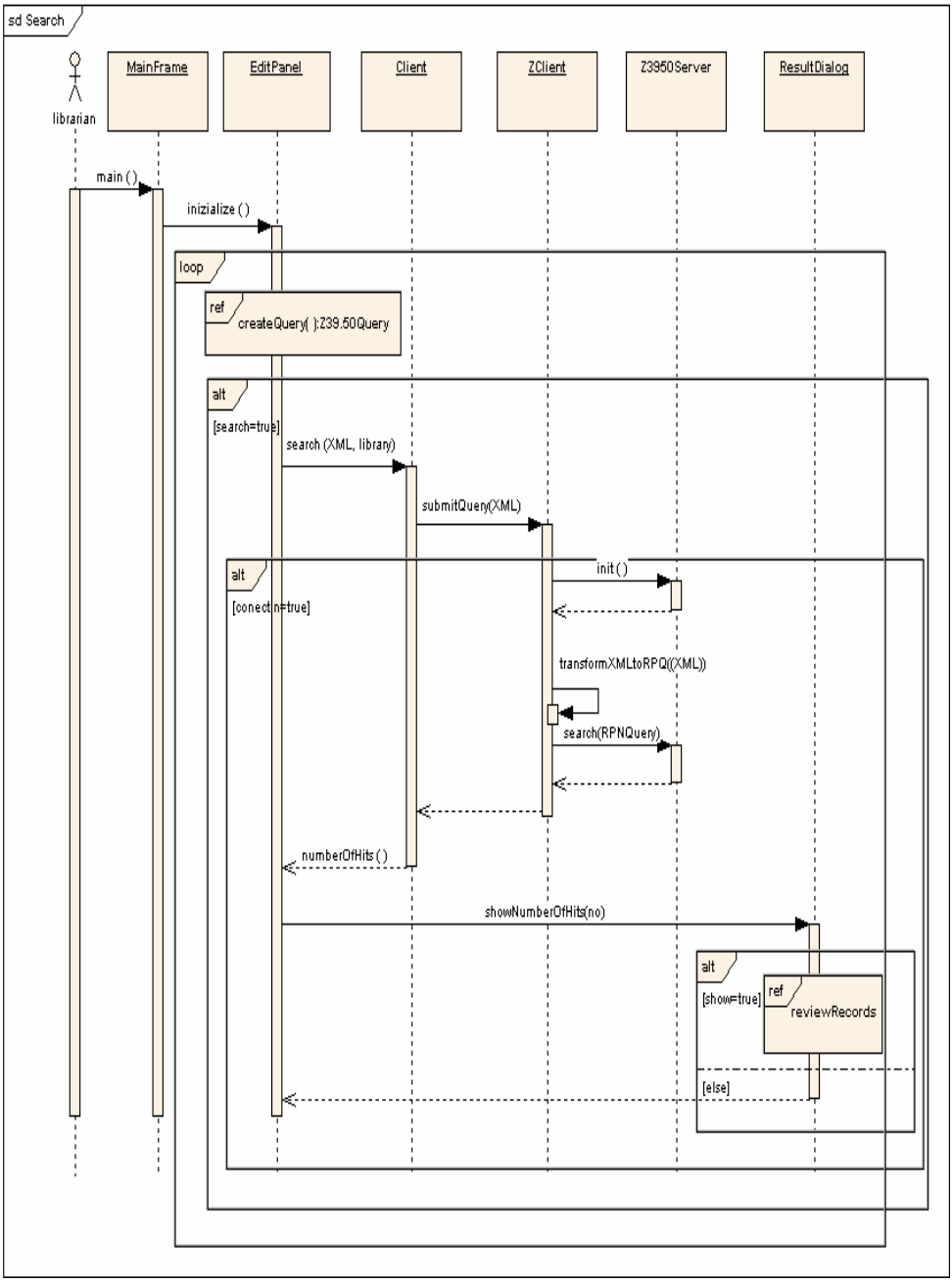
4.4.2 Дијаграм секвенци createQuery

У овом делу дат је путем дијаграма секвенци (слика 4.5) опис метода који на основу података које је корисник унео формира XML документ који представља инстанцу XML шеме дате у одељку 3.1.2. Објекти на дијаграму секвенци представљају инстанце истоимених класа са дијаграма класа (слика 4.3), док објекат *Node* представља инстанцу истоимене класе која припада Јавином пакету *dom* за рад са XML документима.

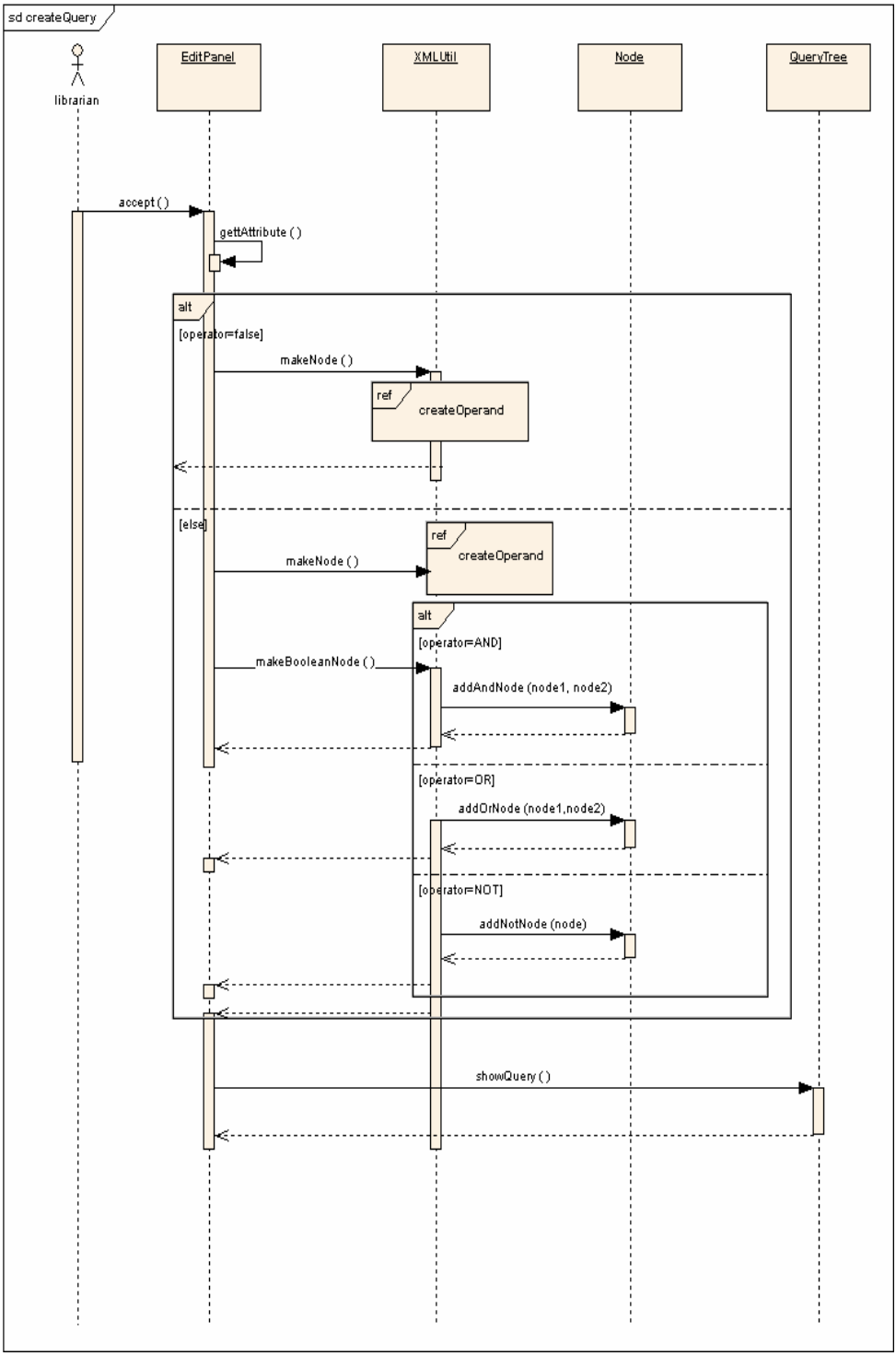
Формирање операнда започиње слањем поруке *accept()* коју иницира корисник систем - *librarian*. Када је објекат класе *EditPanel* примио поруку, он шаље поруку *getAttribute*(). Задатак ове поруке је да преузме све унете атрибуте и њихове вредности. Након тога потребно је формирати операнд.

Уколико се формира први операнд то значи да корисник није изабрао оператор и тада ће објекат класе *EditPanel* послати поруку *makeNode*() објекту класе *XMLUtil*. Креирање једног операнда описано је дијаграмом *createOperand*. На крају објекат класе *Node* враћа поруку која садржи креирани елемент (то је елемент *constraintModel* приказан на слици 3.5) и он представља један операнд у упиту.

У случају да корисник жели да направи сложенији упит који укључује логичке операторе, објекат класе *EditPanel* ће прво послати поруку *makeNode()* чиме ће условити извршавање акција описаних дијаграмом *createOperand*. На овај начин ће се од унетих параметара формирати нови операнд, а затим ће се слањем поруке *makeBooleanNode()* креирати нови елемент. У зависности који логички оператор је изабран објекат класе *Client* ће слати одговарајуће поруке. На пример, ако је изабран оператор AND послаће поруку *addAndNode*(*node1*,*node2*), где *node1* и *node2* представљају операнде који су раније креирани. Као резултат добија се елемент приказан на слици 3.2. Након што је операнд формиран он се може и приказати на екрану. Секвенца ових порука се понавља све док корисник не одлучи да изврши претраживање удаљене базе. Тада се овако формиран XML документ трансформише у форму која је погодна за даљи рад.



Слика 4.4 *Дијаграм секвенци Search*



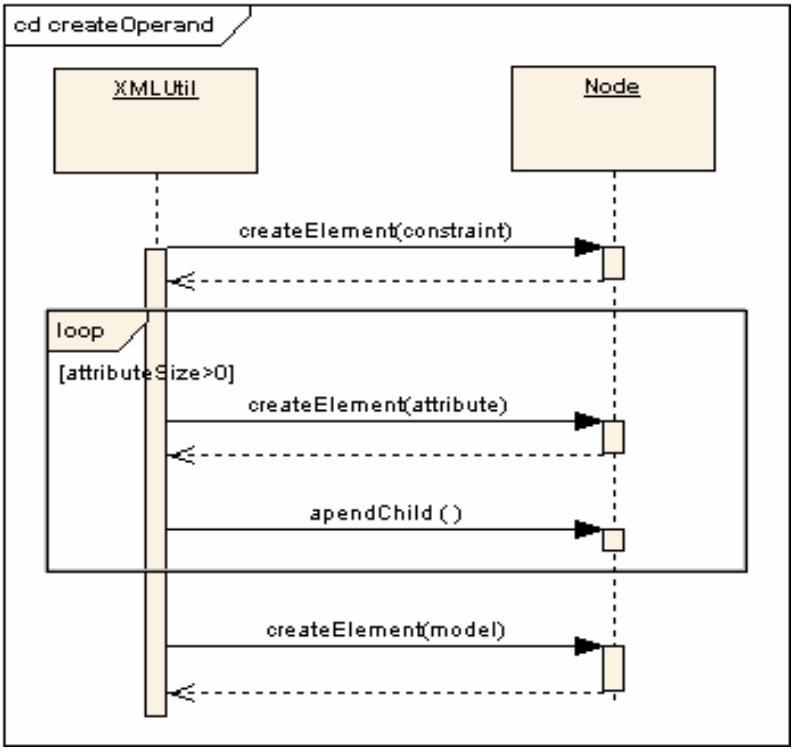
Слика 4.5 *Дијаграм секвенци createQuery*

4.4.3 Дијаграм секвенци *createOperand*

Дијаграмом createOperand (слика 4.6) описан је низ акција које је потребно извршити како би се од унетих параметара са екранске форме формирао један операнд, односно један елемент у XML документу који је приказан на слици 3.5.

Као што је описано у поглављу 3.1.2 атрибути из скупа *bib*-1 су представљени појединачним елементима у оквиру елемента *constraint*, док је сам израз који се тражи представљен елементом *model*.

Прво је потребно креирати елемент *constraint*, што је на дијаграму приказано слањем поруке *createElement(constraint)*. Затим се за сваки атрибут креира нови елемент и додаје на већ формирани XML документ, на дијаграму је то описано слањем порука *createElement(attribute)* и *apendChild()*, респективно. Након тога потребно је још формирати и елемент *model* којим је описан израз који се тражи.



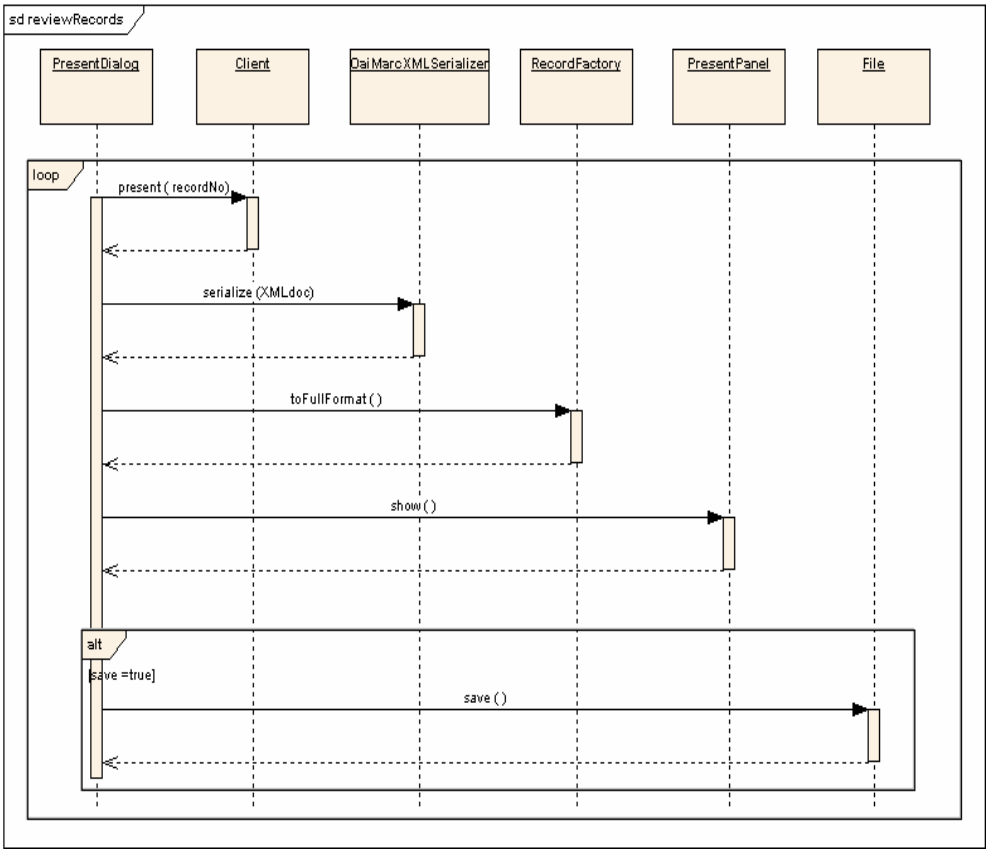
Слика 4.6 *Дијаграм секвенци createOperand*

4.4.4 Дијаграм секвенци *reviewRecords*

Ако постоје тражени записи могуће их је прегледати, корисник може неограничен број пута да прегледа једном пронађене записе и то је на дијаграму описано комбинованим фрагментом *loop*.

Прво је потребно да се специфицира који запис се жели погледати, и то је на дијаграму описано слањем поруке *present(recordNo)* објекту класе *Client*, где параметар *recordNo* представља редни број записа у скупу добијених резултата претраживања.

Када је запис добављен он је представљен у виду XML документа и због тога се врши серијализација у објектну репрезентацију записа. Ова активност је описана слањем поруке *serialize()* објекту класе *OaiMarcXMLSerialize*, која припада пакету *marc21*.



Слика 4.7 Дијаграм секвенци *reviewRecords*

Класа *RecordFactory* такође припада пакету *marc21*, и задужена је за приказивање записа у MARC 21 пуном формату. Након тога, запис се приказује у оквиру екранске форме што је реализовано слањем поруке *show()* објекту класе *PresentPanel*.

Уколико је потребно неки запис преузети и сачувати тада објекат класе *PresentDialog* шаље поруку *save()* објекту класе *File*. У овој реализацији едитора за претраживање и преузимање библиографских записа предвиђено је да се преузети записи чувају у виду XML документа у фајл систему, међутим може се обезбедити и складиштење у базу података система у који би се интегрисао едитор за претраживање.

Поглавље 5

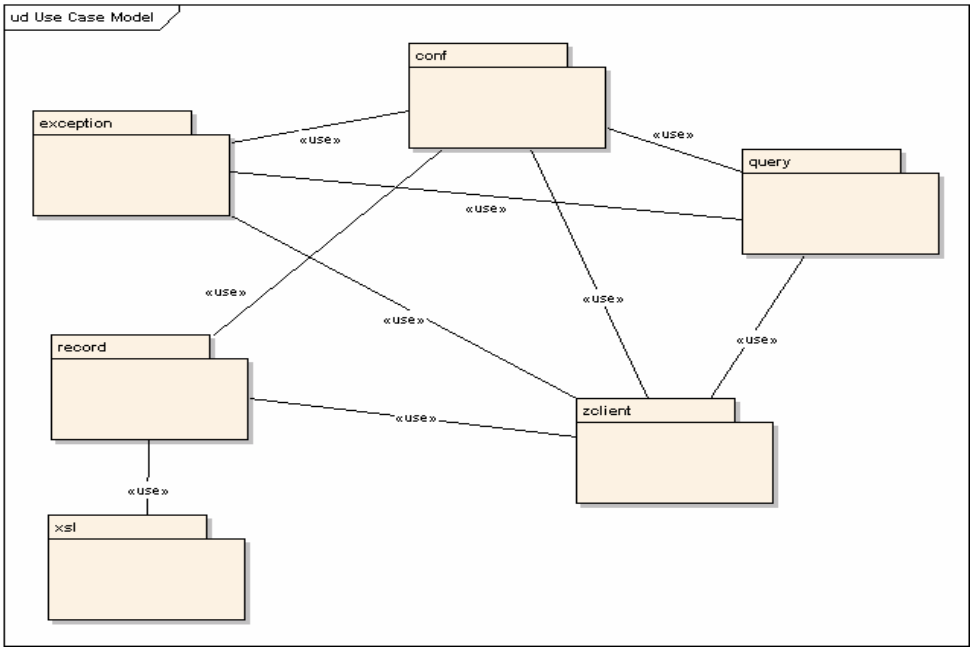
Имплементација система

У овом поглављу је описана имплементација едитора за претраживање и преузимање библиографских записа. Имплементација је заснована на статичким и динамичким моделима који су приказани у четвртом поглављу.

Едитор за претраживање и преузимање библиографских записа је имплементиран у програмском језику Јава, а за реализацију клијентске стране протокола Z39.50 искоришћен је алат JAFER.

5.1 Архитектура алата JAFER

JAFER (Java Access For Electronic Resources) [JAFER] је алат који имплементира клијентску страну протокола Z39.50 у програмском језику Јава и заснован је на XML технологијама. Алат се развија на универзитету у Оксфорду и представља *open source* библиотеку. Верификација овог пакета је урађена на Оксфордском универзитету у оквиру имплементације *web* портала који пружа преузимање библиографских записа по протоколу Z39.50. Дијаграм пакета дат је на слици 5.1.



Слика 5.1 Дијаграм пакета система JAFER

У пакету *conf* се налазе разни конфигурациони фајлови, а између осталог се налази и XML документ *bib1Attributes.xml* којим је описан скуп атрибута *bib*-1. Део овог документа дат је листингом 5.1. У овом документу је дато више скупова атрибута који се могу користити при претраживању. За сваки такав скуп чува се информација о имену скупа, у атрибуту *name* елемента *attributeSet*, а атрибутом *name* елемента *attributeType* дефинише се тип атрибута из тог скупа. Конкретно за скуп *bib*-1, тип атрибута је *Use* атрибут, *Relation* атрибут и остали типови из скупа *bib*-1. Сваки тип атрибута има своје предефинисане вредности и оне се памте у елементу *attribute*. Атрибутом *name* дефинише се име атрибута, а са атрибутом *value* његова нумеричка вредност. На пример, постоји атрибут *title* који је типа *Use* атрибута и његова нумеричка вредност је 4. Разлог мапирања имена атрибута у јединствену нумеричку вредност лежи у самом протоколу Z39.50, јер се у синтакси упита који се преноси овим протоколом очекује нумеричка вредност.

```
<attributeSets syntax="1.2.840.10003.3.1">
 <attributeSet default="true" name="bib1" version="Z39 50">
   <attributeType name="semantic" value="1">
     <attribute name="personal_name" value="1" />
     <attribute name="corporate_name" value="2" />
     ......................
     <attribute name="title" value="4" />
   </attributeType>
   <attributeType name="relation" value="2">
     <attribute name="lt" value="1" />
     <attribute name="le" value="2" />
     <attribute name="eq" value="3" />
     ..........................
   </attributeType>
   ......................
 </attributeSet>
</attributeSetc>
```

Листинг 5.1 Део документа *bib1Attributes.xml*

Изузеци који се могу јавити дефинисани су класама из пакета *exception.* Стандардом Z39.50 дефинисан је одређени скуп грешака које се јављају при комуникацији између клијентске и серверске стране. Те грешке се називају *дијагностике*, и уколико на пример, сервер одбије комуникацију са клијентом, серверска страна протокола ће вратити шифру грешке (дијагностике). Списак шифара грешака и њихових значења налази се у фајлу *bib1Diagnostics.xml*, који се налази у пакету *conf*. Део XML документа *bib1Diagnostics.xml* дат је листингом 5.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<bib1Diagnostics syntax="1.2.840.10003.4.1">
  <condition value="0">unknown</condition>
  <condition value="1">permanent system error</condition>
  <condition value="2">temporary system error</condition>
  <condition value="3">unsupported search</condition>
  <condition value="4">Terms only exclusion (stop)  words
  </condition>
  <condition value="5">Too many argument words</condition>
  <condition value="6">Too many boolean operators
  </condition>
  <condition value="7">Too many truncated words</condition>
  <condition value="8">Too many incomplete subfields
  </condition>
  <condition value="9">Truncated words too short
  </condition>
  ...................
  <condition value="12">Too many records retrieved
  </condition>
  ...................
</bib1Diagnostics>
```

Листинг 5.2 *Део документа bib1Diagnostics.xml*

Класе које су задужене за креирање упита дефинисане су у оквиру пакета *query*. Класа *QueryBuilder* омогућава креирање упита који је репрезентован објектом класе *Node* из пакета *org.w3c.xml*. Класа *XMLRPNQuery* задужена је да упит, који је представљен објектом класе *Node,* трансформише у синтаксу упита типа-1.

Пакет *records* садржи класе које су задужене да пронађени запис, који се путем протокола Z39.50 шаље у виду одређеног формата серијализује у XML документ. То значи да сервер са којим је успостављена комуникација позивом сервиса *Present* шаље библиографски запис у виду бинарне репрезентације. На пример, ако је у *Present* захтеву наведено да се желе преузети MARC запис тада ће сервер послати запис у формату познатом под називом ISO2709 [ISO2709]. Коришћењем класа из овог пакета се тај запис серијализује у XML документ.

Овим алатом је омогућено да корисник преузме запис у виду XML документа који може бити инстанца било које XML шеме. Из тог разлога у пакету *xsl* се налазе XSL документи којима се инстанце једне шеме трансформишу у инстанце друге шеме. У овом алату добијени запис се прво серијализује у шему записа [OAIMARC], а затим се применом XSL трансформација трансформише у неке друге шеме записа као што је на пример шема записа [MARCSlim].

Класе из пакет *zclient* воде рачуна о успостављању комуникације са серверском страном. Затим, у овом пакету се налазе и класе које имплементирају сервисе дефинисане стандардом Z39.50. Овај алат тренутно имплементира само сервисе *Initialization*, *Present*, *Search* и *Scan*.

5.2 Едитор за преузимање библиографске грађе

У овом одељку дати су основни елементи имлементације едитора за претраживање и преузимање библиографских записа.

За реализацију корисничког интерфејса искоришћене су `Swing` компоненте. Пакет ***javax.swing*** [Swing] представља пакет за рад са графичким интерфејсом. Садржи пакете и класе који су намењени за имплементацију различитих компоненти које се могу појавити у оквиру корисничког интерфејса. За разлику од пакета *awt (Abstract Window Toolkit)*, ***swing*** пружа софистициране GUI (*graphical user interface*) компоненте које не зависе од платформе и изглед тих компоненти не мора бити исти као изглед корисничког интерфејса на датој платформи. Једина мана ових компоненти је спорије извршавање.

За имплементацију изгледа екранских форми искоришћене су класе из пакета *mainframe*. Функционалност екранских форми је описана случајевима коришћења који су описани у четвртом поглављу. Изглед екранских форми као и начин коришћења система објашњен је у шестом поглављу. Имплементиране су следеће екранске форме:

- ажурирање листе библиотека за претраживање (слике 6.2, 6.3 , 6.4),
- формирање упита (дугме *Prihvati upit* на слици 6.1),

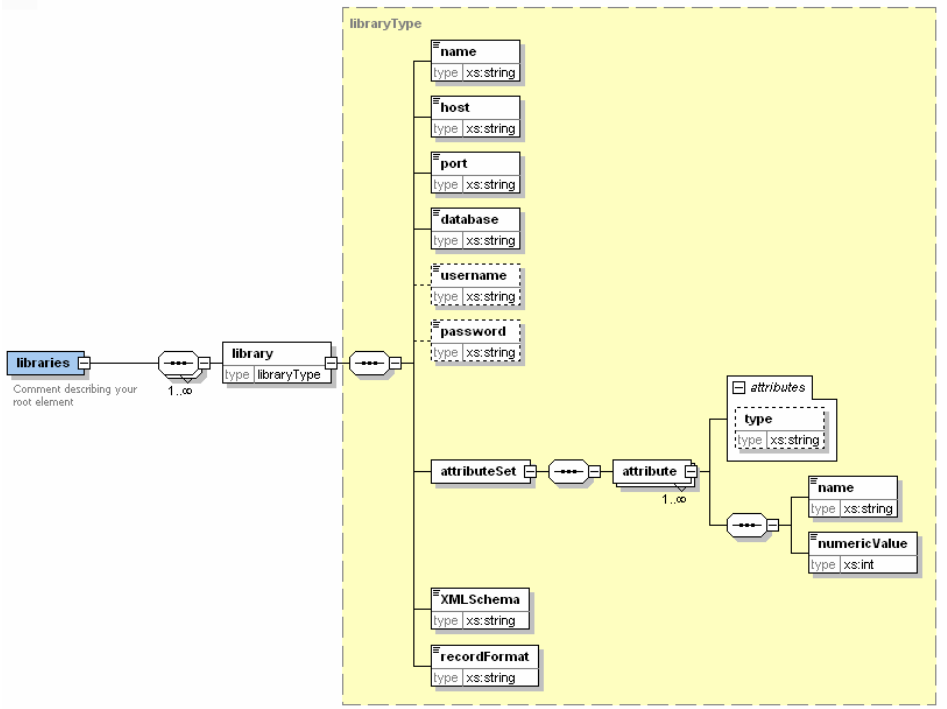
- приказ упита (слика 6.1),
- успостављање конекције са Z-сервером (дугме *Traži* на слици 6.1),
- преглед библиотечких записа (слика 6.6).

Ажурирање листе библиотека за претраживање: За конекцију на неку библиотеку која пружа услуге претраживања и преузимања записа по протоколу Z39.50 потребно је унети одговарајуће параметре за Z39.50 сервер. Обавезни параметри који се очекују су: хост и порт на ком се налази сервер, назив базе којој се приступа, пошто на једном хосту и порту може постојати више база података, и на крају потребно је унети корисничко име и лозинку ако је потребна аутентификација за приступ наведеној бази. Такође, потребно је дефинисати XML шему записа који се преузима, као и податак да ли се захтева целокупан садржај записа (*full format*) или сажети приказ (*brief format*). Пошто свака библиотека има одређен подскуп атрибута из скупа *bib*-1 које подржава, потребно је сачувати и те податке.

За складиштење потребних информација искориштен је *Extensible Markup Language* (XML) [XML]. То је проширив маркерски језик за управљање, приказивање и организовање података. Независтан је од програмских језика, рачунарских платформи и оперативних система. Дизајниран је са циљем да се превазиђу недостаци језика SGML (*Structured Generalized Markup Language*) и HTML (*Hypertext Markup Language*). XML описује класу објеката који се зову XML документи. Основне особине XML докумената су: погодни су за комуникацију на Интернету, програми их лако процесирају, јасни су и читљиви од стране човека, једноставно и брзо се дизајнирају на формалан и концизан начин.

Садржај класе XML докумената се може описати помоћу *XML Schema Language* [XMLSchema]. Поред *XML Schema* језика у ову сврху се користи и *Document Type Definition* (DTD). *XML Schema* језик је далеко напреднији од DTD-а. Основне предности *XML Schema* језика у односу на DTD су: запис у XML синтакси, детаљнији и чвршћи модели, подршка за типове података, простори имена, документовање.

Потребни подаци се чувају у XML документу који представља инстанцу XML шема приказане на слици 5.2.



Слика 5.2 XML шема документа који садржи конфигурационе параметре за библиотеке

Коренски елемент *libraries* може садржати више поделемената *library* у ком се налазе подаци везани за једну библиотеку. Елемент *name* садржи име библиотеке и оно ће бити приказано у падајућој листи из које корисник бира библиотеку коју жели да претражи. Елементи *host* и *port* садрже податке о конкретној адреси на којој се сервер налази, а елемент *username* и *password* су опциони и у њима се памте информације о корисничком имену и лозинки уколико сервер захтева идентификацију. Елемент *database* представља назив базе података којој се приступа и која садржи податке о записима. Елемент *attributeSet* представља скуп атрибута које Z39.50 сервер библиотеке подржава. Самим Z39.50 стандардом није дефинисано да свака библиотека мора да омогући претраживање по свим атрибутима, већ је то индивидуално и зависи од имплементације Z39.50 сервера који је употребљен. Сваки подржани атрибут чува се у елементу *attribute* који има атрибут *type*, у коме се памти информација о типу атрибута, на пример да ли је то *Use* атрибут, *Relation* атрибут или неки други из скупа *bib-1*. У поделементу *name* се чува податак о називу атрибута (на пример *title*) а у поделементу *numericValue* се налази нумеричка вредност тог атрибута која је дефинисана у оквиру скупа *bib*-1. У зависности која библиотека је селектована подаци из ових елемената се динамички учитавају у падајуће

листе којима су представљени атрибути на почетној екранској форми едитора.

Приликом конекције на библиотеку коју је корисник изабрао сви потребни параметри се читају из XML документа *library.xml,* који представља инстанцу описане шеме. Ради једноставнијег рада са XML документима искоришћен је алат *XMLBeans* [XMLBeans]. Ово је јавно доступан алат и омогућава да се на основу XML шеме за сваки елемент из шеме изгенеришу одговарајуће класе које се састоје од `get` и `set` метода. Потребна функционалност додавања, брисања или измена података за неку библиотеку обезбеђена је кроз методе класе `XMLUtil`:

- `public static List getLibName()`- метод враћа листу назива библиотека које постоје у систему.
- `public static LibraryType getLibrary(String name)`- овај метод за дато име библиотеке враћа објекат класе `LibraryType`. Класа `LibraryType` је изгенерисана на основу елемента *library* који је сложеног именованог типа *libraryType* и са одговарајућим `get` и `set` методама добијају се сви потребни подаци о датој библиотеци.
- `public static void addLibrary(String name, String database, String host,String port, String username, String password)` – метод је задужен за додавање нове библиотеке, односно додавање новог елемента у XML документ.
- `public static void updateLibrary(String name, String database, String host,String port, String username, String password)`- метод врши ажурирање одговарајућих параметара за селектовану библиотеку.
- `public static void deleteLibrary(String name)`- метод брише селектовану библиотеку.
- `public static void addAttribute(String[] old, String[] newatt, String lib)`- метод за селектовану библиотеку врши ажурирање већ постојећег атрибута или уколико тај атрибут не постоји додаје нови атрибут за дату библиотеку. Атрибути су јединствено одређени према њиховом називу (поделемент *name* елемента *attribute*).
- `public static void deleteAttribute(String attribute, String lib)`-метод брише селектовани атрибут за изабрану библиотеку.
- `public static List load(String library)`- метод за изабрану библиотеку враћа све њене дефинисане атрибуте.

За имплементацију екранске форме која омогућава читање, брисање и ажурирање параметара из овог документа задужене су класе `LibraryDialog`, `LibraryTableModel` и `AttributeTableModel`. Класе `LibraryTableModel` и `AttributeTableModel` наслеђују класу `AbstractTableModel` и оне имплементирају табеле у којима се графички приказују подаци о имену библиотеке, односно подаци о типу и вредности атрибута из скупа *bib*-1.

Формирање упита: За креирање упита искоришћена је објектна репрезентација XML документа *Dom (Document Object Model)* [DOM] која омогућава динамички приступ и промену садржаја и структуре документа. Модификацију и динамички приступ овој структури могуће је вршити уз помоћ различитих програмских језика, између осталих и програмског језика Јаве. Пакет **org.w3c.dom** садржи интерфејсе чије инстанце представљају логичке делове неког XML документа, као што су елементи, атрибути, коментари, инструкције за обраду и остало. Класе из овог пакета омогућавају креирање XML документа и делове документа, затим навигацију кроз документ, копирање и уклањање делова документа, измену садржаја елемената и атрибута у оквиру документа.

Формирање упита је описано дијаграмом секвенци *createQuery* у поглављу четири. Пре свега креира се објекат класе `Node` из пакета *org.w3c.xml*, који представља објектну репрезентацију упита који је описан инстанцом XML шеме дате у поглављу 3.1.2. При имплементацији се прво учитају све вредности атрибута са екранске форме и од тога се формира један елемент XML документа. Када корисник унесе један операнд тада може изабрати неки логички оператор и унети параметре за други операнд. У зависности који оператор је изабран позиваће се одређене методе из класе `XMLUtil`:

- `public static Node makeNode(String profil,String term)`- метод на основу терма који се претражује и атрибута који су представљени параметром `profil` креира један операнд, односно један чвор у XML документу.
- `public static Node makeBoolNode(Node query1,Node query2,String oper)`- овај метод на основу два операнда креира нови операнд који је повезан са логичким оператором представљеним параметром `oper`.
- `public static Node makeQuery(List operandlist)`-овај метод као параметар очекује листу операнада, где је сваки операнд представљен објектом класе `OperandBean`. Класа `OperandBean` садржи податке који се односе на један операнд, односно терм који се тражи, затим вредности атрибута из скупа *Bib-1* и оператор који

ће се користити уколико је упит сложен, као и одговарајуће `get` и `set` методе. Метод је тако реализован да се пролази кроз сваки операнд и позивају методе `makeNode` и `makeBoolNode`.

Пошто је сам упит типа-1 дефинисан у виду реверзне пољске нотације, а и XML шема приказана у поглављу 3.1.2 је креирана у сличном маниру, при имлементацији је искоришћена та погодност да се на основу претходно креираног XML елемента и елемента који је креиран од података са екранске форме креира нови елемент који представља оператор. Овај поступак креирања XML документа ће се понављати све док корисник не одлучи да изврши претраживање.

Едитор је имплементиран да подржи упит типа-1, који је описан стандардом Z39.50, али одређеном модификацијом едитор би се могао прилагодити за формирање осталих врста упита које су такође дефинисане да се могу користити при комуникацији путем протокола Z39.50. Под модификацијом би се подразумевало прилагођавање екранске форме захтевима које налаже нова врста упита, док би сама архитектура система остала непромењена. То значи да би се у том случају такође креирао XML документ који би био инстанца неке друге шеме, то би могле бити и шеме описане у поглављима 3.2.2 и 3.3.2.

Приказ упита: Креирани упит се приказује у доњем делу екранске форме. Упит се приказује у виду стабла и за имплементацију су искоришћене `QueryTree`, `QueryTreeModel` и `QueryTreeCellRenderer` које имплементирају Јавине класе `JTree`, `TreeModel` и `DefaultTreeCellRenderer`, респективно. Приказани упит се у доњем делу може само гледати али не и ажурирати.

Класа `QueryTreeModel` на основу креираног упита формира објектну репрезентацију стабла. Класа `QueryTreeCellRenderer` има само један метод:

```
public Component getTreeCellRendererComponent(JTree tree,
Object value,boolean sel,boolean expanded,boolean leaf, int
row,boolean hasFocus)
```

и он је задужен за графичку репрезентацију стабала. На пример, у њему се може дефинисати икона која ће се видети када је чвор затворен а која када је чвор отворен. Затим дефинише се назив чвора у зависности од вредности која се налази у параметру `value`.

Успостављање конекције са Z-сервером: Након креирања упита може се приступити претраживању изабране библиотеке. На почетку врши се конекција на изабрани сервер. У едитору је то имплементирано креирањем објекта класе `ZClient` из пакета *jafer* који припада алату JAFER.

Ова класа води рачуна о успостављању конекције на изабрани сервер и о слању упита. Слање упита и преузимање записа имплементирано је методима класе `Client`:

- `public static int search(ZClient client,Node query,String library)`- метод подешава параметре за изабрану библиотеку (параметар `library`) и шаље крирани упит (параметар `query`) позивом методе `public int submitQuery(Node query)` која припада класи `ZClient` из пакета *jafer*. Резултат извршавања овог метода је број погодака пронађених при претраживању.
- `public static Node present(ZClient client,int noPresent)`- метод врши преузимање пронађених записа. Параметар `noPresent` дефинише који запис се преузима док објекат класе ZClient је задужен за извршавање *Present* сервиса дефинисаног у стандарду Z39.50.

Преглед библиотечких записа: Број погодака претраживања описан је случајем коришћења *Prikaz broja pogodaka*. Алат JAFER је тако реализован да када Z39.50 клијент, у овом случају је то овај имплементирани XML едитор за претраживање и преузимање библиографских записа, пошаље *Present* захтев одређен број резултата се смешта у меморијску структуру која је представљена објектом класе `Hashtable` из пакета *java.util*. На тај начин је омогућено да корисник захтева приказ одређених записа без поновног приступања серверу. Уколико неки тражени запис није у меморијској структури, тада ће се поново проследити *Present* захтев, да би се добили наредни записи из скупа погодака.

При имплементацији едитора кориснику је омогућено да прегледа појединачне записе у виду пуног формата (*full format*) и у виду каталошких листића. Да би се запис приказао у пуном формату прво се запис који је представљен као XML документ смести у објектну структуру која је описана класама из пакета *marc21*, а затим се позивом одговарајуће методе из тог пакета запис серијализује у тип података *String* који представља пун формат записа и као такав се приказује на одговарајућој екранској форми.

Предвиђено је да се интеграцијом едитора у ситем БИСИС омогући преглед записа у виду различитих каталошких листића. Систему за генерисање каталошких листића би било потребно проследити само објектну структуру записа коју би он даље могао да обрађује.

У тренутној имплементацији едитора предвиђено је да корисник може претраживати само једну библиотеку у једном тренутку. Међутим минималним изменама би се могло омогућити да корисник из скупа понуђених библиотека селектује више библиотека које жели да претражи

са истим упитом. Измене би се састојале у прилагођавању екранске форме за преглед резултата, тако да за сваки запис постоји и информација из које библиотеке је тај запис преузет (као што је то сада у систему БИСИС).

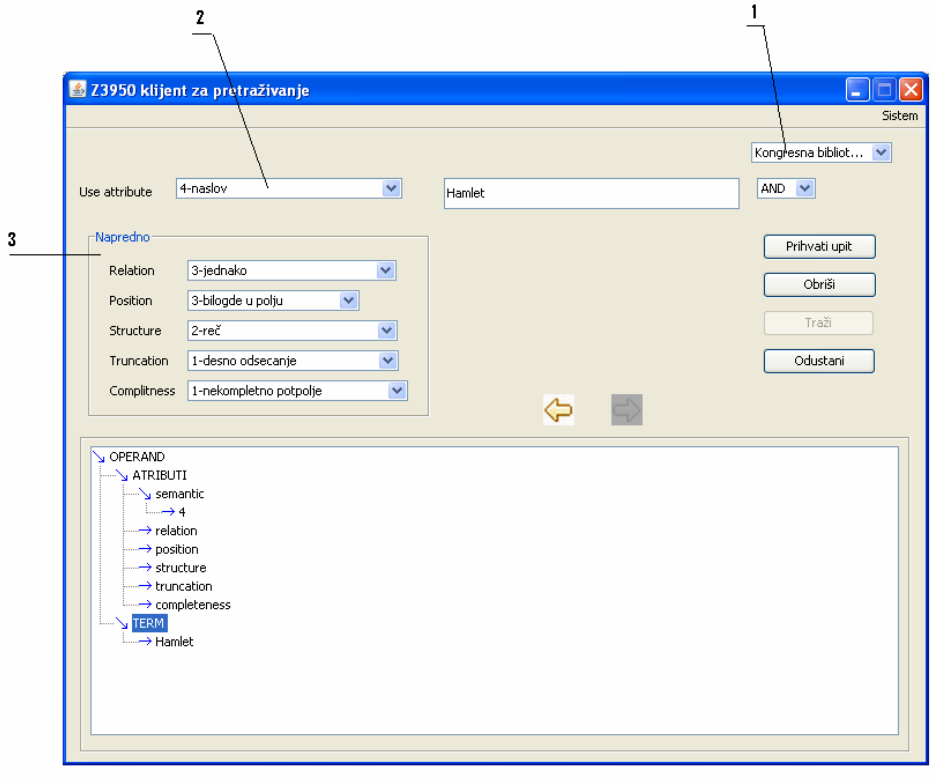
Преузети запис се може снимити у фајл систем у формату XML документа, али би се интеграцијом едитора у систем БИСИС запис могао складиштити у локалну базу података.

Поглавље 6

Опис коришћења система

У овом поглављу је описан начин коришћења едитора за претраживање и приказивање библиографских записа. Употреба едитора приказана је на примерима различитих упита.

Приликом покретања апликације кориснику се отвара почетна екранска форма која је приказана на слици 6.1. На овој екранској форми корисник може изабрати библиотеку коју жели да претражи избором из падајуће листе означене бројем 1. У свим примерима који следи биће изабрана Конгресна библиотека у Вашингтону, и на њој је уједно извршена и верификација система.



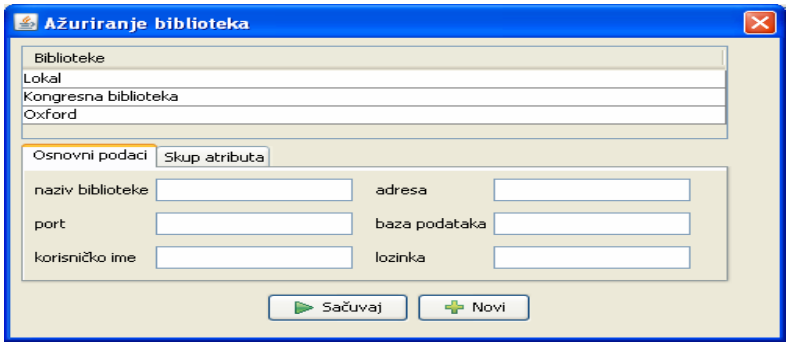
Слика 6.1 Почетна екранска форма

6.1 Конфигурисање параметара библиотеке

Да би се омогућило претраживање по библиотечком фонду неке библиотеке, корисник прво мора изабрати библиотеку из падајуће листе. Уколико библиотека није дата у падајућој листи тада је корисник мора додати избором менија *Sistem* на слици 6.1. На слици 6.2 приказана је екранска форма за додавање нове библиотеке и ажурирање параметара већ постојећих библиотека која се отвара након избора из менија *Sistem*.

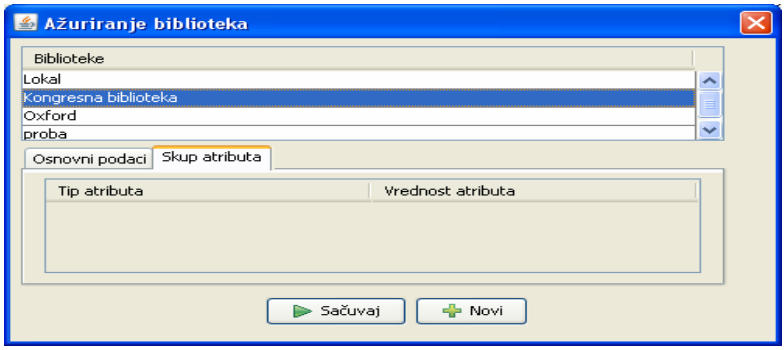
Ова екранска форма има две картице и на слици 6.2 приказана је картица *Osnovni podaci*. Притиском на дугме *Novi* кориснику се омогућава унос потребних параметара за конекцију као што су назив библиотке, адреса, назив базе података и томе слично. Име библиотеке јединствено означава одређену библиотеку. Уколико корисник није задовољан унетим параметрима притиском на дугме *Novi* поништавају се све унете вредности. Притиском на дугме *Sačuvaj* сви унети подаци се чувају.

Кориснику је такође омогућена и промена постојећих података везаних за одређену библиотеку. Селектовањем једне од библиотека из табеле приказане у горњем делу екранске форме (слика 6.2) приказују се и сви остали параметри везани за ту библиотеку. Корисник може да измени неке од параметара и притиском на дугме *Sačuvaj* извршиће се ажурирање. Слично, селектовањем неке библиотеке и притиском типке *Delete* на тастатури врши се брисање селектоване библиотеке.



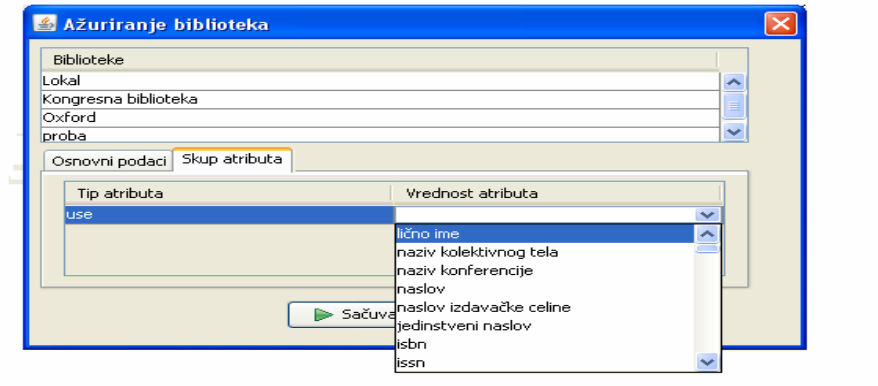
Слика 6.2 Екранска форма за ажурирање параметара библиотеке

Картицом *Skup atributa* дефинисани су атрибути које изабрана библиотека подржава. Изглед екранске форме приказан је на слици 6.3 на којоју се види да је селектована Конгресна библиотека и за коју није додат ни један атрибут.



Слика 6.3 Екранска форма за ажурирање параметара библиотеке- картица *Skup atributa*

Притиском на дугме *Novi* додаје се нови атрибут. Корисник је потребно да из падајуће листе изабере врсту атрибута (*Use, Relation, Stucture, Position,Truncation, Completness*) и за изабрану врсту атрибута изабере одређену вредност из падајуће листе као што је приказано на слици 6.4. Притиском на дугме *Sačuvaj* врши се чување унетих вредности.



Слика 6.4 Картица *Skup atributa* - додавање атрибута

6.2 Претраживање по вредностима *Use* атрибута

У овом одељку на следећем примеру описана је употреба едитора приликом претраживања по вредностима *Use* атрибута.

Пример 1: Пронаћи све библиографске записе који у наслову садрже реч *Hamlet*.

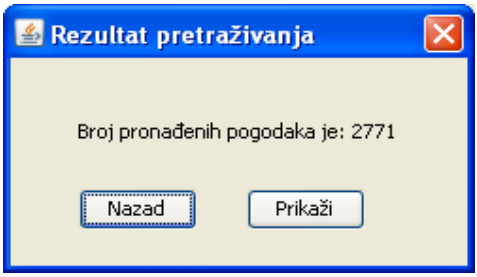
Да би се задовољио постављени задатак корисник прво мора да изабере одговарајући *Use* атрибут из падајуће листе означене бројем 2 на слици 6.1 чија вредност је *4-naslov*, и служи за претраживање по наслову. Ефикасно претраживање обавезује корисника да познаје семантику вредности *Use*

атрибута. Списак вредности *Use* атрибута и семантика појединачних вредности као и њихово мапирање на одговарајућа поља и потпоља библиографског записа дат је у Прилогу 1.

Бројем 3, на слици 6.1 означене су падајуће листе из којих се бирају напредни атрибути из скупа *bib-1* и то ће бити детаљније објашњено у одељку 6.4. Атрибути који су груписани у напредне атрибуте су опциони и уколико их корисник не селектује апликација ће узети подрезумеване вредности тих атрибута које су дефинисане у оквиру саме апликације. У овом конкретном примеру корисник не мора да селектује напредне атрибуте јер постављени упит то не захтева. Терм *Hamlet* који се тражи уноси се у једнолинијски едитор.

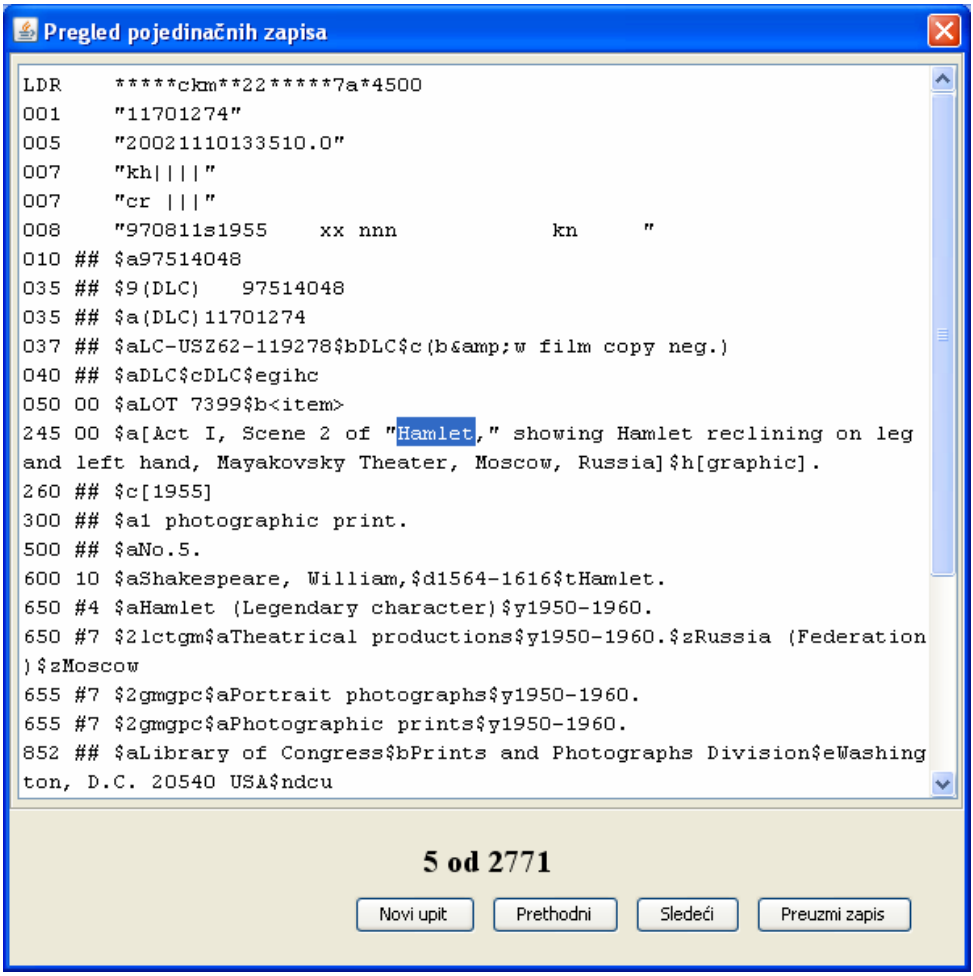
Притиском на дугме *Prihvati upit* врши се формирање упита који је приказан у доњем делу екранске форме на слици 6.1. Формирани упит је графички репрезентован као стабло где у овом примеру имамо коренски елемент *OPERAND* који у себи садржи поделементе којима су описане вредности изабраних атрибута (поделемент *ATRIBUTI*) и вредност терма (поделемент TERM). Уколико корисник није задовољан креираним упитом притиском на дугме *Odustani* обрисаће се целокупни садржај упита који је до тада креиран.

Након притиска на дугме *Traži*, могу се јавити одређене грешке типа одбијена конекција на сервер, сервер не подржава неке атрибуте и томе слично и у том случају ће систем генерисати изузетак а кориснику приказати одговарајући дијалог са поруком о грешци. Уколико није дошло до грешке појављује се прозор (слика 6.5) са информацијом о броју погодака.



Слика 6.5 Прозор са бројем погодака

Притиском на дугме *Nazad* корисник одустаје од прегледа записа и враћа се на почетну екранску форму (слика 6.1). Притиском на дугме *Prikaži* отвара се нова екранска форма за преглед резултата (слика 6.6).



Слика 6.6 Прозор за преглед појединачних записа

Централни део екранске форме представља део за приказ пронађеног записа у пуном формату. Такође постоји информација о редном броју записа који се тренутно гледа. Кориснику је омогућена навигација кроз скуп добијених резултата притиском на дугме *Prethodi* и *Sledeći*, а притиском на дугме *Preuzmi zapis*, библиографски запис се снима у фајл систем у виду XML документа. Притиском на дугме *Novi upit* корисник се враћа на почетну екранску форму (слика 6.1) за постављање новог упита.

Пример једног преузетог записа у виду XML документа дат је у листингу 6.1. Интеграцијом едитора у систем БИСИС притиском на дугме *Preuzmi zapis* отвараће се одговарајућа екранска форма за обраду библиографских записа.

```
</varfield>
  <varfield i1=" " i2=" " id="010">
        <subfield label="a">   97514048 </subfield>
  </varfield>
  <varfield i1=" " i2=" " id="037">
        <subfield label="a">LC-USZ62-119278</subfield>
        <subfield label="b">DLC</subfield>
  </varfield>
  <varfield i1=" " i2=" " id="040">
        <subfield label="a">DLC</subfield>
        <subfield label="e">gihc</subfield>
  </varfield>
 <varfield i1="0" i2="0" id="050">
        <subfield label="a">LOT 7399</subfield>
  </varfield>
<varfield i1="0" i2="0" id="245">
    <subfield label="a">[Act I, Scene 2 of "Hamlet," showing Hamlet   reclining on leg and left hand,
          Mayakovsky Theater, Moscow, Russia]
    </subfield>
    <subfield label="h">[graphic].</subfield>
  </varfield>
<varfield i1=" " i2=" " id="955">
       <subfield label="a">qw05</subfield>
  </varfield>
  <varfield i1=" " i2=" " id="010">
        <subfield label="a">   97514048 </subfield>
  </varfield>
  <varfield i1=" " i2=" " id="037">
        <subfield label="a">LC-USZ62-119278</subfield>
        <subfield label="b">DLC</subfield>
        <subfield label="c">(b&amp;amp;w film copy neg.)</subfield>
  </varfield>
  <varfield i1=" " i2=" " id="040">
        <subfield label="a">DLC</subfield>
        <subfield label="e">gihc</subfield>
  </varfield>
  <varfield i1="0" i2="0" id="050">
        <subfield label="a">LOT 7399</subfield>
        <subfield label="b">&lt;item&gt;</subfield>
  </varfield>
  <varfield i1="0" i2="0" id="245">
    <subfield label="a">[Act I, Scene 2 of "Hamlet," showing Hamlet   reclining on leg and left hand,
          Mayakovsky Theater, Moscow, Russia]
    </subfield>
    <subfield label="h">[graphic].</subfield>
  </varfield>
  <varfield i1=" " i2=" " id="260">
        <subfield label="c">[1955]</subfield>
  </varfield>
```

Листинг 6.1 Преузети библиографски запис

```
<varfield i1=" " i2=" " id="500">
        <subfield label="a">No. 5.</subfield>
 </varfield>
   <varfield i1="1" i2="0" id="600">
        <subfield label="a">Shakespeare, William,</subfield>
        <subfield label="d">1564-1616</subfield>
        <subfield label="t">Hamlet.</subfield>
   </varfield>
   <varfield i1=" " i2="4" id="650">
        <subfield label="a">Hamlet (Legendary character)</subfield>
        <subfield label="y">1950-1960.</subfield>
   </varfield>
    <varfield i1=" " i2="7" id="650">
        <subfield label="a">Theatrical productions</subfield>
        <subfield label="z">Russia (Federation)</subfield>
        <subfield label="z">Moscow</subfield>
        <subfield label="y">1950-1960.</subfield>
        <subfield label="2">lctgm</subfield>
   </varfield>
   <varfield i1=" " i2="7" id="655">
        <subfield label="a">Portrait photographs</subfield>
        <subfield label="y">1950-1960.</subfield>
        <subfield label="2">gmgpc</subfield>
   </varfield>
   <varfield i1=" " i2="7" id="655">
        <subfield label="a">Photographic prints</subfield>
        <subfield label="y">1950-1960.</subfield>
        <subfield label="2">gmgpc</subfield>
   </varfield>
  <varfield i1=" " i2=" " id="852">
        <subfield label="a">Library of Congress</subfield>
        <subfield label="b">Prints and Photographs Division</subfield>
        <subfield label="e">Washington, D.C. 20540 USA</subfield>
        <subfield label="n">dcu</subfield>
   </varfield>
   <varfield i1="4" i2="1" id="856">
        <subfield label="3">b&amp;amp;w film copy neg.</subfield>
        <subfield label="d">cph</subfield>
        <subfield label="f">3c19278</subfield>
        <subfield label="u">http://hdl.loc.gov/loc.pnp/cph.3c19278
    </subfield>
   </varfield>
   <varfield i1=" " i2=" " id="953">
        <subfield label="a">TE01</subfield>
   </varfield>
   <varfield i1=" " i2=" " id="985">
 <subfield label="a">pp/cph</subfield>
 </varfield>
```

Листинг 6.1 (наставак)

```
  <varfield i1=" " i2=" " id="991">
        <subfield label="b">c-P&amp;P</subfield>
        <subfield label="h">LOT 7399</subfield>
        <subfield label="i">&lt;item&gt;</subfield>
        <subfield label="v">obj.</subfield>
        <subfield label="w">MUMS VM File</subfield>
  </varfield>
  <varfield i1=" " i2=" " id="991">
        <subfield label="b">c-P&amp;P</subfield>
        <subfield label="h">LC-USZ62-119278 (b&amp;film copy neg)</subfield>
        <subfield label="w">MUMS VM File</subfield>
  </varfield>
</oai_marc>
```

Листинг 6.1 (наставак)

6.3 Претраживање употребом логичких оператора

Имплементирани едитор омогућава употребу логичких оператора *AND*, *NOT* и *OR* и њихов избор врши се избором из падајуће листе.

Значење логичких оператора је следеће:
Нека је садржај једног операнда Текст1 а другог Текст2. Тада је:

1. Оператор *AND* у упиту Текст1 AND Текст2 повезује два елемента упита на такав начин да издваја оне библиографске записе који поседују први наведени садржај (Текст1), а истовремено поседују и други (Текст2).

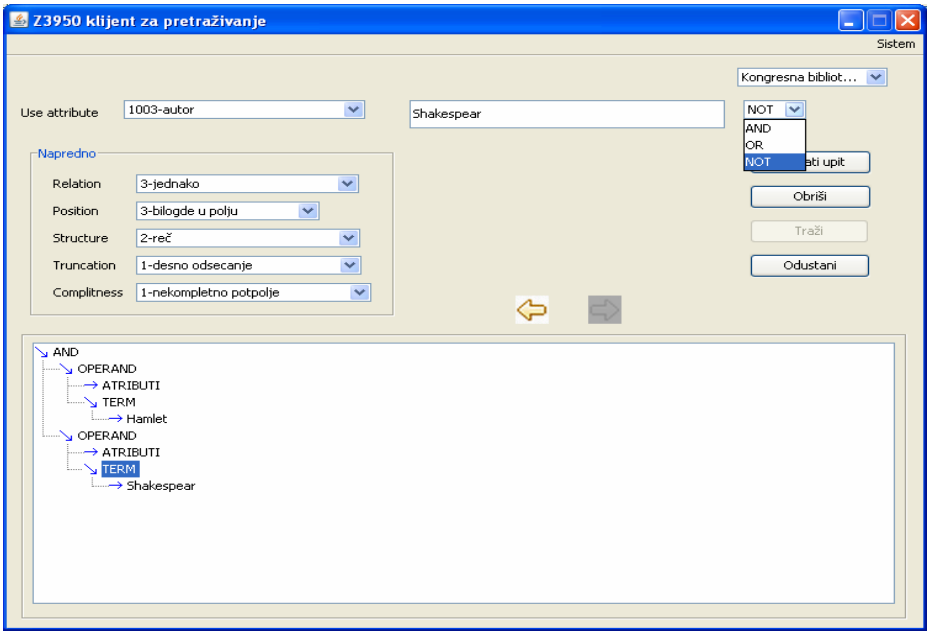
2. Оператор *NOT* у упиту Текст1 NOT Текст2 повезује два елемента упита на такав начин да издваја оне записе који поседују први наведени садржај (Текст1), а истовремено не поседују други (Текст2).

3. Оператор *OR* у упиту Текст1 OR Текст2 повезује два елемента упита на такав начин да издваја оне јединице које поседују први наведени садржај (Текст1), или поседују други (Текст2), а могу и оба.

Пример 2: Наћи све записе који у наслову садрже реч *Hamlet*, аутор је *Shakespear* (Шекспир) и нису на енглеском језику.

Да би се формирао упит којима се решава постављени задатак, корисник прво мора креирати операнд где је *Use* атрибут *4-naslov*, а терм који се тражи је *Hamlet*. Ово је досада исто као и у примеру 1, једино што још треба да се уради је да се из падајуће лист изабере оператор AND којим ће се повезати следећи операнд. Када је формиран први операнд (притиском на дугме *Prihvati upit*), корисник треба да креира нови операнд. Потребно је да изабере *Use* атрибут *1003-autor*, који означава аутора књиге и унесе терм *Shakespear*. Пошто корисник не жели књиге на енглеском језику приликом

формирања другог операнда било је неопходно да се изабере логички оператор NOT. Формирани операнд приказан је на слици 6.7.

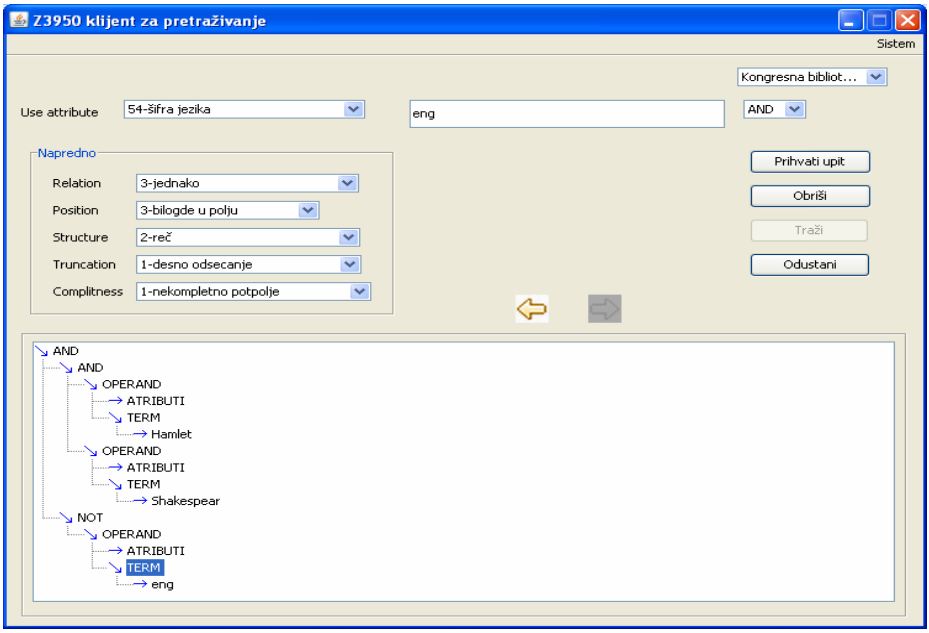


Слика 6.7 Други операнд упита

На сличан начин формира се и трећи операнд, потребно је селектовати *Use* атрибут *044-šifra jezika*, који представља шифру језика на ком је написана књига, по шифарнику језика који се примењује за обраду записа по MARC 21 формату шифра за енглески језик је *eng* [MarcCode]. На слици 6.8 приказан је формирани операнд.

Кориснику је омогућено кретање кроз формиране операнде притиском на дугмад ⇦ и ⇨ . На тај начин могуће је изменити већ креирани операнд и притиском на дугме *Prihvati upit* измењени операнд се ажурира. Такође, притиском на дугме *Obriši* брише се операнд који је тренутно селектован.

Графичи приказ упита дат је у доњем делу екранске форме на слици 6.8. Графички приказ одговара префиксној нотацији упита. Односно прво следи оператор па затим операнди. Тако у овом примеру коренски елемент у стаблу означава логички оператор који повезује операнде а поделементима OPERAND приказани су конкретни операнди са вредностима изабраних атрибута (поделемент ATRIBUTI) и изразом који се тражи (поделемент TERM).



Слика 6.8 Трећи операнд упита

Притиском на дугме *Traži* врши се претраживање изабране библиотеке и даљи принцип рада едитора је описан у првом примеру.

6.4 Напредно претраживање

У овом одељку су приказани примери упита који укључују напредне атрибуте из скупа *bib*-1, и који су груписани у оквиру секције *Napredno* (слика 6.9). Сваки атрибут из овог дела допуњује вредност *Use* атрибута, што значи да се не могу селектовати само вредности ових атрибута а да при томе није селектована и вредност за *Use* атрибут.

Слика 6.9 Напредни атрибути из скупа *bib*-1

Пример 3: Пронаћи библиографске записе где је аутор Shakespear (Шекспир), а књиге су објављене у периоду од 2000. до 2007. године, укључујући и године које представљају границе интервала.

Опис упит који корисник треба да постави дат је у табели 6.1

Из табеле се види да упит има три операнда и сви су повезани логички оператором AND. Први операнд има само *Use* атрибут чија вредност означава аутора. Пошто треба да се дефинише период у ком су књиге издате, искоришћена су два операнда који имају *Use* атрибут *031-datum izdavanja* који представља датум издавања књиге. Међутим, уколико се не би за сваки операнд дефинисао и атрибут *Relation*, добио би се следећи упит: „Нађи све књиге које је написао Шекспир и које су издате 2000. и 2007. године“ (што је различито од постављеног упит).

	тип атрибута	**вредност атрибута**	**терм**
Први операнд	Use	1003-autor	Shakespear
оператор	AND		
Други операнд	Use	031-datum izdavanja	2000
	Relation	4-veće ili jednako	
оператор	AND		
Трећи операнд	Use	031- datum izdavanja	2007
	Relation	2-manje ili jedanko	

Табела 6.1 Табеларни приказ упит за пример 3

Из тих разлога за други операнд је дефинисан *Relation* атрибут са вредношћу *4-veće ili jednako*. *Relation* атрибут има задатак да допуни значење *Use* атрибута и у овом конкретном примеру то значи да година издања мора бити већа или једнака од 2000. Слично је за трећи операнд дефинисан *Relation* атрибут са вредношћу *2- manje ili jedanko* и на тај начин је реализован критеријум да је година издања мања или једнака од 2007.

Пример 4: Пронаћи књиге које у наслову садрже фразу *Thinking in Java* (табела 6.2).

Овај упит се састоји само од једног операнда који има два атрибута: *Use* атрибут са вредношћу *4-naslov* и *Structure* атрибут са вредношћу *1-fraza*. Атрибутом *Structure* је дефинисан услов да треба наћи тачну фразу, да је уместо ове вредности селектована вредност *2-reč*, тада би били пронађени и записи који у наслову садрже све наведене речи, али које не морају бити у наведеном редоследу. Такође овим упитом су пронађени и записи који садрже тачну фразу у наслову али имају поред тога и неке друге изразе. Такав један случај је запис који у наслов садржи израз *Thinking in Java programming language*.

	тип атрибута	**вредност атрибута**	**терм**
Први операнд	Use	4-naslov	*Thinking in Java*
	Structure	1-fraza	

Табела 6.2 Табеларни приказ упит за пример 4

Уколико корисник жели да пронађе само записе који садрже тачан израз потребно је да се постави вредност *Trunction* атрибута на *100-bez odsecanja*.

Пример 5: Пронаћи књиге које у предметној одреднице садрже изразе које почињу са речју *Method* (табела 6.3).

Тражени упит има само један операнд при чему *Use* атрибут има вредност *21-predmetna odrednica* која означава предметну одредницу, и додат је и атрибут *Truncation* са вредношћу *1- desno odsecanje*. Атрибут *Truncation* служи да омогући претраживање по почетку речи, крају речи и слично. У овом примеру је искоришћена вредност *1- desno odsecanje* што значи да предметна одредница књиге мора да почне низом карактера *Method*. Формирани упит приказан је у табели 6.3.

	тип атрибута	**вредност атрибута**	**терм**
Први операнд	Use	21-predmetna odrednica	*Method*
	Truncation	1- desno odsecanje	

Табела 6.3 Табеларни приказ упит за пример 5

Овако постављеним упитом биће пронађени и записе који у предметној одредници садрже и речи као што су: *Methods*, *Methodist*, *Methodology* и томе слично. Да је било потребно да се пронађе само запис који садржи тачну реч *Method* тада би *Truncation* атрибут имао вредност *100-bez odsecanja*.

Са одговарајућим променама вредности атрибута *Truncation* могу се добити записи код којих на пример наслов почиње са задатом речју, наслов се завршава са задатим низом карактера или се задати низ карактера налази у средини израза који представља наслов.

Прилог 1

Вредности типова атрибута *bib-1*

У овом прилогу дат је скуп вредности атрибута *bib-1*. То су следећи атрибути:

- *Use Attributes*
- *Relation Attributes*
- *Truncation Attributes*
- *Structure Attributes*
- *Completeness Attributes*
- *Position Attributes*

Опис и примери ових атрибута дати су у одељку 2.4.1. У наведеним табелама дате су усвојене вредности за све наведене типове атрибута.

Табела 1: Списак Use атрибута и њихово мапирање на одговарајућа поља и потпоља MARC 21 формата

Вредност	**Нумеричка вредност**	**Потпоља у MARC 21 формату**
Лично име	1	100, 400, 600, 700,800
Назив колективног тела	2	110, 410, 610, 710,810
Назив конференције	3	111, 411, 611, 711, 811
Наслов	4	130, 21X-24X, 440, 490, 730, 740, 830, 840 и потпоље $t у следећим пољима: 400, 410, 410, 600, 610, 611, 700, 710, 711, 800, 810, 811
Наслов издавачке целине	5	440, 490, 830, 840, потпоље $t из следећих поља: 400,410, 411, 800, 810, 811
Јединствени наслов	6	130, 240, 730, потпоље $t из следећих поља: 700,710, 711
ISBN	7	020
ISSN	8	022, 4XX$x, 7XX$x

Идентификатор записа у Конгресној библиотеци	9	010, 011
Идентификатор записа у Британској Националној Библиографији	10	015
Идентификатор записа у Националној библиотеци Француске	11	015
Идентификатор записа у локалном систему	12	001, 035
Дјуијева децимална класификација (DDK)	13	082
UDK класификација	14	080
Bliss класификација	15	
Сигнатура Конгресне библиотеке	16	050
Сигнатура Националне библиотеке за медицину	17	060
Сигнатура Националне пољопривредне библиотеке	18	070
Класификациони број из класификације метематичких одредница	19	
Сигнатура у локалном систему	20	
Предметна одредница	21	600, 610, 611, 630, 650, 651, 653, 654, 655, 656, 657, 69X

Одредница Rameau	22	Предметна одредница из *Repertoire d'authorite de matieres encyclopedique unifie* креирана од стране Националне библиотеке Француске
BDI предметна одредница	23	Предметне одреднице из *Bibliotek Dokumentasjon Informasjon* предметног каталога који користи пет Скандинавских земаља (Данска,Финска, Исланд, Норвешка,Шведска)
INSPEC предметна одредница (Предметне одреднице из *Information Services for the Physics and Engineering Communities* предметног каталога)	24	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 2
MESH предметна одредница (Предметне одреднице из предметног каталога Националне медицинске библиотеке)	25	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 2
PA предмет (Предметне одреднице из нормативне датотеке *Psychological Index Terms*)	26	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 2
Предметна одредница Конгресне библиотеке	27	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 0
RVM предметна одредница (Предметне одреднице из нормативне датотеке *Repertoire des vedettes-matiere*)	28	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 6

Локални предметни каталог	29	
Датум	30	005, 008/00-05, 008/07-10, 260$c, 008/11-14, 033
Датум издавања	31	008/07-10, 260$c,046, 533$d
Датум набавке	32	541$d
Кључни наслов	33	222
Надређени наслов	34	243
Упоредни стварни наслов	35	Поље 246 где други индикатор има вредност 1
Корични стварни наслов	36	Поље 246 где други индикатор има вредност 4
Наслов на споредној насловној страни	37	Поље 246 где други индикатор има вредност 5
Наслов над текстом	38	Поље 246 где други индикатор има вредност 6
Текући наслов	39	Поље 246 где други индикатор има вредност 7
Хрптени наслов	40	Поље 246 где други индикатор има вредност 8
Остале варијанте наслова	41	212, поље 246 где други индикатор има вредност 3, 247, 740
Претходни наслов	42	247, 780
Скраћени наслов	43	210, 211, 246
Проширени наслов	44	214, 246
Предметне одреднице из нормативне датотеке *PREserved Context Index System*	45	
Предметне одреднице из нормативне датотеке *Regeln fur den Schlagwortkatalog*	46	
Предметна пододредница	47	6XX$x, 6XX$y, 6XX$z

Број националне библиграфије	48	015
Број обавезног примерка	49	017
Класификациони број владиних докумената	50	086
Издавачки број	51	028
Број записа у Немачкој Националној библиотеци	52	015
Локални број (локација на полици) додељен од стране система	53	
Код језика	54	008/35-37, 041
Код за географско подручје	55	043
Сигла	56	040, 852$a
Име и наслов	57	100/2XX, 110/2XX, 111/2XX, потпоља $a и $t у следећим пољима: 400, 410, 411, 600, 610, 611, 700, 710, 711, 800, 810, 811
Географско име	58	651
Место издавања	59	008/15-17, 260$a
Коден	60	030
Генерисање микрооблика	61	007/11
Абстракт	62	520
Напомена	63	5XX
Аутор-наслов	1000	100/2XX, 110/2XX, 111/2XX, потпоља $a и $t из поља: 400,410,411, 700, 710, 711, 800, 810, 811
Тип записа	1001	Лидер/06
Име	1002	100, 110, 111, 400, 410, 411, 600, 610, 611, 700, 710, 711, 800, 810, 811

Аутор	1003	100, 110, 111, 400, 410, 411, 700, 710, 711, 800, 810, 811
Аутор-лично име	1004	100, 400, 700, 800
Аутор-име колективног тела	1005	110, 410, 710, 810
Аутор-назив конференције	1006	111, 411, 711, 811
Број из стандарда (ISSN, CODEN...)	1007	010, 011, 015, 017, 018, 020, 022, 023, 024,025, 027, 028,030, 035, 037
Предметна одредница за дечију књигу	1008	600, 610, 611, 630, 650, 651 и вредност другог индикатора за наведена поља је 1
Лично име као предметна одредница	1009	600
Тело текста	1010	Користи се за претраживање оног дела текста за које сервер сматра да представља тело текста
Датум/време креирања записа	1011	008/00-05
Датум/време последње модификације	1012	005
Идентификациони број нормативне датотеке	1013	Користи се код full-text претраживања да би се серверу дефинисао формат документа који треба да буде враћен клијенту. Атрибут не садржи само информацију о формату већ и податак о систему који је креирао тај формат.
Концепт-текст	1014	Застарело
Концепт-референце	1015	Застарело
Било који атрибут	1016	Претраживање се врши по свим Use атрибутима које сервер подржава
Избор сервера	1017	Сервер самостално одлучује по којим Use атрибутима ће се претраживати.

Издавач	1018	260$b
Креатор оригиналног каталошког записа	1019	008/39, 040
Уредник	1020	100 $a, 700 $a и у потпољу $e одговарајућег поља се налази вредност 'ed.'
Код библиографског нивоа	1021	*Лидер/07*
Географска класификација	1022	052
Напомена о цитирању/референцама	1023	510
Кодирани картографски математички подаци	1024	034
Музички кључ	1025	Потпоље $r у следећим пољима: 130, 240, 243, 630, 700, 730
Наслов часописа	1026	247, 780, 785
Број извештаја	1027	027, 088
Извор набавке	1028	037
Број поглавља/секције/дела	1030	Потпоље $n у следећим пољима:130, 240, 243, 630, 700, 730
Тип материјала	1031	*Лидер/06, 007, 008, 502*
Јединствени број записа у локалном систему	1032	
Наслов матичне јединице	1033	773$t
Природа дела	1034	008/24-27
Било где	1035	Разлика између атрибута *Било који* и атрибута *Било где* је у томе што овај други атрибут представља много скупљу операцију.
Аутор-Наслов-Одредница	1036	1XX, 2XX, 4XX, 6XX, 7XX, 8XX

Табела 2 *Relation* атрибути (атрибути чија је нумеричка вредност од 1 до 6 имају стандардну семантику која се примењује и за релације између бројева)

Вредност	**Нумеричка вредност**	**Значење**
мање	1	
мање или једнако	2	
једнако	3	
веће или једнако	4	
веће	5	
није једнако	6	
фонетско поклапање	100	Два терма се морају фонетски поклопити иако могу постојати разлике у писању тих термова.
Корен речи	101	Проналази све речи које имају исту коренску основу као и терм који је задат.
Релевантно	102	Користи се за проналажење записа који су релевантни за дати терм у оквиру неког *Use* атрибута. Алгоритам по ком се утврђује шта је релевантно за неки терм дефинише сам Z-сервер.
Увек се поклапа	103	Избором атрибута *увек се поклапа*, сервер занемарује унети терм и проналази све записе који имају неку вредност у изабраном *Use* атрибуту. Ако је *Use* атрибут *наслов* биће селектовани сви записи који имају било коју вредност за овај атрибут.

Табела 3 *Trunaction* атрибути

Вредност	Нумеричка вредност	Значење
десно одсецање	1	Биће пронађене све речи које су исте као дати терм или све речи које почињу исто као дати терм.
Лево одсецање	2	Биће пронађене све речи које су исте као дати терм или које се завршавају са датим термом.
Лево и десно	3	Биће понађене речи које су исте са датим термом и све речи које могу бити дуже али у себи садрже дати терм, било на почетку, крају или у средини.
Без одсецања	100	Биће проналађене речи које су идентичне датом терму.
Знак # у терму	101	Ово значи да терм који се тражи садржи специјалан карактер # који се мора на посебан начин обрадити. Овај карактер замењује један или више карактера у терму. На пример, ако је терм "Уч#е" биће пронађене речи као што су учење, учионице, учитељице.
Регуларни израз-1	102	Означава да је терм у виду регулатног израза који је дефинисан стандардом *IEEE 1003.2 Volume 1,Section 2.8 „Regular Expression Notation"*.
Регуларни израз-2	103	Терм је у виду регуларног израза који је дефинисан у оквиру локалног система.

Табела 4 *Structure* атрибути

Вредност	Нумеричка вредност	Значење
фраза	1	Више речи раздвојених празним карактером. Приликом претраживања тражи се тачна вредност терма и речи морају бити у задатом редоследу. За додатна филтрирања скупа резулта могу се користити и остали типови атрибута, као што су *Truncation* и *Completness*.
Реч	2	Терм се састоји од низа карактера и не садржи празне карактере.
Кључ	3	Кључ представља низ карактера који су преузети из неких других речи које су индексиране, али кључ не мора представљати целу реч. Уколико се терм састоји од више речи сваки део кључа не би требало да је дужи од 6 карактера. На пример ако корисник изабере кључ за структуру а унесе терм «*coplan rodeo*», систем може вратити погодак «*Copland, Aaron, 1900- Rodeo*» и слично.
Година	4	Терм се састоји од четири цифре.
Нормализовани датум	5	Датум, месец и година у нотацији дефинисаној по стандарду *Generalized Time in ASN.1 (ISO 8824)* стим да је једино обавезно унети четири цифре за годину.
Листа речи	6	Представља листу речи које су раздвојене празним карактером. Ред речи у листи се не мора поштовати, а везе између речи су дефинисане од стране Z-сервера. Остали типови атрибута се примењују на сваку појединачну реч из листе.

Ненормализовани датум	100	Датум, месец и година, терм није структуиран.
Име (нормализовано)	101	Терм је стуктуиран, на пример зна се тачан редослед речи и синтакса (нпр. *"презиме, име"* где је и зарез део синтаксе). Резултат претраживања зависи од имплементације на самој серверској страни.
Име (ненормализовано)	102	Терм који се тражи је неструктуирано име, на пример "*име презиме*". Резултат претраживања зависи од имплементације на самој серверској страни и могу се добити различити резултати у односу када се као вредност користи фраза. На пример, ако је корисник изабрао ову вредност и унео терм «john smith» серверска страна то може третирати као «smith, j#», а не као тачну фразу.
Структура	103	Терм има структуру, односно формат који је или условљен самим *Use* атрибутом или је тај формат дефинисан од стране сервера.
Urx	104	Терм је идентификатор документа, на пример идентификатор који је преузет из Z39.50 URL.
Слободан текст	105	Терм је текст који је унео корисник.
Документ-текст	106	Терм је текст који је издвојен из документа.
Локални број	107	Број који је значајан за серверску страну
низ карактера	108	Цео терм се посматра као низ карактера а не као скуп појединачних речи.
Нумеричка вредност	109	Терм представља број.

Табела 5 *Position* атрибути

Вредност	Нумеричка вредност	Значење
први у пољу	1	Терм који се тражи мора бити први податак у пољу.
Први у потпољу	2	Терм који се тражи се може појавити у било ком потпољу али мора бити први податак у оквиру тог потпоља.
Било где у пољу	3	Терм који се тражи се може јавити било где у оквиру поља.

Табела 6 *Completness* атрибути

Вредност	Нумеричка вредност	Значење
непотпуно потпоље	1	Поред терма који се тражи могу се наћи и друге додатне речи у пољу и потпољу у коме се налази и тражени терм.
Комплетно потпоље	2	Ниједна друга реч осим траженог терма се не сме налазити у потпољу у ком се налази терм. Међутим, друге речи се могу налазити у осталим потпољима у оквиру датог поља.
Комплетно поље	3	Ниједна друга реч осим траженог терма се не сме налазити у пољу у ком се налазе терм.

Литература

[AttributeSets] Attribute Sets, http://www.loc.gov/z3950/agency/defns/oids.html#3 [30. јул 2007.]

[Bath] The Bath Profile, http://www.collectionscanada.ca/bath/tp-bath2-e.htm [30. јул 2007.]

[Bib1] Bib-1 Attribute Set, http://www.loc.gov/z3950/agency/defns/bib1.html [10. јул 2006.]

[BISIS03] Д. Сурла, З. Коњовић, Б. Пуповац, Б. Милосављевић, М. Видаковић, Т. Тошић, Т. Зубић. *Упутство за коришћење библиотечког софтверског система БИСИС вер. 3*, Група за Информационе технологије, Нови Сад, 2003.

[BISIS04] Монографија *Дистрибуирани библиотечки информациони систем БИСИС*, уредници Д. Сурла, З. Коњовић, Група за Информационе технологије, Нови Сад, 2004.

[Boberić06] Боберић, Д., Сурла, Д., *XML SCHEMA OF QUERY LANGUAGE DEFINED BY Z39.50 STANDARD*, рад изложен на XVII Conference on Applied Mathematics "PRIM 2006", Kragujevac, 2006, зборник апстракта, стр. 19.

[**Boberić07a**] Боберић, Д., *ИМПЛЕМЕНТАЦИЈА ЕДИТОРА ЗА ПРЕУЗИМАЊЕ БИБЛИОГРАФСКИХ ЗАПИСА*, зборник радова YU INFO 2007, (CD), Копаоник

[**Boberić07b**] Боберић, Д., *ЕДИТОРИ ЗА ПРЕУЗИМАЊЕ БИБЛИОГРАФСКИХ ЗАПИСА ПО Z39.50 СТАНДАРДУ*, зборник радова YU INFO 2007, (CD), Копаоник

[**Boberić07c**] Боберић, Д., *XML едитор за преузимање библиографских записа*, магистарска теза, Природно-математички факултет, Нови Сад 2007

[BookWhere] BookWhere, WebClarity Software Inc, http://www.bookwhere.com [10. јул 2006.]

[Budimir04] Будимир, Г., Сурла Д., *Систем за контролу квалитета XML библиографских записа*, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2004.

[CANMARC] Canmarc, http://www.collectionscanada.ca/marc/index-e.html [10. јул 2007.]

[CCL] Common Command Language, http://www.indexdata.dk/yaz/doc/tools.tkl#CCL [12. септембар 2006.]

[CopyCat] Surpass Copycat, http://www.surpasssoftware.com/copycat.htm [10. јул 2006.]

[DANMARC] Danmarc, http://www.kat-format.dk/ [15. јул 2007.]

[Dimić07a] Димић Б., *XML едитор за обраду библиографске грађе*, магистарска теза, Природно-математички факултет, Нови Сад 2007

[Dimić07b] Димић, Б., Милосављевић Б., *Контрола библиографских записа по UNIMARC стандарду*, CD-ROM YU INFO 2007, Копаоник 2007.

[DOM] M. Akif, S.Brodhead, A. Cioroianu, J. Hart, E. Jung, D. Writz**,** *Java XML Programmer's Reference*, Wrox ress Ltd, 2001.

[Eclipse] Eclipse Community, http://www.eclipse.org/ [15. јун 2006.]

[EndNote] EndNote, http://www.endnote.com [10. јул 2006.]

[GILS] Application Profile For The Government Information Locator Service (GILS), http://www.gils.net/prof_v2.html [17. јул 2007.]

[INNOPAC] Innovative's Millennium, www.iii.com [15. октобар 2006.]

[ISO2709] *Format for Information Exchange* (ISO 2709), http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=7675&COMMID=&scopelist= [12.септембар 2006.]

[JAFER] Java Access For Electronic Resources, http://www.jafer.org [10. мај 2006.]

[Jakšić04] Jakšić, M., Mijić, V., Surla, D., *XML dokument UNIMARC formata*, PMF, Univerzitet u Novom Sadu, 2004.

[Java] Java programming language, http://www.java.sun.com

[MARC21] Marc Standards, http://www.loc.gov/marc/ [12.mart 2007.]

[MarcEdit] MarcEdit, http://www.oregonstate.edu/~reeset/marcedit/html/index.html [10. јул 2006.]

[MARCSlim] Marc 21 XML Schema, http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd [26.maj 2006]

[Mijić03] Мијић, В., *XML едитор за опис UNIMARC формата,* магистарска теза, Природно-математички факултет, Департман за математику и

информатику, Нови Сад, 2003.
http://diglib.ns.ac.yu/ndltd/docs/set1/ndltd133/MijicVMagistarskaTeza.pdf

[Milosavljević07] Milosavljević B., Dimić B., *XML schema of UNIMARC format variant and bibliographic record in BISIS software system*, NSJOM, Vol. 37, No.1, 2007, pp 115-128.

[OAIMARC] Open Archives Initiative,
http://www.openarchives.org/OAI/oai_marc.xsd [24. април 2006]

[Pict1] *Search end Retrieval Using Non-Proprietary Standards-Based Communications Protocol Z39.50*, K. Trichkov, M. Tsenov, E. Trichkova, Department Hierarchical Systems, Institute of Computer and Communication-Bulgarian, Academy of Sciences, Acad. G. Bonchev str. bl.2, 1113 Sofia, Bulgaria, http://isdg.fmi.uni-sofia.bg/reports/trichkov.pdf [17. јул 2007]

[Rađenović06a] Rađenović J., Milosavljević B., Surla D., *Generating catalog cards by the BISIS library software*, INFOTEKA, 7(2006)1-2, str. 61-74

[Rađenović06b] Рађеновић Ј., Сурла, Д., Милосављевић Б., *Софтверски пакет за генерисање библиотечких каталошких листића* (монографија), Иновациони центар за електронске библиотеке и архиве, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2006.

[RECS] Record Syntaxes supported by Z39.50 protocol,
http://www.loc.gov/z3950/agency/defns/oids.html [17. јул 2007]

[RLQ] Ranked List Query Specification,
http://www-nlpir.nist.gov/projects/rlq/RLQ_spec_Feb96_ZIG.html
[12 септембар 2006.]

[RLQANS1] ASN.1 for Ranked List Query,
http://www-nlpir.nist.gov/projects/rlq/RLQ_asn1_Feb96_ZIG.html
[12. септембар 2006.]

[Servers] Z3950 servers, http://isbnsearch.sourceforge.net/servers.html
[15. октобар 2006.]

[SIRSI] Classic Z39.50 Server Configuration, SirsiDynix,
http://www.sirsidynix.com/Resources/Pdfs/Company/Z3950_Classic.pdf
[15. октобар 2006.]

[SRWWSDL] Search & Retrieve Web Service,
http://www.loc.gov/standards/sru/xml-files/srw-types.xsd, [01.септембар 2007]

[Statistic] Z39.50 Target Statistic, http://libraryoftexas.org/zspy/stat.php
[01.септембар 2007]

[Surla] Сурла, Д., Коњовић, З., Милосављевић, Б., Зарић, М., Сладић, Г., Протић, З., Комазец, С., Окановић, Д., Приказ реализације мрежне дигиталне библиотеке докторских, магистарских и дипломских радова. *Инфотека*, 5 (1-2), 2004.

[Swing] *The swing Tutorial*. Sun Mycrosystems,Inc,1995-2003. http://java.sun.com/docs/books/tutorial/uiswing. [10.септембар 2005]

[Škrbić04] Škrbič, S., Surla D., *Upravljanje XML bibliogarfskim zapisima u XML tehnologiji,* XXXI Simpozijum o operacionim istraživanjima SYM-OP-IS 2004, Iriški Venac 2004. pp133-136

[Škrbić05] Škrbić, S., Surla D., *Obrada bibliografske građe u XML native tehnologiji*, PMF, Univerzitet u Novom Sadu, 2005.

[Tag] Z39.50 Tagsets, http://cheshire.berkeley.edu/tagsets.html [20. јул 2007.]

[UML2.0] Unified Modeling Language: Superstructure, version 2.0, 2004, http://www.omg.org [13. децембар 2006.]

[UKMARC] *The UKmarc Exchange Record Format*. The British Library, National Bibliographic Service. http://www.bl.uk/services/bibliographic/marc/marcexchange.html [10. јул 2007.]

[USMARC] *MARC for library use : understanding integrated USMARC*, Walt Crawford, Boston : G.K. Hall, 1989

[VOYAGER] LC Z39.50/SRW/SRU Server Configuration Guidelines, http://www.loc.gov/z3950/lcserver.html [15. октобар 2006.]

[Vulić] Вулић, Т., Билић, Г., *Ретроспективна конверзија каталошких листића у формат YUMARC,* Природно-математички факултет, Институт за математику, Нови Сад, 1997.

[Willow] Java Willow standalone application, http://www.washington.edu/willow/ [10. јул 2006.]

[XER] XML Encoding Rules (XER), http://asf.gils.net/xer/ [18. март 2007.]

[XML] M. Birbeck, J. Diamond, J. Duckett и други, *Profesional XML 2nd Edition*, Wrox Press Ltd, 2001.

[XMLBeans] Learn about XMLBeans, http://xmlbeans.apache.org/ [18. јул 2007.]

[XMLSchema] XML Schema Part 0: Primer Second Edition, W3C Recommendation 28. October 2004, http://www.w3.org/TR/xmlschema-0/ [11.новембар 2006.]

[Z3950] Information Retrieval (Z39.50):Application Service Definition andProtocol Specification http://www.niso.org/standards/resources/Z39-50.pdf [15. март 2006.]

[Zarić03] Зарић, М. Сурла, *Д.*, Java implementation of the Protocol for Metadata Harvesting. In *Proceedings of the 3d International Conference on Informatics and Information Technology,* Bitola 2003.

[Zarić04a] Зарић, М. Сурла, *Д*, Metadata Dissemination using OAI-PMH. In *Proceedings of the International Conference on Distributed Library Information Systems,* Охрид, 2004.

[Zarić04b] Зарић, М. Сурла, Имплементација протокола за прикупљање метаподатака у мрежи дигиталних библиотека. *Инфотека 5 (1-2):пп. 99-112*

[ZEBRA] Index Data, Zebra http://indexdata.dk/zebra [15. октобар 2006.]

[ZeeRex] Z39.50 Explain, Explained and Re-Engineered in XML http://explain.z3950.org/ [30. август 2007.]

[Zeremski01a] Зеремски, М., Сурла, *Д.*, Моделирање *UNIMARCформата у XML – Data* језику, Зборник радова са Симпозијума о рачунарским наукама и информационим технологијама YU INFO 2001 (на CD-ROM-у), Копаоник, 2001.

[Zeremski01b] Зеремски, М., Сурла, *Д.*, *Моделирање UNIMARC формата помоћу XML Schema језика,Зборник радова са* XXVIII Југословенског симпозијума о операционим истраживањима SYM-OP-IS 2001, Београд, 2001. стр. 261-264.

[Zeremski01c] Зеремски, М., Сурла, *Д.*, *Моделирање библиографских записа у облику XML докумената,* ИНФОТЕКА: Часопис за информатику и библиотекарство, Београд, 2(2001) 1-2, стр. 93-97.

[Zeremski01d] Zeremski, M., Surla, D., *Modeliranje bibliografskih zapisa pomoći XML Schema jezika,* Zbornik radova sa XXVIII Jugoslovenskog simpozijuma o operacionim istraživanjima SYM-OP-IS 2001, Beograd, 2001. str. 265-268.

[Zeremski02] М. Зеремски, *Моделирање UNIMARC формата у XML технологији.* Магистарска теза, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2002.

[Zeremski04] Zeremski, M., Surla, D., *Validation of XML bibliographic records in Java environment,* Novi Sad Journal of Mathematics, 2004. (in print).

[ZOOM] ZOOM: The Z39.50 Object-Orientation Model, http://zoom.z3950.org/

[ZSearcher] Zsearcher Information, http:// www.integralconcepts.com/zsearcher.html [10. јул 2006.]

CIP – Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

025.3:007] : 006.72

БОБЕРИЋ, *Данијела*
Preuzimanje bibliografskih zapisa po Z39.50 standardu / Danijela Boberić, Dušan Surla. - Novi Sad : Prirodno-matematički fakultet, Departman za matematiku i informatiku, 2007 (Novi Sad : ABM ekonomik). - 113 str. : graf.prikazi ; 24 cm

Tiraž 150. - Bibliografija. - Abstrakt.

ISBN 978-86-7031-134-3
1. Сурла, Душан
а) Библиотечки информациони системи - Комуникациони стандард Z39.50

COBIS.SR-ID 225878279

Мр Данијела Боберић је истраживач приправник Природно-математичког факултета у Новом Саду. Рођена је (1982) у Зрењанину. Дипломирала је (2005) на Природно-математичком факултету у Новом Сад, где је (2007) одбранила и магистарску тезу. Од марта 2007. године запослена је на Природно-математичком факултету у Новом Саду и ангажована је на пројекту Апстрактни модели и примена у рачунарским наукама (бр. 144017) који финансира Министарство науке и заштите животне средине Републике Србије. Има четири објављена научна рада који припадају области пројектовања библиотечких информационих система.

Др Душан Сурла је редовни професор Природно-математичког факултета у Новом Саду. Рођен је (1942) у Омсици, Република Хрватска. Дипломирао је (1969) на Филозофском факултету у Новом Саду. Магистрирао је (1976) на Машинском факултету у Новом Саду, а докторирао (1980) на новосадском Факултету техничких наука. На Природно-математичком факултету у Новом Саду ради од 1976. а за редовног професора изабран је (1991) за научну област информатика. Објавио је око 200 библиографских јединица из области роботике и области информациони системи.

ISBN 978-86-7031-134-3