Универзитет у Новом Саду

Природно – математички факултет

Департман за математику и информатику

Бојана Димић Сурла

Софтверски систем за каталогизацију по MARC 21 формату

- докторска дисертација -

Нови Сад, 2009.

Предговор

Предмет истраживања приказаних у овој докторској дисертацији припада области пројектовања информационих система. Описано је моделирање и имплементација софтверског система за каталогизацију по MARC 21 формату. За моделирање софтверског система коришћен је UML 2.0, а имплементација је извршена генерисањем софтверских компоненти на основу спецификације у Xtext окружењу. Архитектура система за каталогизацију базарина је на Eclipse plug-in технологији.

Дисертација садржи следећа поглавља:

1. Увод
2. Развој библиотечког софтверског система БИСИС
3. Опис софтверског окружења
4. Библиографски формати
5. Спецификација система за каталогизацију по MARC 21 формату
6. Имплементација система за каталогизацију по MARC 21 формату
7. RCP апликација система за каталогизацију
8. Закључак

У **првом** поглављу дефинисани су циљеви дисертације и укратко су описани стандарди у библиотекарству. Приказан је преглед постојећих резултата истраживања из предметне области и укратко описано изабрано софтверско окружење у којем је имплементиран систем за катакогизацију.

Истраживања у овој дисертацији добијена су у оквиру развоја библиотечког софтверског система БИСИС, који је укратко описан у **другом** поглављу. Детаљније је описана четврта верзија овог система а посебно едитор за каталогизацију који је развијен у оквиру ове верзије и припада предмету истраживања ове дисертације.

Коришћена методологија за развој софтвера и изабрано софтверско окружење за имплементацију система за каталогизацију приказани су у **трећем** поглављу. Описани су основне концепти одједињеног процеса, затим развој софтвера вођен моделима и развој заснован на софтверским компонентама. Описана је Eclipse платформа и EMF окружење. Описан је софтверски алат Xtext који се користи за развој доменски специфичних језика. Такође су описани основни концепти Eclipse plug-in технологије.

У **четвртом** поглављу описани су MARC библиографски формати. Дати су и модели формата и библиографских записа у облику XML шема и дијаграма класа.

У поглављу **пет** описана је примена обједињеног процеса на развој система за каталогизацију и дата је спецификација тог система. Спецификација је дата преко дијаграма случајева коришћења, дијаграма пакета, дијаграма класа и дијаграма секвенци.

У **шестом** поглављу описана је имплементација система за каталогизацију. Описана је граматика специфицирана у Xtext окружењу на основу које је генерисан основни едитор у облику Eclipse plug-in-ова. Такође је дата спецификација ограничења којима се проверава исправност записа у едитору, спецификација помоћи за унос података у едитору и темплејти за трансформацију записа. Описано је проширење основног едитора додатним функционалностима система за каталогизацију у Eclipse plug-in технологији, а то су унос локацијских података, експорт и импорт записа, приказ каталошких листића и библиотечко окружење.

У **седмом** поглављу је описано генерисање RCP апликација система за каталогизацију на основу креираних plug-in-ова система за каталогизацију укључујући и plug-in-ове стандардне Eclipse платформе. Укратко је описано и коришћење ове апликације.

На крају је дат **закључак** рада и наведени могући правци даљег истраживања.

У дисертацији је усвојен и коришен Harvard стил цитирања референци.

Захваљујем се ментору и члановима комисији која су својим сугестијама и конкретним предлозима допринели да структура дисертације буде прегледнија и да приказани резултати буду јасније истакнути.

Такође се захваљујем својој породици на разумевању и подршци.

Нови Сад, 2009. Бојана Димић Сурла

Садржај

Поглавље 1

Увод

Предмет истраживања у овој докторској дисертацији припада области информационих система, односно, пројектовању и имплементацији библиотечких информационих система. На званичној Интернет страници Конгресне библиотеке постоји посебан одељак који садржи списак и опште карактеристике библиотечких информационих система који се развијају широм света [1]. На овој страници описано је око сто библиотечких информационих система међу којима има комерцијалних, бесплатних и *open source* софтверских решења. Произвођачима библиотечких информационих система омогућено је да, путем посебне форме за упис на постојећу листу софтвера додају податке о сопственом производу. Софтверски системи укључени у поменуту листу подељени су на основу функционалности коју подржавају у три групе:

- сервиси за MARC записе [2] (*MARC Record Services*) у коју су укључени системи за дистрибуцију MARC 21 записа;
- MARC системи [3] (*MARC Systems*) која садржи алате за манипулацију са MARC 21 записима у коју спадају и разне врсте едитора за каталогизацију;
- специјализовани MARC алати [4] (*MARC Specialized Tools*) у коју спадају системи који подржавају неке додатне функције над записима, као што су разне врсте конверзија и валидација записа.

Већина софтверских система који припадају посматраној групи подржавају целокупну библиотечку функционалност за библиотеке различитих величина и потреба, како оних специјализованих на пример медицинских или академских библиотека тако и општих. Под библиотечком функционалношћу коју подржавају ови софтвери подразумевају се циркулација, каталогизација, претраживање и преузимање записа се удаљених локација, разне врсте извештавања, *on-line* јавно доступан каталог (*OPAC - Online Public Access Catalog*), управљање набавком и слично.

Одређени број софтверских система реализован је као *web* апликација, иако је већина доступна у виду *desktop* апликације, док је *OPAC* код већине апликација реализован као *web* апликација.

Карактеристика већине софтвера јесте да подржавају каталогизацију на разним језицима и скуп карактера подржан UNICODE стандардом.

Поред MARC 21 стандарда библиотечки софтверски системи представљени на Интернет адреси Конгресне библиотеке подржавају и друге библиотечке стандарде, као што су UKMARC, ISO 2709 и Z39.50 протокол и информатичке стандарде као што су XML и релационе базе података.

Развој софтвера из области библиотечког пословања је такође и предмет научних истраживања. Најчешћа тема истраживања из ове области односи се на имплементацију библиотечких стандарда у софтверским системима, комуникација и размена података између различитих библиотечких система, дигиталне библиотеке, као и анализа, дизајн и имплементација корисничког интерфејса библиотечких информационих система.

Циљ истраживања у овој докторској дисертацији је развој једног дела библиотечког информационог система који се односи на обраду библиографске грађе, односно каталогизацију. Каталогизација представља основни сегмент свих библиотечких информационих система. Она се односи на унос података о библиотечком фонду библиотеке и заснива се на библиографским стандардима за обраду библиографске грађе.

1.1 СТАНДАРДИ У БИБЛИОТЕКАРСТВУ

Библиотечки информациони системи су углавном базирани на стандардима у библиотекарству. Неки од стандарда који се користе у развоју библиотечких информационих система описани су у овом одељку.

UDC (**U**niversal **D**ecimal **C**lasification) [5] је међународни класификациони систем по коме се свакој публикацији, на основу области и подобласти које обрађује, додељује низ децималних бројева одвојених сепараторима тачака, при чему је за сваки број дефинисано његово значење у оквиру основних или проширених UDC таблица.

ISSN (**I**nternational **S**tandard **S**erial **N**umber) [6] је међународни стандардни број за серијске публикације.

ISBN (**I**nterantional **S**tandard **B**ook **N**umber) [7] је међународни стандардни број књиге који се додељује свакој књизи како би била јединствено идентификована.

ISO 10646 (UNICODE) [8] је стандард чији је циљ да јединственом кодном страном обухвати што већи број писама света.

ISBD стандард (**I**nternational **S**tandard **B**ibliographic **D**escription) [9] је међународни стандард за библиографски опис различитих типова публикација у машински читљивом облику, који се заснива на груписању података у области и на јединственом интерпункцијском систему.

MARC формат (**MA**shine **R**eadable **C**ataloguing **F**ormat) за библографски опис публикација настао је на основу **ISBD**-а, а много је погоднији од њега за аутоматизовану обраду библиографских података [10] . MARC је развијен пре више од 30 година у Конгресној библиотеци (*Library of Congress*) [11]. На основу овог формата, за потребе различитих држава и библиотечких система развили су се други библиографски формати, на пример: USMARC у Америци, CAN/MARC у Канади, UKMARC у Великој Британији. Године 1999. настао је формат MARC 21 [12] на основу формата USMARC и CAN/MARC.

UNIMARC (**UNI**versal **Ma**shine **R**eadable **C**ataloguing **F**ormat) [13] је међународни стандард за библиографски опис публикација у машински читљивом облику. Развијен је да би се превазишле разлике у примени **MARC**-а.

Z39.50 (амерички стандард ANSI/NISO Z39.50 и еквивалентан међународни стандард ISO 23950) [14] је стандард који дефинише како се врши проналажење потребних података у удаљеној бази података и преузимање пронађених података са удаљене базе, као и начин комуниције два рачунара који треба да остваре претходно наведено. Усаглашавањем овог стандарда са савременим Интернет технологијама настали су протоколи SRU (Search/Retrieve via URL) и SRW (Search/Retrieve Web service) [15].

Поред стандарда за размену библиографских података постоје и стандарди за размену података о корисницима, један од њих је Z39.83 [16].

1.2 ПРЕГЛЕД РЕЛЕВАНТНЕ ЛИТЕРАТУРЕ

У овом одељку дат је преглед научних истраживања која су повезана са темом докторске дисертације. Радови који су наведени у оквиру овог прегледа односе се на развој корисничког интерфејса система за каталогизацију, као и на примену стандарда за каталогизацију у развоју библиотечких информационих система. Поред тога, дат је и опис постојећих софтверких решења едитора за каталогизацију.

Едитор за каталогизацију представља један део радне станице каталогизатора која је била предмет бројних истраживања. Историјски развој радне станице каталогизатора био је предмет рада (Lange, 1993). У овом раду описан је развој процеса каталогизације од руком писаних каталошких листића, преко креирања каталошких листића на куцаћим машинама, прво ручним, а затим електронским, до појаве MARC записа и радних станица на PC рачунарима.

У раду (Leroy and Thomas, 2004) је описан утицај развоја Web-а на каталогизацију. Ту је наведено да је доступност великог броја информација путем Web-а као и могућност да више апликација буде отворено у различитим прозорима направило велики утицај на процес креирања библиографских записа. Наведени су и неки проблеми који настају услед коришћења бројних ресурса са Web-а, а то су појава великог броја грешака због копирања целог текста. Као последица тога јавља се потреба за аутоматском провером правописних грешака, као и да се записи приликом едитовања могу детаљно прегледати од стране каталогизатора.

Рад (Khurshid, 2001) се бави генералним принципима радне станице каталогизатора, њеном конфигурацијом, и утицајем радне станице на продуктивност каталогизатора. Као једно од значајних функционалности радне станице каталогизатора поред могућности брзог приступа удаљеним и локалним електронским ресурсима наводи се и пренос записа кроз мрежу као и софистицирано едитирање записа. Поред тога, наведено је да је у радним станицама које користе Windows могуће побољшати ефикасност каталогизације тако што се пронађу библиографски записи из других институција и коришћењем cut-and-paste функцију извуку се напомене о сажецима или неки други дужи делови записа и додају у сопствени каталог.

Значајан аспект радне станице каталогизатора представљају разни електронски алати за каталогизацију који нуде информације које су раније биле доступне само у папирној форми. Ови електронски алати за каталогизацију појављују се у микроформи, на CD-ROM-у, и на Web-у. Систематска анализа неких од ових алата дата је у раду (Khurshid, 2003), а у раду (Roper and Panell, 2001) анализиран је један алат за каталогизацију Cataloger's Desktop, производ Конгресне Библиотеке, и то прво издање које је изашло 2001. године. У оба рада је закључено да коришћење анализираних алата побољшава ефикасност рада каталогизатора.

У радовима (Belić and Surla, 2008a; Belić and Surla, 2008b) описано је моделирање и имплементација Web апликације за формирање електронског каталога библиографије истраживача, односно институција. Архитектура ове апликације је таква да се подаци могу уносити без познавања библиографског

стандарда, а приликом складиштења врши се мапирање унетих података на библиографски запис по усвојеном билиографском стандарду. За измену скупа библиографских података за унос потребно је само измени екранску форму за унос тих података док преостали део система остаје исти.

У оквиру BookMARC пројекта [17] извршене су значајна истраживања која се односе на спецификацију XML докумената који садрже податке о библиографским форматима. Развијене су шеме за UNIMARC и MARC 21 формат [18]. Резултат овог истраживања који се односи на UNIMARC форамат објављен је у раду (Carvalho, 2005), а целокупна шема UNIMARC формата дата је у [19]. Ова шема садржи податке о библиографском формату, као што су поновљивост поља и потпоља, обавезност, шифарници везани за потпоља и позиције карактера у заглављу записа. Поред тога што ова шема има велики број додатних елемената који се користе за креирање UNIMARC упутства, а то су детаљни описи елемената формата, упутства за каталогизацију, примери, напомене, па чак и подаци о табелама у оквиру којих ће се исцртавати елементи упутства. Због тога што обухвата све ове елементе шема је веома сложена и велика, као и XML документ који је њена инстанца.

Тема рада (Carvalho, et. al., 2004) је креирање XML шеме за MARC 21 формат, а у [20] је дата целокупна шема MARC 21 формата развијена у оквиру истог пројекта. Слично као шема за UNIMARC формат и ова шема је прилагођена креирању упутства али ова шема није тако потпуна као у случају шеме за UNIMARC јер недостају неки концепти који су дефинисани MARC 21 форматом, као што су заглавље записа и директоријум.

У наведеној литератури истакнуто је да је основна сврха истраживања била да се изврши формална репрезентација правила за каталогизацију који су прописани библиографским стандардима UNIMARC и MARC 21. Добијени резултати имају значајну примену у процесу валидације и краирању помоћи за каталогизацију.

Преглед постојећих едитора за каталогизацију. Овде је дат преглед неколико постојећих апликација за каталогизацију преузет из (Димић и Сурла, 2007). Неке од апликација приказаних у овом одељку доступне су путем Интернета, било као некомерцијални софтвери, било као демо верзије комерцијалних софтвера, док су подаци о другим апликацијама добављени путем упутства за употребу комерцијалних софтвара. Анализа функционалности доле навединих апликација представља полазну основу за истраживање у овој дисертацији.

MarcEdit

MarcEdit [21] је апликација написана у програмском језику *Perl* за оперативни систем *Windows,* њен аутор је Terry Reese са универзитета у Орегону (*Oregon State University*).

MarcEdit је апликација која у потпуности подржава функционалност за рад са библиогарфским записима. Састоји се од неколико делова, одвојених софтверских пакета, као што су Z39.50 клијент, апликације за конверзију података између различитих начина репрезентације записа и едитор за унос података о публикацијама.

Податке је у запис могуће додавати или модификовати директним уносом у текстуално поље у коме је приказан запис у MARC 21 формату. На овај начин омогућена је врло једноставна модификација, са једним недостатаком који се односи на одсуство било какве контроле над унетим подацима. Други начин уноса података је коришћење неких од функција понуђених у менију *Tools*.

Помоћни прозор за унос потпоља садржи падајући мени који омогућава селекцију поља, затим се у суседно тектуално поље уноси ознака потпоља, а затим у одговарајуће текстуално поље и текст. Оваквим уносом може доћи до грешке која се односи на невалидност записа. Узрок грешке могао би бити избор поља из падајућег менија који форматом није дефинисан, избор контролног поља (која не могу садржати потпоља) или уносом ознаке потпоља које није дефинисано за изабрано поље. За шифрирана потпоља не постоји начин селекције шифре из шифарника.

Polaris

Polaris је комерцијални софтверски производ америчке фирме *GIS Information Systems* [22], који у потпуности подржава каталогизацију по MARC21 стандарду.

Поред стандардних пречица за рад са текстом, основни прозор едитора садржи основне податке о запису, као што су контролни број, назив софтвера у ком је запис креиран, статус записа и наслов публикације на коју се запис односи. Највећи део основног прозора резервисан је за сам MARC 21 запис.

Селекцијом заглавља записа или контролног поља отвара се помоћни прозор у коме су понуђене шифре дозвољене за одговарајућу позицију карактера. Оваквим начином уноса вредности смањена је могућност грешке која се односи на унос погрешне шифре. Међутим, шифарници су обезбеђени само за заглавље записа и контролна поља, док код шифрираних потпоља не постоји овакав тип контроле.

IsisMarc

IsisMarc [23] је у основи интерфејс за унос података у *CDS/ISIS* базе података и ради под оперативним системом *Windows*. Апликација представља унапређено окружење које је замена за стандардну *Winisis* екранску форму за унос података.

IsisMarc је *open source* производ *UNESCO*-овог сектора за комуникације и информације (*Communication and information Sector*) у Француској, а настао је уз сарадњу ове институције са Конгресном библиотеком у Америци и SUI програма аргентинског министарства за образовање (*SIU Program of the Ministry of Education of Argentina*).

IsisMarc едитор омогућава кретање кроз колекцију записа и то или секвенцијалним кретањем кроз скуп записа или непосредном селекцијом записа према редном броју. Селекцијом одговарајућег записа омогућена је његова модификација, такође могуће је креирање новог записа који се додаје текућој колекцији.

Делови заглавља записа који су најрелевантнији за обраду записа издвојени су у горњем делу едитора, где је путем падајућег менија могуће изабрати код за одговарајућу позицију карактера у заглављу записа. Предност поменутог приступа састоји се у једноставном прегледу и измени информација битних за обраду записа, као што су статус записа, библиографски ниво, облик каталогизације и др. Могуће побољшање могло би се постићи навођењем текстуалног описа информације која се смешта у заглавље записа.

Испод дела за унос заглавља записа налазе се два падајућа менија за унос вредности индикатора. Ова два менија постају активна када се селектује поље за које је дефинисан индикатор и садрже могуће вредности индикатора за селектовано поље.

Свако поље у запису представљено је својом ознаком, описом информације која се чува у пољу и садржајем поља. Испред поља и потпоља која имају особину поновљивости налази се знак '%' чијом селекцијом је могуће додати ново поновљено поље, чиме је уведена контрола поновљивости. Селекцијом знака '±' испред поља које садржи потпоља отвара се листа потпоља дефинисана за селектовано поље. Знаком '?' испред поља представљена је пречица до спецификације поља у *html*-у. Ова спецификација одговара спецификацији са званичне Интернет адресе Конгресне библиотеке, али се датотеке које чине ову спецификацију јављају као саставни део инсталације софтвера. Оваквим приступом задовољена је особина једноставног увида у податке које се односе на сам формат.

Вредност контролних поља која садрже тачно дефинисане позиције карактера уносе се у посебном прозору. У горњем делу тог прозора приказан је текући садржај поља подељен на позиције карактера, док централни део прозора заузима текстуални опис делова поља са могућношћу модификације вредности. У случају селекције позиције карактера која је шифрирана или уноса вредности шифрираног потпоља отвара се одговарајући шифарник из ког је могуће преузети код.

Двокликом миша на део за унос вредности нешифрираног потпоља отвара се проширено текстуално поље за унос. Оваквим решењем задовољена је особина едитора која се односи на могућност ефикасног уноса односно модификације унетих података. Уносом вредности у текстуално поље аутоматски се ажурира и одговарајућа вредност у линији која се односи на поље чије потпоље се модификује.

Concourse

Concourse [24] је библиотечки софтверски систем који је комерцијални производ фирме *Book Systems, Inc* [25] из САД-а. Као саставни део овог система понуђен је професионални едитор за унос библиографских података.

Модел едитора који је развијен у оквиру система *Concourse* уводи специфичан приступ каталогитације по MARC 21. Специфичност се огледа у томе што се каталогизација може вршити на два начина, један начин не подразумева познавање стандарда, док се на други начин каталогизација врши у специјалном едитору, *Concourse Pro* за чије коришћење је потребно познавање стандарда.

Подаци о публикацији подељени су на логичке целине, на основу којих је унос подељен у неколико табова. У првом табу уносе се основни подаци о публикацији. Други таб носи назив *Analitics* и овде се уносе подаци који се односе на аналитику, као што су предметне одреднице, наслов серијске публикације, напомене и кратак садржај. Трећи таб, *Other* омогућава унос физичких карактеристика, као и остале наслове публикације.

Таб *Media* омогућава везивање мултимедијалних линкова за запис. Такође, овде је могуће унети URL адресу која се на било који начин односи на публикацију која се обрађује.

Таб који носи назив *MARC* пружа кориснику могућност прегледа записа у MARC 21 формату који је аутоматски генерисан на основу података унетих у прва четири таба. Овде није дозвољена модификација података.

Евентуално побољшање описаног решења састојало би се у томе да се уведе могућност дефинисања тагова од стране корисника са скупом поља за унос,

због тога што се може јавити потреба за различитим груписањем података. Такође, одсуство података о формату могло би се сврстати у недостатке описаног решења. При креирању едитора овог типа, као и приликом генерисања нових табова са пољима за унос посебна пажња мора се посветити пресликавању између поља за унос и елемената записа.

Описани едитор има могућност аутоматске провере правописа (*spell check*) и валидацију унетих података. Такође, уколико се обрађују записи који имају неке податке исте, могуће је једноставним пречицама са тастатуре копирати вредности претходно обрађеног записа.

БИСИС

Библиотечки софтверског система БИСИС развија се од 1993. године. Прва верзија овог система завршена је 1996. године. Те године је ова верија система уведена у неколико факултетских библиотека у Републици Србији. Од 2005. године у току је континуирана реализација пројекта *Библиотечка мрежа матичних градских библиотека Војводине*, која је базирана на систему БИСИС вер. 3. Основна идеја која је и реализована успостављањем библиотечке мреже Војводине у оквиру овога пројекта јесте да су библиотеке у мрежи повезане комукационо и софтверски тако да путем Интернета међусобно размењују библиографске записе.

У складу са досадашњим искуством и захтевима факултетских, градских и специјализованих библиотека, урађена је нова (четврта) верзија система БИСИС и у току 2008. године све библиотеке библиотечке мреже Војводине прешле су на четврту верзију софтвера. Софтверска архитектура ове верзије омогућује прилагођавање система специфичним потребама сваке библиотеке што омогућује да се задовоље сви захтеви електронског пословања библиотеке.

У четвртој верији система значајно је унапређен кориснички интерфејс како за обраду библиографске грађе тако и за позајмицу библиотечке грађе. Такође је додат већи број извештаја о библиографским записима и о коришћењу библиотечке грађе.

Истраживање у овој дисертацији односи се на унапређење корисничког интерфејса за обраду библиографске грађе у верзији 4 и зато је он детаљније описан у наредном поглављу.

1.3 СОФТВЕРСКО ОКРУЖЕЊЕ

За развој софтверског система за каталогизацију по MARC 21 формату могу се користити различита софтверска окружења, а једно од њих је Eclipse

платформа [26], која је коришћена у овој дисертацији. Eclipse је изабран као софтверско окружење у ком се врши спецификација система за каталогизацију, као и софтверско окружење у ком се специфицирани систем за каталогизацију извршава.

Eclipse је *open source* пројекат који представља развојно окружење за израду софтвера. Овај алат обезбеђује рад са различитим програмским језицима и на различитим платформама. *Eclipse* обезбеђује *plug-in* оквир за интеграцију додатних софтверских алата.

Eclipse платформа је заснована на програмском језику Јава. Програмски језик *Java* [27] је *open source* производ фирме *Sun Microsystems*. То је објектно оријентисани програмски језик, специјално дизајниран да што мање зависи од специфичних карактеристика рачунарског система. *Java* омогућава да се једном написан и преведен програм може извршавати на различитим платформама које је подржавају.

Целокупну архитектуру Eclipse-а чине plug-in-ови плус Platform Runtime, мали програм који представља језгро платформе. Језгро Eclipse-а је окружено стотинама и хиљадама plug-in-ова. Plug-in није ништа друго него Јава програм који на неки начин проширује функционалност Eclipse-а. Сваки Eclipse plug-in може или да користи сервисе других plig-in-ова или да прошири њихову функционалност да би се као такав користио од стране других plug-in-ова (Clayberg and Rubel, 2008).

Eclipse је отворена платформа дизајнирана тако да се може једноставно проширити (*extend*) додатним функционалностима. У језгру ове платформе налази се Eclipse SDK (Software Development Kit) око кога се могу креирати различити софтверски производи и алати, који се даље могу проширивати у оквиру Eclipse-а. Архитектура Eclipse-а је нарочито погодна за проширивање, а проширивост се постиже коришћењем plug-in технологије [28].

Кориснички интерфејс у оквиру Eclipse-а развија се коришћењем SWT – Standard Widget Toolkit технологије [29]. Као додатак SWT технологији, за развој екранских форми Eclipse-а користи се и JFace [30]. JFace се заснива на SWT-у али нуди неке сложеније концепте за развој корисничког интерфејса. Опис развоја корисничког интерфејса коришћењем технологија SWT и JFace дат је у (Harris and Warner, 2004).

На основу скупа Eclipse plug-in-ова могу се генерисати независне софтверске компоненте коришћењем технологије Rich Client Platform - RCP [31]. RCP апликација представља подскуп plug-in-ова Eclipse-а, а сам израз Rich Client

Platform односи се на минималан скуп plug-in-ова неопходних да се направи једна RCP апликација (McAffer and Lemieux, 2005).

У оквиру Eclipse платформе развијено је окружење за моделирање софтверских система EMF - Eclipse Modeling Framework [32]. EMF је софтверско окружење које омогућава развој софтвера генерисањем програмског кода на основу структуираног модела података (Budinsky et al., 2003). EMF модел може се генерисати коришћењем Eclipse-ових софтверских алата на основу UML модела, XML шеме, на основу Јава програмског кода (Powell, 2004).

Поред тога, EMF представља подршку концепту развоја софтвера вођеним моделима и представља основу за развој других пројеката из ове области, као што је и OpenArchitectureWare (oAW) [33]. Опис коришћења софтверског алата oAW дат је у (Efftinge et al., 2008). Један део oAW пројекта је софтверски алат Xtext који се користи за креирање доменски специфичних језика. Креирање доменски специфичних језика са Xtext-ом описано је у (Kolb and Voalter, 2008).

У оквиру софтеврског алата Xtext парсер за обраду текста имплементиран је коришћењем ANTLR – Another Tool For Language Recognition [34].

1.3.1 Примена Eclipse plug-in и RCP технологија

Eclipse платформа налази велику примену као окружење за развој софтвера у различитим програмским језицима. Она омогућава интеграцију различитих софтверских алата развијених у облику plug-in-ова. Међутим, Eclipse окружење се у многим истраживањима користи и као окружење за крајњег корисника. У овим истраживањима развијају се Eclipse plug-in-ови који интегрисани у Eclipse платформу нуде готов софтверски производ који се користи од стране крајњег корисника. Овако развијени Eclipse plug-in-ови могу се издвојити у независну апликацију коришћењем RCP технологије. Описани концепт развоја софтвера представља и основу за развој система за каталогизацију који је тема ове дисертације. У овом одељку дат је преглед истраживања која се односе на развој Eclipse plug-in-ова и RCP апликација за крајњег корисника.

У раду (Spjuth, et al., 2007) описана је архитектура и основне карактеристике софтверског алата *Bioclipse* који се користи за управљање хемијским и биолошким подацима, као што су молекули, протеини, и слично. *Bioclipse* је написан у Јави и заснива се на Eclipse Rich Client Platform технологији и plug-in софтверској архитектури. Оваква архитектура омогућава да се апликација може једноставно проширити од стране других система било којим потребним

функцијама. *Bioclipse* обезбеђује интуитиван кориснички интерфејс, едитирање у две димензије (*2D-editing*), приказивање у три димензије (*3D-visualizing*), конверзију између различитих формата фајлова, израчунавање хемијских карактеристика, и још додатних функционалности и све то интегрисано у кориснички пријатељску (*user-friendly*) десктоп апликацију. Поред тога, едитирање у *Bioclipse*-у подржава и стандардне функције за копирање (*cut* и *paste*), за превлачење (*drug and drop*), и за враћање потеза (*undo/redo*).

У раду (Watson and DeBardeleben, 2006) описан је Eclipse plug-in *Parallel Tools Platform (PTP)* за покретање, контролисање и праћење паралелног извршавања програма на више рачунара који се користи у научним истраживањима. Први део овог рада резервисан је за опис Eclipse архитектуре и опис компоненти корисничког интерфејса од којих се састоји Eclipse. У другом делу рада наведене су основне карактеристике корисничког интерфејса и архитектуре *PTP* plug-in-а. Кориснички интерфејс *PTP*-а развијен је коришћењем стандардних Eclipse компоненти корисничког интерфејса тако да подржава приказ стања процеса који се паралелно извршавају на различитим рачунарима. Поред приказа, обезбеђено је управљање и надгледање паралелних процеса преко интуитивног корисничког интерфејса. Архитектура *PTP* апликације дизајнирана је тако да подржи различите врсте паралелних система а коришћењем Eclipse plug-in технологије омогућена је проширивост овог система за нека будућа достигнућа у паралелном извршавању апликација.

Eclipse plug-in технологија и RCP налазе примену и у развоју апликација за финансијско пословање, праћење и анализу кретања вредности акција на берзи у оквиру великих светских компанија. У (Taft, 2006) је наведено да RCP технологија нуди управо оно што је потребно једној компанији, а то је окружење које омогућава поновну употребу (*reuse*) захваљујући plug-in архитектури као и смањење трошкова развоја и одржавања софтвера. Поред тога, истиче се значај могућности ажурирања Eclipse апликација (*Eclipse technology's update capabilities*) као и одличне графичке перформансе RCP корисничког интерфејса.

Један од постојећих едитора који су развијени као plug-in за Eclipse је TeXLipse [35] који омогућава едитирање LaTeX докумената. У (Ojala, 2005) дата је техничка документација пројекта TeXLipse. У њој се наводи да је он развијен као Eclipse едитор који подржава бојење синтаксе, комплетирање уноса, пријављивање грешака у уносу, кориснички дефинисане темплејте, структуирани приказ документа и слично. У позадини овог едитора налази се модел LaTeX документа и парсер заснован на EBNF граматици.

1.4 ПРЕДМЕТ И ЦИЉ ИСТРАЖИВАЊА

Предмет истраживања је развој софтверског система за каталогизацију. Под софтверским системом за каталогизацију подразумева се апликација која омогућава кориснику односно библиотекару да уноси библиографске податке за одговарајућу публикацију онако како то прописује изабрани библиографски стандард. Неке од основних функционалности које би овакав један едитор требало да обезбеди су следеће: ефикасан унос библиографских података; контрола унетих података; увид у правила за каталогизацију која су прописана библиографским стандардом; експорт и импорт библиографских записа; ефикасно коришћење нормираних података; приказ записа у форми лисића.

Главни проблеми који се јављају у области каталогизације су:

- формирање библиографских записа који задовољавају ограничења библиографских формата;
- верификација исправности и квалитета библиографских записа;
- израда корисничког интерфејса за каталогизацију базираног на савременим компонентама за Windows окружење.

Постојећи едитори за каталогизацију који су описани у одељку 1.2 реализују два приступа за унос података:

- унос библиографских података у класичне екранске форма са текстуалним пољима и лабелама, у које се подаци уносе без познавања стандарда и
- унос библиографског записа у форми слободног текста при чему структура тог текста није дефинисана и нису подржане одговарајуће контроле уноса.

У овој дисертацији усвојен је приступ уноса библиографског записа у форми слободног текста. За разлику од свих пронађених едитора за каталогизацију који користе овај приступ, у овој дисертацији идеја је да се реализује едитор у ком је структура текста дефинисана MARC 21 форматом и подржана је контрола садржаја записа по овом формату у реалном времену.

Циљ истраживања је развој софтверског система за обраду библиографске грађе у облику едитора за унос структуираног текста, по угледу на савремене едиторе за писање програмског кода. Реализација овог циља састоји су у следећем:

- формална спецификација модела библиографских записа;
- формална спецификација ограничења над моделом библиографских записа;

- генерисање софтверске компоненте едитора за каталогизацију на основу формалне спецификације функционалности едитора.

Полазне хипотезе у истраживању су биле да је могуће:

- специфицирати модел библиографских записа у облику језика специфичног за домен;
- да се сва ограничења над моделом библиографских записа могу формално описати језиком за опис ограничења над објектима (Object Constraint Language - OCL);
- да се на основу формалне спецификације може генерисати програмски код едитора за обраду библиографске грађе.

Резултати истраживања у овој дисертацији су потврдили наведене полазне хипотезе и основни добијени научни резултати су следећи:

- Дат је предлог XML шеме библиографских формата MARC 21 и UNIMARC и објектног модела креираног на основу ове шеме (одељак 4.3).
- На основу постојеће XML шеме MARC 21 записа креиран је објектни модел MARC 21 записа (одељак 4.4).
- У софтверском алату Xtext специфицирана је граматика за опис модела MARC 21 формата. Дат је комплетан листинг ове спецификације, као и опис појединачних концепата граматике (одељак 5.2).
- На основу специфициране Xtext граматике генерисан је EMF модел за MARC 21 библиографски запис (одељак 5.4).
- Над генерисаним EMF моделом библиографског записа спецификацирана су ограничења којима се провера исправност и квалитет MARC 21 записа. Спецификација ограничања извршена је коришћењем језика Check (одељак 5.6).
- На основу Xtext граматике генерисан је основни едитор за каталогизацију по MARC 21 формату (одељак 5.5.).
- Основни едитор проширен је додатним функционалностима система за каталогизацију, а то су унос локацијских података, експорт и импорт записа, приказ каталошких листића и библиотечко окружење. Ово проширење имплементирано је у Eclipse plug-in технологији. Моделирање и имплементација ових додатних функционалности система за каталогизацију дати су у шестом поглављу.
- На основу креираних plug-in-ова којима је имплементиран систем за каталогизацију, коришћењем RCP технологије генерисана је

апликација која се може користити независно од Eclispe-а што је описано у седмом поглављу.

Поглавље 2

Развој библиотечког софтверског система БИСИС

Резултати истраживања приказани у овој дисертацији добијени су у оквиру развоја библиотечког софтверског система БИСИС. Овај систем се развија од 1993. године и тренутно је актуелна четврта верзија система, која је базирана на XML технологијама.

2.1 БИСИС ВЕРЗИЈА 1

Ова верзија система је настала као резултат пројекта *Систем за формирање и претраживање информација*. Овај пројекат је периоду од 1993. до 1996. године финансирало Министарство за науку и технологију Републике Србије. Резултати истраживања на овом пројекту приказани су у монографији (Лазаревић и др., 1996).

Софтверски систем БИСИС вер. 1.0 реализован је у програмском језику C, а као систем за управљање базом података коришћена је db_VISTA. За индексирање и претраживање библиографских записа, коришћен је текст сервер, који је у оквиру наведеног пројекта реализован у Институту „Михајло Пупин“ у Београду. За обраду библиографске грађе усвојен је UNIMARC формат. У овој верзији дефинисане су следеће базе података: База UNIMARC формата, База радног окружења библиотекара, База библиографске грађе и База циркулације библиотечких докумената. Овако формиране базе података су се уз потребне измене користиле и у наредним верзијама система.

У овој верзији система подржане су следеће функције: Дефинисање стандарда за одржавање библиографске грађе и структуирање записа, Дефинисање радног окружења библиотекара, Формирање библиографске грађе, Библиотекарско извештавање и документовање и Корисничко претраживање.

Спецификација информационих захтева ове верзије урађена је по методологији описаној у 4. поглављу монографије (Лазаревић и др., 1996), која се базира на Структурној системској анализи и Проширеном моделу објекти- везе. Та методологија примењивана је уз коришћење CASE алата ARTIST који је развијен на Факултету организационих наука у Београду.

2.2 БИСИС ВЕРЗИЈА 2 И ВЕРЗИЈА 3

Софтверски систем БИСИС вер. 2.0 реализован је у Oracle окружењу и програмском језику Java. Од Oracle-ових производа користи се Oracle Server верзија 8.0.3 и текст сервер ConText Cartridge верзија 2.3.6, која подржава рад са текстом у Unicode-у. Систем је предвиђен да ради у Интернет, односно Интранет окружењу. Кориснички интерфејс је имплементиран као Java аплет. Овом верзијом су подржане све функције из претходне верзије система.

Основна разлика између верзија 2.0 и 3.0 је текст сервер за индексирање и претраживање библиографских записа. 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. Овом верзијом су такође подржане све функције из претходне верзије система.

Моделирање система извршено је коришћењем обједињеног језика моделирања (Unified Modeling Language – UML). Имплементација система реализована је као Интернет апликација у Java окружењу.

2.3 XML ТЕХНОЛОГИЈЕ У БИБЛИОТЕКАРСТВУ

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 библиографских 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 (Zeremski and Surla, 2003; Зеремски и Сурла, 2001а; Зеремски и Сурла, 2001б) показано је да се помоћу XML шема језика могу моделирати сви концепти UNIMARC формата, као и сви елементи тих концепата. 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 (Зеремски и Сурла, 2001ц; Зеремски и Сурла, 2002). Овако формирана XML шема могла би се користити за валидацију библиографских записа формираних по UNIMARC формату као и за контролу квалтета библиографских записа.

Међутим, у радовима (Milosavljević and Dimić, 2007; Dimić et al., 2010) дат је предлог XML шема које описују концепте библиографских формата (MARC21, UNIMARC и YUMARC који представља варијанту UNIMARC формата) као што су поља, потпоља, индикатори, шифарници, док појаве тих концепата садрже конкретне податке о формату, односно спецификацију појединачних поља, потпоља, индикатора и друго. Поред тога, у овим радовима описане су и XML шеме библиографских записа формираних по тим форматима. Овакве шеме омогућавају да се у софтверским системима ради само са изабраним деловима формата за обраду библиографске грађе.

Контрола квалитета XML библиографских записа. Монографија (Будимир и Сурла, 2004) посвећена је контроли квалитета библиографских записа. Дефинисана је и систематизована контрола за утврђивање квалитета библио-графских 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 (Шкрбић и Сурла, 2004; Шкрбић и Сурла, 2005; Škrbić and Surla, 2008) 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 раду (Vidaković and Racković, 2006) дат је предлог формирања каталошких листића применом XML технологије. У радовима (Rađenović et al., 2006;

Рађеновић и др, 2006; Rađenović et al., 2009) 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.

Имплементацију XML едитора за библиографске формате и библиграфске записе. Детаљан опис функционалности едитора за обраду библиографске грађе у трећој верзији система БИСИС дато је у (Сурла и др., 2003). 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 (Мијић, 2003) дат је предлог моделирања и имплементације XML едитора за опис UNIMARC формата. У монографији (Шкрбић и Сурла, 2005) приказано је моделирање и имплементација XML едитора који илуструје начин употребе XML native технологија у развоју библиотечког информационог система заснованог на UNIMARC стандарду.

2.4 БИСИС ВЕРЗИЈА 4

Верзија 4 система БИСИС је резултат истраживања у оквиру следећих магистарских теза (Рађеновић, 2006; Боберић, 2007; Тешендић 2007; Димић 2007). У монографији (Тешендић и Сурла, 2006) и радовима (Tešendić et al., 2009; Milosavljević and Tešendić, 2010) описан је део система који се односи на коришћење библиотечке грађе, односно циркулацију. У монографији (Боберић и Сурла, 2007) и раду (Boberić and Surla, 2009) и описан је едитор за креирање упита по Z39.50 протоколу.

У четвртој верзији система БИСИС креиран је текст сервер за индексирање и претраживање библиографских записа коришћењем библиотеке *Lucene* [36]. Опис овог текст сервера за индексирање и претраживање библиографских записа дат је у раду (Milosavljević et al, 2010).

Један од важних сегмената у развоју библиотечких система је креирање разних врста каталошких листића на основу библиографских записа и овај аспект је значајно унапређен у четвртој верзији система БИСИС. У монографији (Рађеновић и др, 2006) и раду (Rađenović et al, 2009) предложено је решење за креирање каталошких листића у облику софтверског пакета *Report.* У оквиру овог пакета генерисање каталошких листића извршено је формирањем темплејта у алату FreeMarker [37]. Формирањем темплејта на основу којих се врши генерисање каталошких листића избегнуто је преправљање апликације приликом проширивања, јер се додавање нових листића и концепата врши додавањем нових темплејта односно изменом

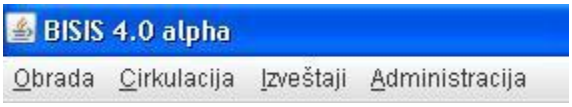
постојећих темплејта. На овај начим могуће је ажурирати каталошке листиће без поновног компајлирања изворног кода. Такође, предложено решење омогућава динамичко учитавање имена каталошких листића, које се своди на учитавање имена темплејта. На овај начин омогућено је да пакет *Report* буде независан и доступан свим сегментима система БИСИС.

Четврта верзија система БИСИС уводи концепт дефинисања улога библиотекара који користе овај систем. Библиотекари могу имати улоге за каталогизацију, циркулацију и администрацију. На основу дефинисаних улога врши се кастомизација система БИСИС што је описано у раду (Dimić and Milosavljević, 2009). У овом раду описани су појмови улога библиотекара, са акцентом на улози за каталогизацију у којој се дефинишу типови обраде који су улазна информација за едитор за каталогизацију.

Развој едитора за каталогизацију у оквиру четврте верзије система БИСИС описан је у монографији (Димић и Сурла, 2007) и радовима (Dimić and Surla, 2009; Dimić et al., 2010). Систем за каталогизацију који је описан у овој дисертацији представља унапређење едитора за каталогизацију који је развијен у оквиру четврте верзије система БИСИС и зато је у наставку дат опис основних карактеристика тог едитора преузет из (Димић и Сурла, 2007).

2.5 ЕДИТОР ЗА КАТАЛОГИЗАЦИЈУ У СИСТЕМУ БИСИС ВЕРЗИЈА 4

На слици 2.1 приказан је главни мени апликације БИСИС верзија 4. Мени *Obrada* садржи акције које се односе на обраду библиографске грађе односно каталогизацију. Мени *Cirkulacija* садржи акције за коришћење библиотечке грађе у раду са члановима библиотеке. Мени *Izveštaju* садржи акције за креирање различитих врста извештаја и статистика на основу библиотечког фонда. Мени *Administracija* садржи акције за администрацију система у коју спадају додела улога библиотекарима, дефинисање типова обраде, администрација шифарника који се користе у циркулацији и каталогизацији и слично.



Слика 2.1 Главни мени апликације

На слици 2.2 приказане су акције менија *Obrada*. Акцијом *Pretraživanje* отвара се прозор за претраживање библиографске грађе, а акцијом *Novi zapis*... отвара се едитор за обраду библиографске грађе за унос новог записа.

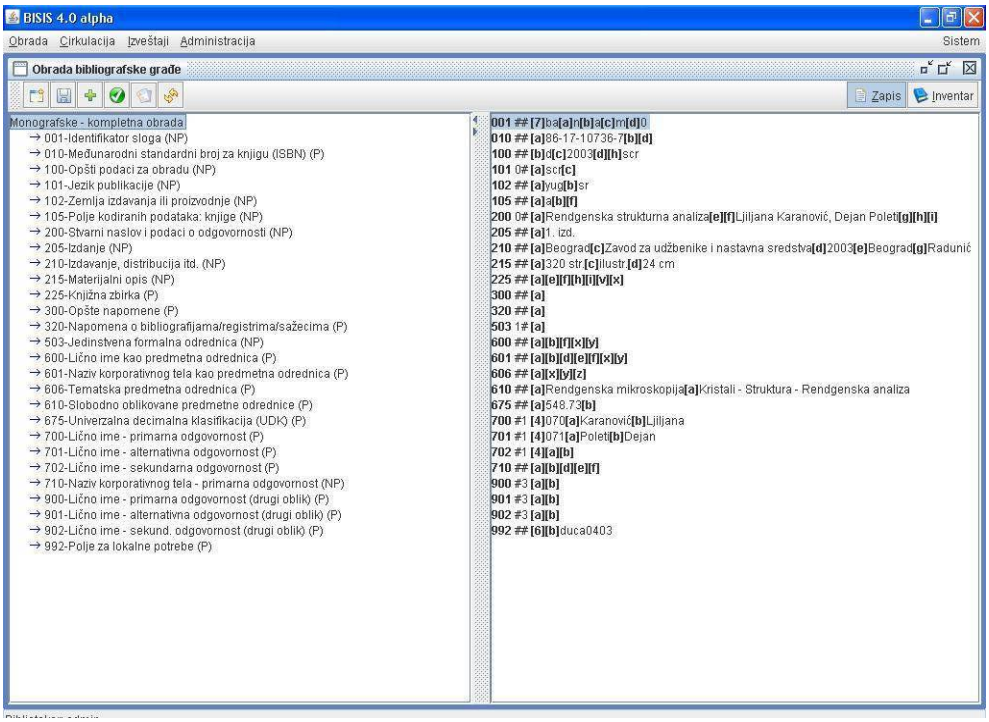
Слика 2.2 Акције менија *Obrada*

Претраживање библиографске грађе детаљно је описано у монографији (Димић и Сурла, 2007). После претраживања може се изабрати један од записа који се налазе у скупу резултата претраживања и отворити у едитору за каталогизацију за измену или унос новог записа на основу постојећег. Поред тога, едитор за обраду библиографске грађе у систему БИСИС могуће је отворити позивом акције *Novi zapis*... из менија *Obrada* (слика 2.2).

Приликом покретања едитора као улазни параметар користи се окружење библиотекара који је пријављен на систем. Сваки каталогизатор има дефинисане типове обраде од којих је један означен као подразумевани (*default*) тип обраде и он ће се користити уколико библиотекар не изабере неки други тип обраде.

Један тип обраде везан је за један тип публикације (серијска, монографска, аналитика,...) и састоји се од скупа поља и потпоља. За сваки тип публикације стандардом је дефинисан скуп поља и потпоља, а тип обраде је подскуп тог скупа. Типом обраде дефинише се и скуп обавезних потпоља као и подразумеване вредности за потпоља и индикаторе. Екранска форма едитора за каталогизацију отвара се за одређени тип обраде, при чему се приказују само она поља и потпоља која су дефинисана тим типом обраде. Приликом покретања едитора аутоматски се додељују подразумеване вредности дефинисане типом обраде, а приликом снимања врши се контрола да ли су унете вредности за потпоља која су типом обраде дефинисана као обавезна.

На слици 2.3 приказана је екранска форма едитора за каталогизацију који је развијен у оквиру четврте верзије система БИСИС. Овај едитор се састоји из два дела. На левој страни екранске форме приказан је део стабла YUMARC формата, а на десној страни стабло библиографског записа који се обрађује у едитору. Коренски елемент стабла формата садржи назив и опис библиографског формата. Поделементи коренског елемента су поља дефинисана форматом. Поља су представљена називом и описом, а податак у загради означава да је поље поновљиво (*R*) или није поновљиво (*NR*). Слично су представљена и потпоља.



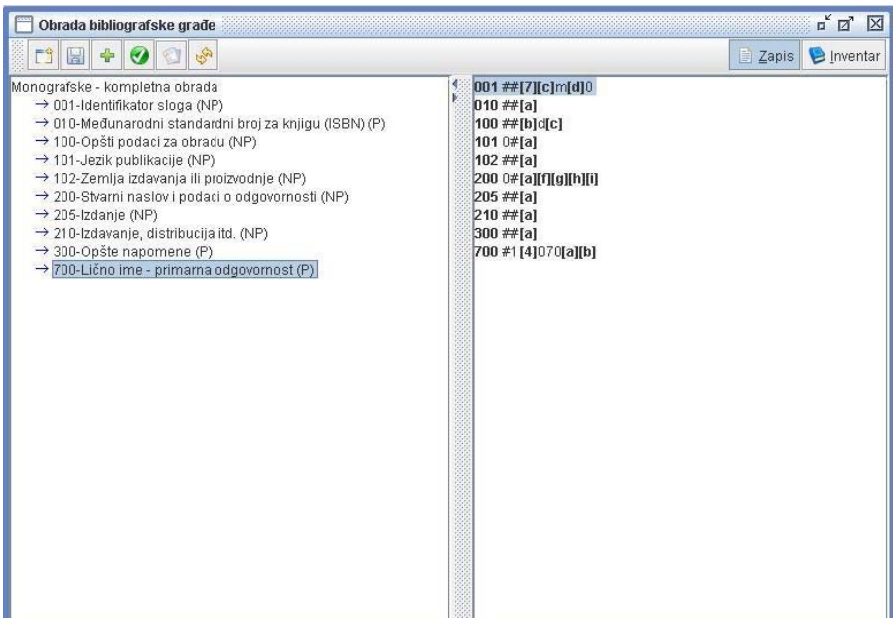
Слика 2.3 Екранска форма едитора за каталогизацију у четвртој верзији система БИСИС

Унос података у едитору врши се директно у стаблу библиографског записа, као што је приказано на десној страни на слици 2.3. За сваки елемент записа отвара се текстуални едитор за унос података. За елементе формата који имају предефинисани скуп вредности, као што су индикатори, нека потпоља и позиције карактера отвара се едитор са додатним дугметом за отварање посебног прозора из кога се селектује шифра. Приликом уноса појединочног елемента врши се контрола унетог податка.

Сваки елемент у запису представљен је хијерархијски у стаблу формата (лева страна екранске форме на слици 2.3) својом спецификицијом. Приказивањем података о формату на екранској форми едитора каталогизатору је омогућен увид у податке о формату који му помажу приликом креирања записа. Овакав хијерархијски начин представљања података о формату је прегледан и омогућава каталогизатору да једноставно добије основне податке о формату и структури библиографског записа који креира. Из стабла формата он може прочитати где у запису треба да унесе одређени библиографски податак као и која су ограничења над елементима записа у шта спадају поновљивост елемента и спецификација елемента за одређени тип публикације.

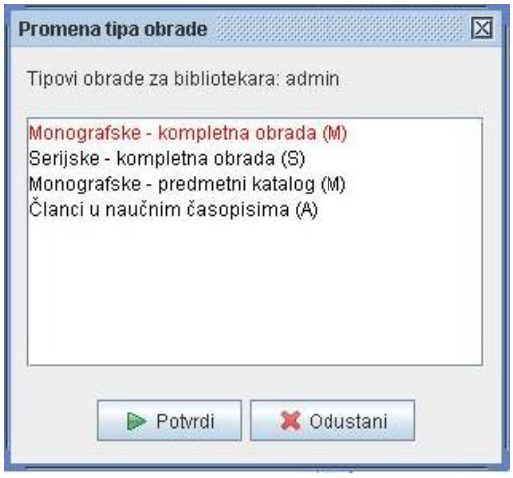
После претраживања библиотечког фонда један од записа који су добијени као резултат претраживања може се отворити у едитору. Пре отварања записа у едитору проверава се да ли пријављени библиотекар може да обрађује тип публикације који одговара типу публикације изабраног записа. Уколико нема, то значи да библиотекар није овлашћен за обраду типа публикације на коју се запис односи и пријављује се одговарајућа порука. Ако постоји одговарајући тип обраде, отвара се едитор при чему се сада стабла библиографског формата и библиографског записа формирају на основу тог типа обраде и структуре самог записа који се отвара. Односно, обе структуре ће садржати поља и потпоља из записа са додатком потпоља која су обухваћена типом обраде. На слици 2.3 приказан је изглед едитора приликом учитавања једног записа из листе резултата претраге.

Приликом отварања новог записа у едитору у стаблу формата (лева страна екранске форме) и стаблу записа (десна страна екранске форме) налази се одговарајући скуп поља и потпоља (слика 2.4). Који ће се скуп поља и потпоља појавити зависи од подразумеваног типа обраде за пријављеног библиотекара. Сва потпоља која се налазе у скупу потпоља за тај тип обраде налазиће се заједно са својим припадајућим пољима у стаблу формата и стаблу записа. Поред тога, одређена потпоља у запису садрже вредности које су по типу обраде за њих дефинисане као подразумеване (*default*). На пример, потпоље *c* поља *001* које представља библиографски ниво публикације као подразумевану вредност има шифру *m* која значи да се ради о монографској публикацији.



Слика 2.4 Креирање новог записа у едитору

Приликом отварања новог записа у едитору, пре уноса било ког податка у запис могуће је променити тип обраде притиском на дугме са иконицом . Притиском на ово дугме добија се прозор приказан на слици 2.5. Оног тренутка када корисник унесе први податак у запис ово дугме престаје да буде активно и мора се наставити обрада по изабраном типу обраде. Промена типа обраде сада се може извршити тек након напуштања записа чија обрада је у току.



Слика 2.5 Промена типа обраде

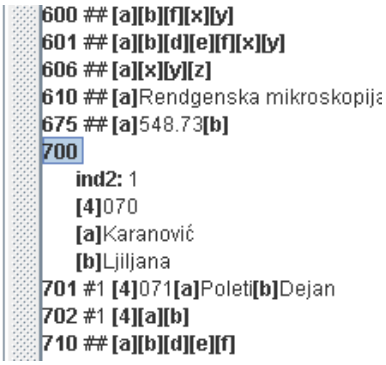
На прозору приказаном на слици 2.5 налази се списак свих типова обраде који су дефинисани за пријављеног библиотекара. За сваки тип обраде наводи се његов назив и у загради почетно слово типа публикације на који се односи. Подразумевани тип обраде за пријављеног библиотекара означен је црвеном бојом. Промена типа обраде постиже се тако што се селектује жељени тип обраде у понуђеној листи и притисне дугме *Potvrdi*.

Промена типа обраде подразумева промену стабла формата и стабла библиографског записа који се сада формирају на основу новоизабраног типа обраде.

2.5.1 Обрада записа у едитору

Обрада библиографских записа врши се над стаблом записа, док стабло библиотечког формата служи за селекцију елемената који се додају у запис. Поред тога, основна улога стабла библиотечког формата је да кориснику омогући увид у све информације које се тичу формата (називе поља, потпоља и индикатора, особине поновљивости, и друго).

Стабло библиографског записа приказује структуру записа, његова поља и потпоља као и садржај записа, односно садржај потпоља и индикатора. Притиском на тастер <Enter> отвара се селектовано поље. На слици 2.6 приказан је пример отвореног поља 700. На овај начин омогућава се кретање по индикаторима и потпољима отвореног поља.



Слика 2.6 Отворено поље 700

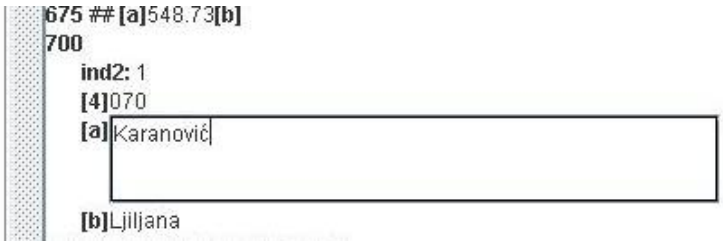
Над записом се у едитору могу вршити следеће операције:

- промена садржаја записа,
- унос садржаја потпоља и индикатора,
- промена структуре записа:
    - додавање поља и потпоља,

- брисање поља и потпоља,
- промена редоследа потпоља.

Промена садржаја записа

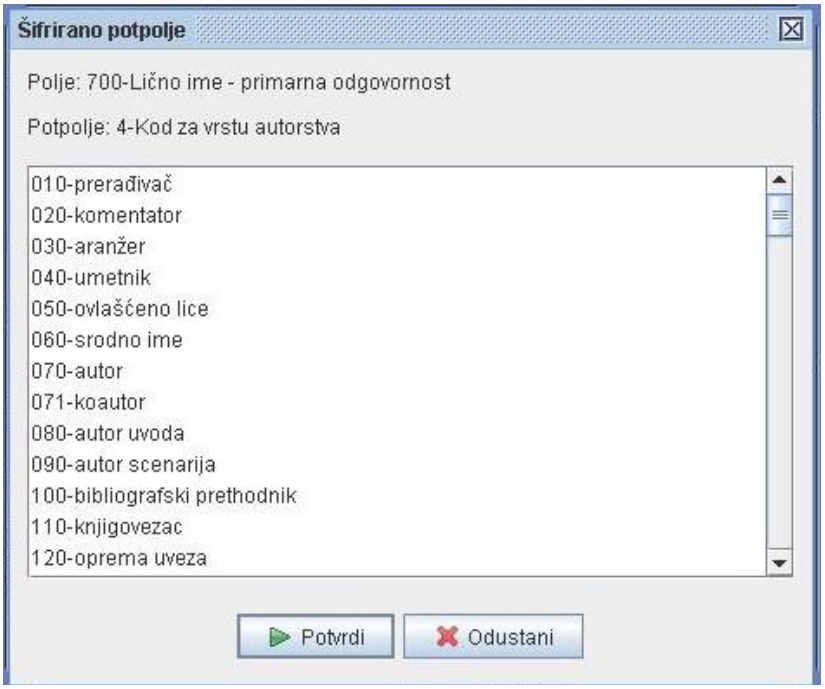
Разликују се две врсте едитора за унос садржаја. На слици 2.7 приказан је едитор за унос садржаја нешифрираног потпоља, а на слици 2.8 едитор за унос садржаја шифрираног потпоља. Едитор за унос вредности за индикаторе је исти као едитор за унос шифрираног потпоља који је приказан на слици 2.8.



Слика 2.7 Едитор за унос нешифрираног потпоља

Слика 2.8 Едитор за унос шифриране вредности

Уколико се ради о уносу шифрираних вредности, корисник може да унесе шифру, али може и да преузме вредност из одговарајућег шифарника који се отвара притиском на дугме са иконицом или пречицом са тастатуре. На слици 2.9 приказан је изглед шифарника за потпоље *4* поља *700*. Вредност индикатора уноси се на исти начин као и вредност шифрираног потпоља. Приликом уноса података врши се контрола унетог садржаја. За шифрирана потпоља проверава се исправност шифре, а за нека потпоља проверава се исправност формата у ком је податак унет (ISBN, ISSN број, унос датума и године). У случају да је нарушена валидност записа пријавиће се одговарајућа порука и садржај неће бити прихваћен.



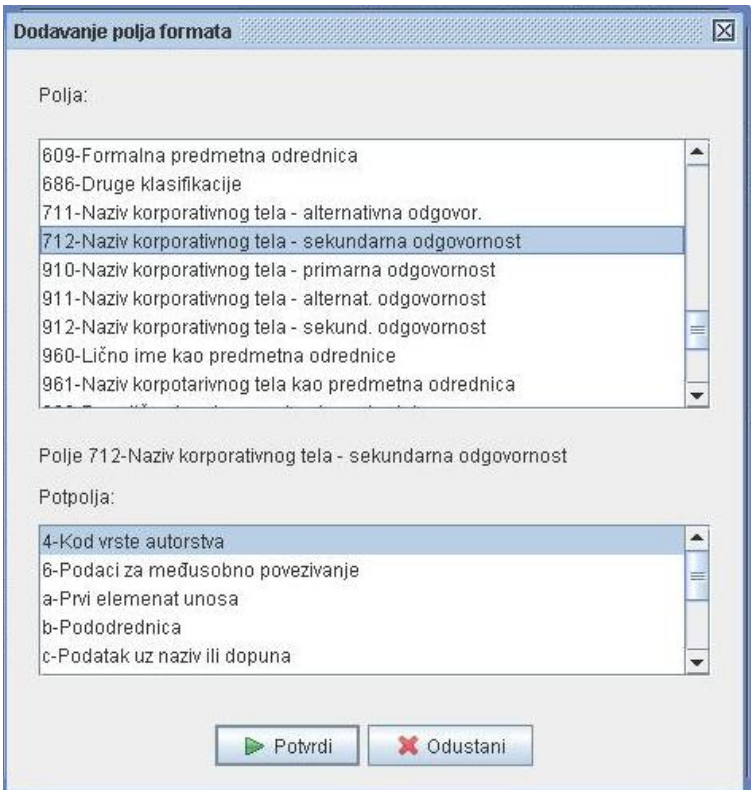
Слика 2.9 Шифарник потпоља 4 поља 700

Промена структуре записа

Код додавања поља и потпоља разликују се два случаја:

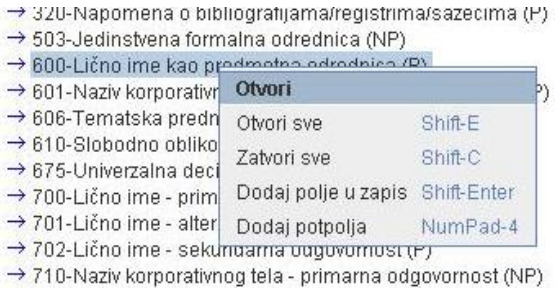
- додавање поља/потпоља које не постоји у запису и у стаблу формата и
- додавања поновљивог поља/потпоља које већ постоји у запису и у стаблу формата.

У првом случају операција додавања поља/потпоља врши се над стаблом формата, али се то поље/потпоље истовремено додаје и у запис.



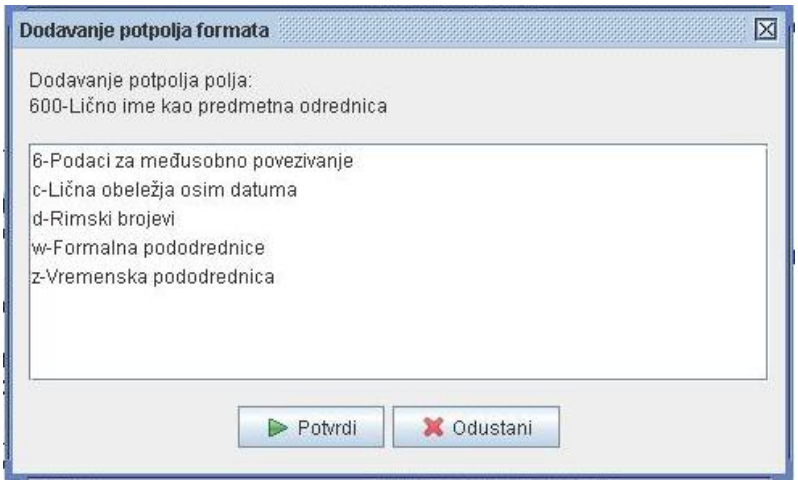
Слика 2.10 Додавање поља формата

На слици 2.10 приказан је прозор за избор поља за додавање у стабло формата. Приликом оваквог додавања поља за свако поље које се додаје мора се додати и макар једно потпоље које се селектује из доње листе. Додавање потпоља у стабло формата врши се избором одговарајуће акције (*Dodaj potpolja*) из падајућег менија који се добија десним кликом миша на одговарајуће поље (слика 2.11). Уколико селектовано поље нема додатних потпоља ова акција ће бити светлије обојена и неће бити активна.



Слика 2.11 Падајућа листа за поље у стаблу формата

На слици 2.12 приказан је прозор за избор потпоља за додавање. Изабрана потпоља додају се у стабло формата и у запис.



Слика 2.12 Додавање потпоља

Додавање поновљивог поља врши се тако што се поље које треба додати селектије у стаблу формата и позиве се одговарајућа акције. Акција за додавање поља покреће се акцијом *Dodaj polje u zapis* из падајућег менија који је приказан на слици 2.11. На овај начин у запис ће се додати поље које је селектовано у стаблу формата. Додавање поља у запис подразумева и додавање индикатора са њиховим подразумеваним (*default*) вредностима. Уколико се додаје поље које већ постоји у запису, а нема особину поновљивости пријављује се порука о грешци.

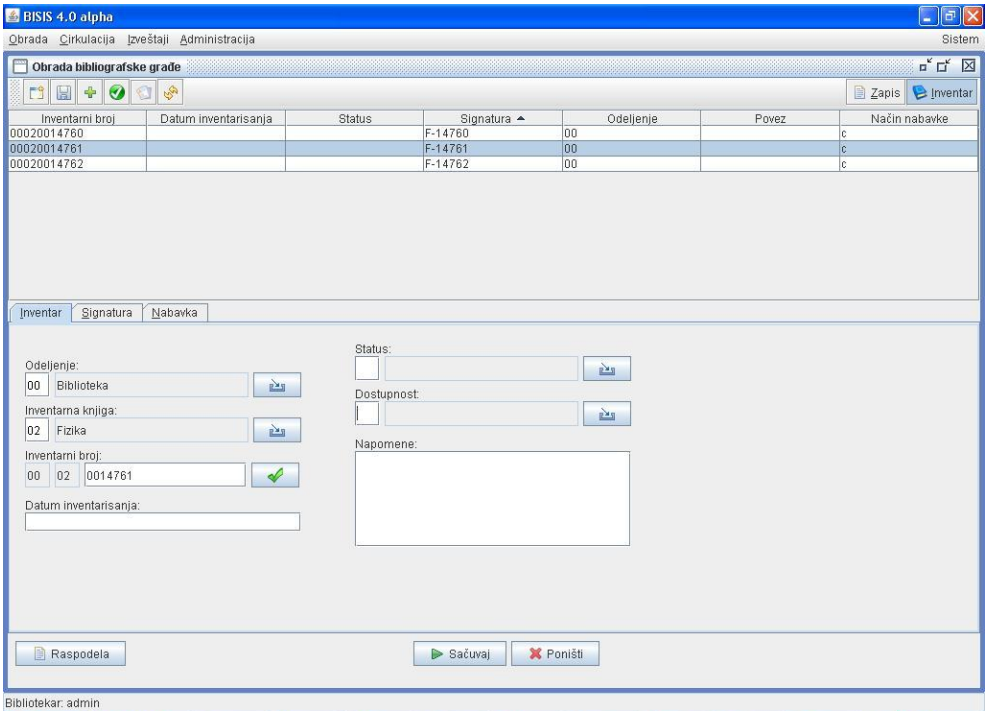
Потпоље се у запис додаје тако што се у стаблу записа прво селектује поље у које треба додати потпоље, затим се потпоље изабере у стаблу формата и притисне се одговарајући тастер. На овај начин потпоље се додаје у поље које је селектовано у стаблу записа. Уколико корисник покуша да дода потпоље које није поновљиво, а потпоље са тим називом већ постоји у селектованом пољу пријавиће се одговарајућа порука.

Брисање поља и потпоља врши се селекцијом одговарајућег елемента који треба да се обрише у стаблу записа и притиском на одговарајуће команде са тастатуре. Уколико је елемент који се брише обавезан, пријављује се одговарајућа порука и елемент не може бити обрисан.

Промена редоследа потпоља у оквиру поља врши се у облику померања селектованог потпоља за једно место испред или иза помоћу пречица са тастатуре.

2.5.2 Обрада локацијских података

Поред креирања библиографског записа, обрада библиографске грађе обухвата и унос локацијских података, односно података о конкретним примерцима публикације које библиотека поседује. Екранска форма за унос локацијских података за библиографску јединицу која се обрађује у едитору добија се притиском на дугме *Inventar* у горњем десном углу прозора едитора. Изглед екранске форме за унос локацијских података за монографске публикације дат је на слици 2.13.



Слика 2.13 Екранска форма за унос локацијских података

Према YUMARC формату за локацијске податке је резервисано поље *996* у случају монографских публикација, односно поље *997* у случају серијских публикација. Едитор за обраду библиографске грађе у четвртој верзији система БИСИС реализован је тако да се локацијски подаци уносе независно од библиографског формата у релациону базу података, па и одговарајућа екранска форма представља стандардну графичку апликацију за унос података у релациону базу података.

Екранска форма за унос локацијских података састоји се из три дела. У горњем делу налази се табела са основним подацима за све примерке који су

унети за пубикацију која се обрађује у едитору. У средњем делу налази се форма за унос података о једном примерку, а у доњем дугмад са акцијама за чување примерака, иницијализацију екранске форме и отварање форма за расподелу примерака.

Над табелом за приказ свих примерака може се извршити сортирање по било којој колони и то или у опадајућем или у растућем редоследу. Над овом табелом могуће је извршити и брисање примерка. Селекцијом једног примерка у табели примерака тај примерак се учитава у доњи део екранске форме и могућа је модификација података за селектовани примерак. На слици 2.13 приказана је екранска форма за унос локацијских података са једним учитаним примерком.

2.5.3 Провера валидности записа

Приликом снимања записа врши се провера његове валидности. Уколико запис није валидан (на пример нису унета сва обавезна потпоља) он не може бити снимљен и кориснику се пријављује одговарајућа порука.

Корисник-библиотекар може у току рада да провери валидност свог записа. Након позива одговарајуће акције кориснику се приказује извештај о валидности записа. На слици 2.14 приказане су поруке које се пријављују као извештај о валидности записа. На слици лево приказана је порука у случају да је запис валидан а десно је приказана порука која се пријављује у случају да запис није валидан.

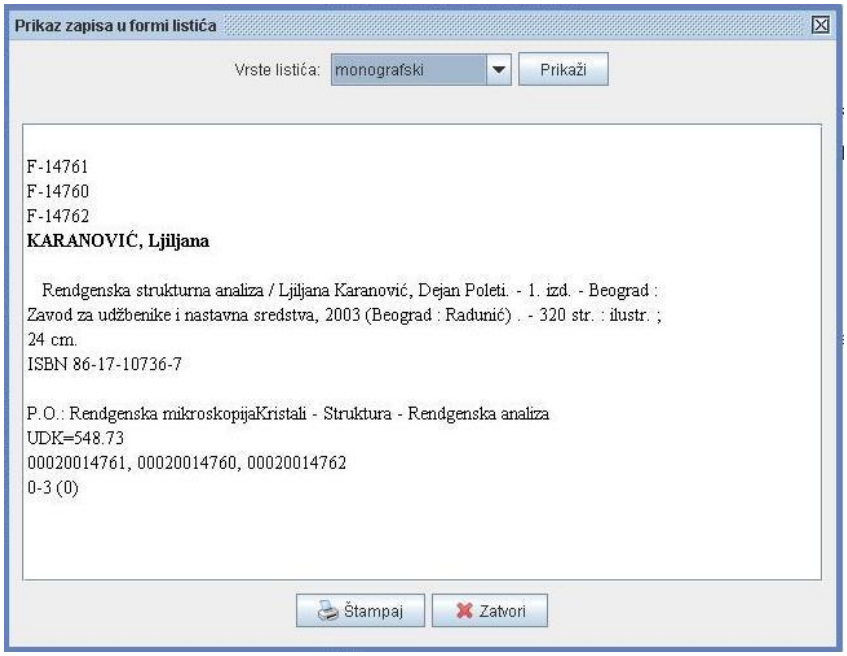
Слика 2.14 Извештаји о валидности записа

2.5.4 Приказ каталошких листића

Запис који је креиран у едитору може се кориснику приказати у облику каталошког листића. Генерисање каталошког листића извршено је коришћењем софтверског пакета *Report* који је део система БИСИС и наведен је у одељку 2.4.

На слици 2.15 приказан је прозор за приказ записа у форми листића. За сваку библиотеку дефинисан је падразумевани листић који се приказује приликом отварања овог прозора. Постоји могућност да се из падајуће листе са врстама

листића изабере нека друга врста листића и притиском на дугме *Prikaži* на белој површини форме приказује за запис у изабраној форми листића.



Слика 2.15 Приказ записа у форми листића

Поглавље 3

Опис коришћене методологије и софтверског окружења

Методологија коришћена у развоју система за каталогизацију је обједињени процес (*unified process*), а имплементација је реализована у Eclipse окружењу и програмском језику Јава.

3.1 ОПИС КОРИШЋЕНЕ МЕТОДОЛОГИЈЕ

Обједињени процес (*unified process*) је методолошки поступак за развој објектно-оријентисаних система (Jacobson, et al., 1999). Основне карактеристике обједињеног процеса развоја софтвера су:

- итеративни и инкрементални развој (*iterative and incremental develipment*),
- вођен случајевима коришћења (*use case driven*),
- софтверска архитектура заузима централно место (*architecture centric*).

Итеративни и инкрементални приступ развоју софтвера сматра се најзначајнијом карактеристиком обједињеног процеса (Larman, 2002). Основна идеја овог приступа састоји се у томе да је развој софтвера подељен на низ краћих мањих пројеката који се називају итерације. Резултат сваке од итерација је један инкремнт односно софтверски производ који је тестиран, интегрисан и извршив (*executable*). Итеративни развој софтвера заснован је на постепеном проширивању и дотеривању система кроз више итерација. Свака од итерација може се реализовати коришћењем различитих методологија развоја софтвера.

У обједињеном процесу развоја софтвера користе се случајеви коришћења за опис функционалних захтева система. За сваку итерацију у развоју дефинише се скуп случајева коришћења који се реализују.

У обједињеном процесу развоја софтвера највећа пажња посвећена је софтверској архитектури система која се развија. У раним фазама развоја софтвера формира се језгро архитектуре софтвера које садржи основу дизајна и подистеме са њиховим интерфејсима и одговорностима у систему.

Развој софтвера за каталогизацију извршен је у три итерације у којима су коришћене следеће методологије развоја софтвера:

- развој вођен моделима (*model-driven development*),
- развој заснован на компонентама (*component-based development*),
- развој заснован на XML-у.

Развој софтвера вођен моделима састоји се од креирања апстрактних модела софтверских система на основу којих се генерише извршни програмски код. Овај приступ у развоју софтвера ослања се на технологију Model Driven Architecture (MDA) [38] коју је предложио OMG конзорцијум и чини је скуп концепата и стандарда. MDA технологија одваја пословну и апликативну логику од платформе на којој се развија софтвер. У овом приступу развоја софтвера модели имају кључну улогу (Gorton, 2006). Они обезбеђују апстракцију система и омогућавају анализу система са различитих становишта и нивоа апстракције. У MDA технологији модели нису само део документације софтверског система, већ и део имплементације јер се на основу њих аутоматизованим поступцима генерише програмски код који се извршава на циљној платформи. На овај начин обезбеђене су следеће добре карактеристике софтвера: преносивост (*portability*), компатибилност (*interoperability*) и могућност поновног коришћења софтвера (*reusability*).

Развој софтвера заснован на компонентама је једна грана софтверског инжењерства чији је приоритет раздвајање система на мање функционалне целине, односно софтверске компоненте [39]. Под појмом софтверска компонента подразумева се софтверски пакет или модул који реализује одређену функционалност, чија имплементација је скривена а комуникација са осталим компонентама се реализује преко интерфејса.

Развој софтвера заснован на XML технологијама подразумева употребу XML-а и XML технологија, као што су XMLSchema и XSLT у различитим аспектима развоја софтвера.

3.2 ОПИС СОФТВЕРСКОГ ОКРУЖЕЊА

Eclipse [26] је софтверска платформа дизајнирана за креирање и интеграцију софтверских алата који се користе у развоју разних врста софтверских апликација. Према основном дизајну Eclipse не нуди много крајњем кориснику, већ је првенствено намењен за оне који развијају софтвер. Међутим, главна вредност овог окружења састоји се у томе што подстиче развој независних подсистема који се заснивају на plug-in моделу. Eclipse обезбеђује отворену архитектуру која омогућава интеграцију различитих plug-in-ова. На овај начин, обезбеђено ја да се они који развијају plug-in

концентришу само на своју област не водећи рачуна о општој архитектури система.

У оквиру Eclipse платформе развијено је окружење за моделирање софтверских система Eclipse Modeling Framework (EMF) [32]. EMF је софтверско окружење које омогућава развој софтвера генерисањем програмског кода на основу структуираног модела података. EMF заузима централно место у развоју система за каталогизацију јер ће објектни модел библиографског запис у оквиру система бити креиран на основу EMF модела. Поред тога, EMF представља подршку концепту развоја софтвера вођеним моделима и представља основу за развој других пројеката из ове области, као што је и OpenArchitectureWare (oAW) [33] који ће се користити за спецификацију система за каталогизацију.

OAW је платформа за развој софтвера вођен моделима. Ова платформа подржава парсирање модела и фамилије језика и генерисање програмског кода на основу њих. Основне карактеристике овог окружења су да је развијено као скуп plug-in-ова за Eclipse, да се заснива на EMF концепту метамоделовања и да се све трансформације и генерисања специфицирају путем токова послова (*workflow*).

Део oAW окружења је софтверски алат Xtext који омогућава спецификацију доменски специфичних језика и генерисање Eclipse текстуалног едитора за те језике. Поред тога, у оквиру oAW платформе развијен је језик за проширење доменски специфичних језика, то је језик Xtend. Језик Xtend је основа за језик *Check* који служи за спецификацију ограничења над језиком и језик Xpand који се користи за спецификацију темплејта за трансформацију језика. У оквиру Xtext окружења постије текстуални едитори у којима се уносе спецификације у свим наведеним језицима. Алат Xtext, као и језици Xtend, Check i Xpand детаљније су описани у овом поглављу.

Парсери за доменски специфичне језике се праве коришћењем неког од већ постојећих парсер генератора (ANTLR, JavaCC, Lex/yacc) или ручним програмирањем парсера. Улога парсер генератора је да генерише одговарајући парсер за текст на основу граматике језика. У софтверском алату Xtext изабран је парсер генератор ANTLR.

ANTLR – Another Tool For Language Recognition [34] је софтверски алат за рад са језицима. ANTLR је окружење за генерисање алата за препознавање, интерпретацију, компајлирање и превођење на основу специфициране граматике језика. ANTLR има снажну подршку за рад са синтаксним стаблима.

У овом поглављу описано је софтверско окружење које је коришћено за имплементацију система за каталогизацију у овој дисертацији. Основни едитор за каталогизацију генерисан је у Xtext технологији што је описано у поглављу 5. Овај едитор генерисан је у облику plug-in-а за Eclipse и може се дупунити додатним функцијама система за каталогизацију коришћењем plug-in технологије што је описано у поглављу 6.

3.3 EMF – ECLIPSE MODELING FRAMEWORK

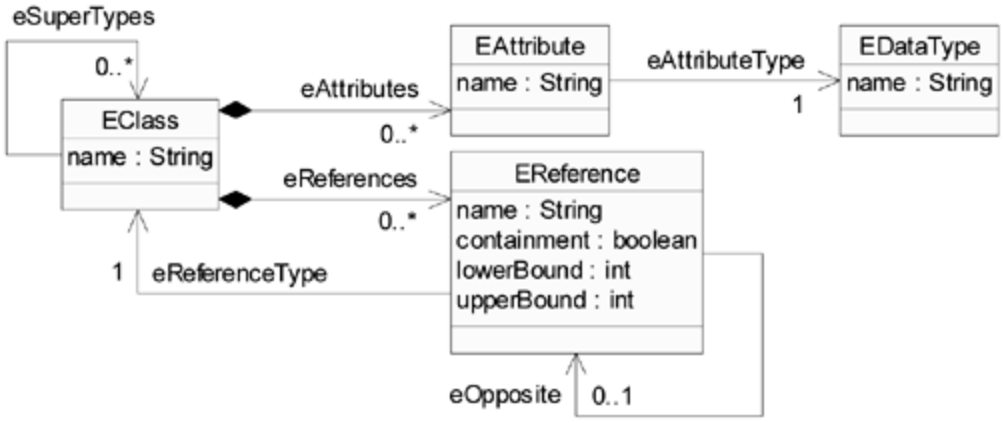
EMF је софтверско окружење за моделирање у оквиру Eclipse платформе (Budinsky et al., 2003). EMF омогућава развој софтвера на основу модела података. Овај одељак садржи опис основних концепата EMF-а који ће се користити у развоји система за каталогизацију.

3.3.1 Ecore

Модел који се користи за спецификацију EMF модела назива се *Ecore*. *Ecore* је и сам по себи EMF модел и он представља метамодел за EMF модел. На слици 3.1 приказан је упрошћени подскуп *Ecore* метамодела који представља језгро (*kernel*) целе спецификације. Модел је приказан преко UML дијаграма класа.

На дијаграму су представљене четири класе:

- *EClass* представља једну класу модела која има своје име, може имати нула или више атрибута и нула или више референци. Да би се подржало наслеђивање, класа се може референцирати на више других класа које наслеђује.
- *EAttribute* представља атрибут класе, има своје име и тип.
- *EDataType* представља тип атрибута, може бити примитиван тип као што је *int* или сложен као што је датум.
- *EReference* представља један крај асоцијације између две класе, има своје име, логички атрибут *containment* који дефинише да ли се ради о вези композиције и референцу на класу која представља други крај асоцијације. Уколико је могућа навигација у супротном смеру асоцијације постоји рефренца на још једну асоцијацију. За асоцијацију се дефинише и кардиналитет помоћу доње границе (*lowerBound*) и горње граница (*upperBound*) броја појављивања.



Слика 3.1 *Ecore* језгро - презето из (Budinsky et al, 2003)

Имена класа на дијаграму са слике 3.1 веома подсећају на термине из UML-а јер UML и јесте обједињени језик моделирања. Ecore је само једноставнији и мањи подскуп целог UML-а који се односи на моделирање класа.

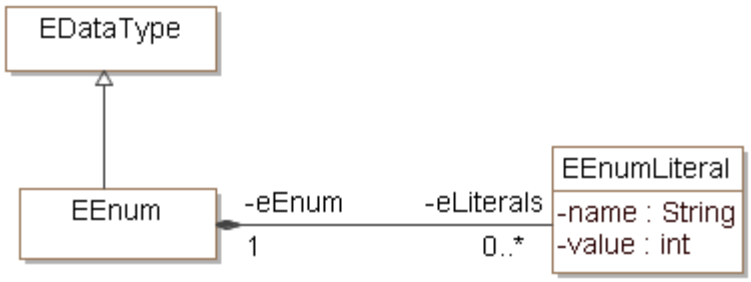
На слици 3.1 приказано је само језгро EMF метамодела, цео метамодел је много сложенији и садржи велики број класа. Тако на пример, класе се могу груписати у пакете који су моделирани класом *EPackage*. Пакет има своје име, а јединствена идентификација пакета добија се на основу његове URI адресе.

И класе које су представљене на дијаграму са слике 3.1 имају велики број атрибута које могу да се специфицирају а нису наведене на дијаграму. Тако на пример атрибути и референце имају много заједничких особина, као што су обавезност, подразумевана (*default*) вредност, јединственост (*unique*), и слично.

Класе могу имати и дефинисане операције које имају своје параметре, а могу и прослеђивати изузетке.

Класом *EDataType* представљен је једноставан тип података. *Ecore* има велики број уграђених типова података који су инстанце класе *EDataType*. Ови уграђени типови одговарају најчешће коришћеним типовима из програмског језика Јава. Неки од уграђених типова су: *EBoolean*, *EByte*, *EChar*, *EInt, EString*.

Постоји и специјалан случај једноставног типа, а то је набројиви тип (*enumerated type*). На слици 3.2 представљена је структура набројивог типа.



Слика 3.2 Набројиви тип - преузето из (Budinsky et al., 2003)

Класа *EEnum* представља модел набројивог типа и она садржи нула или више литерала који су представљени класом *EEnumLiteral*. Сваки литерал има своје име које је представљено атрибутум *name* и дефинише своју бројчану вредност у атрибуту *value*.

3.3.2 Креирање EMF модела

Креирање EMF модела може се коришћењем Eclipse-ових софтверских алата извршити на неколико начина:

- трансформацијом спецификације модела дате у облику XML шеме,
- генерисањем модела на основу Јава интерфејса, при чему операције интерфејса морају имати специјалне коментаре,
- генерисањем на основу UML дијаграма који је специфициран у одређеним софтверским алатима,
- директним едитирањем EMF модела у Eclipse Ecore едитору који је јавно доступан.

Поред наведених начина креирања EMF модела, постоје и одређени софтеврски алати који подржавају креирање EMF модела на основу различитих облика спецификације. Један од таквих алата је и Xtext који ће бити коришћен у развоју система за каталогизацију.

3.4 OAW XTEXT

Xtext је део oAW платформе који служи за развој доменски специфичних језика на основу спецификације дате у нотацији која је слична проширеној Backus-Naur-Form (EBNF) нотацији са том разликом што Xtext подржава и опис апстрактне синтаксе односно метамодела.

На основу граматике XText генерише:

- парсер заснован на ANTLR,
- Eclipse modeling framework (EMF) модел,

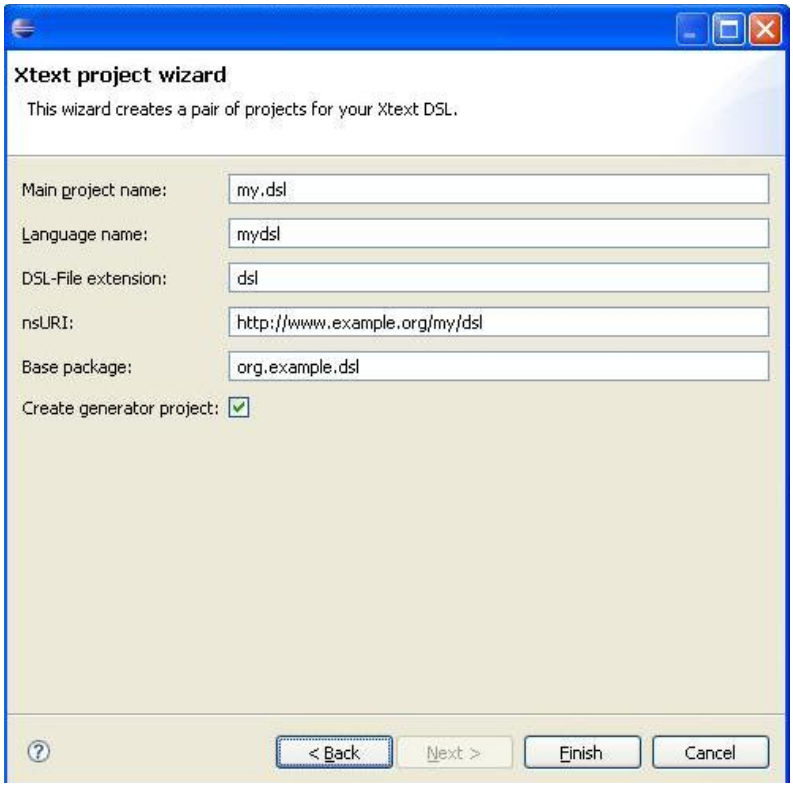
- Eclipse едитор који подржава:
    - бојење синтаксе,
    - комплетирање уноса (помоћ приликом уноса),
    - проверу ограничења над уносом у реалном времену (real-time).

XText је заснован на plug-in технологији Eclipse-а. Приликом креирања Xtext пројекта у Eclipse-у се појављује Wizard приказан на слици 3.3 у ком се специфицирају основни подаци о пројекту и језику који се креира као што су назив пројекта, назив доменски специфичног језика који се креира, екстензија фајлова у који ће се уносити садржај у креираном језику, итд. Потврдом унетих података креирају се три пројекта која су међусобно зависна, то су следећи пројекти:

- my.dsl – пројекат у ком је дефинисан сам језик, у њему се дефинише граматика за доменски специфичан језик, након покретања Xtext генератора овај пројекат ће садржати парсер за креирани језик;
- my.dsl.editor – пројекат који садржи едитор за доменски специфичан језик, он је на почетку празан а попуниће се након креирања граматике језика покретањем Xtext генератора;
- my.dsl.generator – садржи оквир за генерисање кода на основу модела написаног у креираном доменски специфичном језику, у њему се могу дефинисати разни темплејти за трансформацију фајлова написаних у креираном доменски специфичном језику.

Сва три наведена пројекта развијају се као plug-in пројекти, односно пројекти у којима се развија plug-in за Eclipse. Као резултат сваког од ових пројеката добија се plug-in који се користи тако што се укључи у Eclipse окружење.

Централно место у развају доменски специфичног језика у Xtext окружењу представља креирање граматике тог језика. Граматика језика се у нотацији сличној EBNF уноси у фајл *mydsl.xtxt* који је генерисан у оквиру пројекта *my.dsl*. На основу граматике наведене у овом фајлу ће се генерисати парсер за језик, EMF модел, фајл за спецификацију ограничења над језиком, као и Eclipse текстуални едитор за унос података у креираном језику.



Слика 3.3 Xtext визард за креирање новог пројекта

3.4.1 Спецификација граматике доменски специфичног језика

Едитирање фајла са екстензијом *xtext* у оквиру Xtext окружења се врши у специјалном Eclipse едитору који подржава бојење синтаксе и помоћ приликом уноса.

Xtext спецификација граматике састоји се од скупа правила. Правило започиње својим именом иза кога се наводи двотачка („:“) и опис правила која се завршава знаком тачка-зарез („;“).

Постоји неколико врста правила, а то су:

- правило за дефинисање типа (*Type rule*)
- native правила
- набројиво правило (*Enum rule*)

Правило за дефинисање типа

Правило за дефинисање типа је најчешће коришћено правило за спецификацију граматике у Xtext окружењу.

Опис правила за дефисање типа се састоји од токена. Токени у XText-у могу бити:

- кључне речи и
- атрибути правила (*properties*).

Кључне речи се у граматици наводе као стрингови у оквиру наводника и садрже речи и симболе који су део доменски специфичног језика.

Атрибути се наводе тако што се називу атрибута додели тип атрибута, при чему постоје следеће врсте доделе типа:

- *property=Type* – једнострука додела
- *property+=Type* – вишеструка додела
- *property?="keyword"* – логичка додела

Атрибути у оквиру правила могу имати следеће кардиналитете:

- ? - 0..1 ,
- - 0..n ,
- + - 1..n ,
- кардиналитет није наведен - 1..1.

Пример једног правила којим се дефинише тип:

```
Entity :
(isAbstract?="abstract")? "entity" name=ID "{"
(features+=Feature)*
"}";
```

Назив овог правила је *Entity* и њиме је представљен један тип података у доменски специфичном језика, а то је ентитет. Ентитет може започињати кључном речи *abstract* којом је дефисана вредност логичког атрибута *isAbstract* који одређује да ли се ради о апстрактном ентитету. Кардиналитет овог атрибута је означен знаком „?“ који се налази иза затворене заграде и њиме је дефинисано да се овај атрибут може јавити највише једном, а не мора се појавити ни једном у оквиру ентитета. Следи кључна реч *entity* иза које следи име ентитета представљено атрибутом *name* које је уграђеног типа *ID* који значи идентификатор. Пошто није наведен кардиналитет за атрибут *name*, његов кардиналитет је 1..1, што значи да са назив ентитета мора тачно једном појавити у ентитету. Иза назива следи кључна реч која означава почетак блока за дефинисање ентитета („{“). Ентитет садржи више, а не мора ни једну особину које су представљене правилом *Feature* а атрибут који садржи листу

правила је *features*. Правило *Feature* се посебно дефинише у оквиру исте граматике. Дефиниција ентитета се завршава кључном речи "}".

Пример инстанце наведеног правила је:

```
entity Person {
 String name
 String lastName
 Address home
}
```

Овим фрагментом је описан конкретан ентитет , чији назив је *Person* и има три особине, *name*, *lastName* и *home*. Овај ентитет није апстрактни јер није наведена кључна реч *abstract*.

Поред правила за дефинисање типова, често се користе још две врсте правила, то су native и набројива правила.

Native правила

Native правила се наводе у виду стринга у ANTLR синтакси. Ова правила се не обрађују приликом генерисања парсера, већ се непромењена прослеђују парсеру. Ова врста правила се користи када је потребно дефинисати лексичка правила која се не могу изказати Xtext синтаксом или када је потребно редефинисати неко од уграђених лексичких правила. Дефиниција native правила започиње резервисаном речју *Native* иза које следи назив native правила, затим двотачка и стринг у ANTLR нотацији.

Једно од уграђених лексичких правила је правило WS којим се одређује сепаратор између токена и он је дефинисан на следећи начин:

```
WS : "(' '|'\t'|'\r'|'\n')+ {$channel=HIDDEN;}"
```

Стринг *(' '|'\t'|'\r'|'\n')+ {$channel=HIDDEN;}* је написан у ANTLR нотацији, израз *{$channel=HIDDEN;}* говори да се наведени симболи занемарују приликом парсирања текста. У случају да неки од наведених карактера треба укључити у граматику, односно да се тај карактер узима у обзир приликом парсирања текста треба редефинисати ово правило, тако што ће се креирати ново native правило WS које не садржи посматрани карактер.

На пример, ако треба из правила WS искључити карактер '\r' у Xtext фајл у коме се специфицира граматика треба додати следеће правило:

```
Native WS : "(' '|'\t'|'\n')+ {$channel=HIDDEN;}"
```

Парсер који има дефинисано овакво правило ће карактер ‘\r’ узимати у обзир приликом парсирања текста.

Набројива правила

Набројива правила (*Enum rules*) се користе за дефинисање ограниченог скупа предефинисаних алтернатива. Дефиниција набројивог правила започиње резервисаном речју *Enum* и садржи литерале који си раздвојени знаком за алтернативу – ’|’. Сваки литерал у набројивом типу има назив токена и стринговску репрезентацију. Стринговска репрезентација литерала се у парсеру третира као кључна реч. Набројива правила се у оквиру граматике могу користити исто ако и сва друга правила, на пример могу се доделити као тип атрибута.

Пример дефиниције набројивог правила:

```
Enum DataType :
 String="string" | Integer="int" | Boolean="bool";
```

Набројиво правило има назив *DataType* и може имати једну од три вредности: *string*, *int* или *bool*.

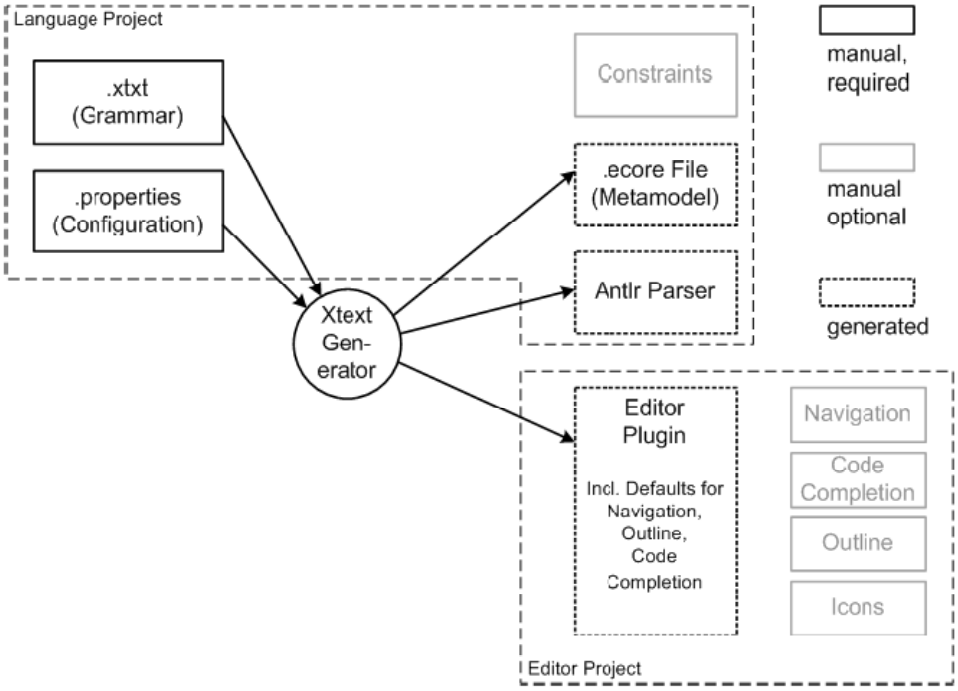
3.4.2 Xtext генератор

У оквиру Xtext окружења реализован је Xtext генератор који на основу граматике генерише остале софтверске компоненте за доменски специфичан језик, као и plug-in који садржи Eclipse текстуални едитор за унос података у креираном језику.

Генерисање се у Xtext-у специфицира дефинисањем токова послова (*workflow*) у виду XML конфигурационих фајлова.

На слици 3.4 графички је приказана архитектура улазно-излазнog система Xtext генератора (Kolb i Voalter, 2008). Целокупна архитектура подељена је на два дела, то су део који се односи на пројекат за развој доменски специфичног језика (*Language Project*) и део који се односи на пројекат за развој едитора за тај језик (*Editor Project*).

Фајлови које Xtext генератор захтева као улаз су фајл у ком је специфицирана граматика језика, а то је фајл са екстензијом *xtext* и фајл у ком су наведени параметри везани за пројекат у ком се генерише језик и који су унети у Wizard-у приказаном на слици 3.1, и то је фајл са екстензијом *properties*. На основу наведених улазних фајлова генератор креира софтверске компоненте које се односе на развој доменски специфичног језика и компоненте за развој едитора за тај језик (Слика 3.4).



Слика 3.4 Xtext генаратор – преузето из (Kolb i Voalter, 2008)

Међу генерисаним компонентама за развој језика налазе се: *Antlr* парсер за дефинисани језик у облику Јава пакета, EMF метамодел у фајлу са екстензијом *ecore*, и окружење за дефинисање ограничења над креираним језиком.

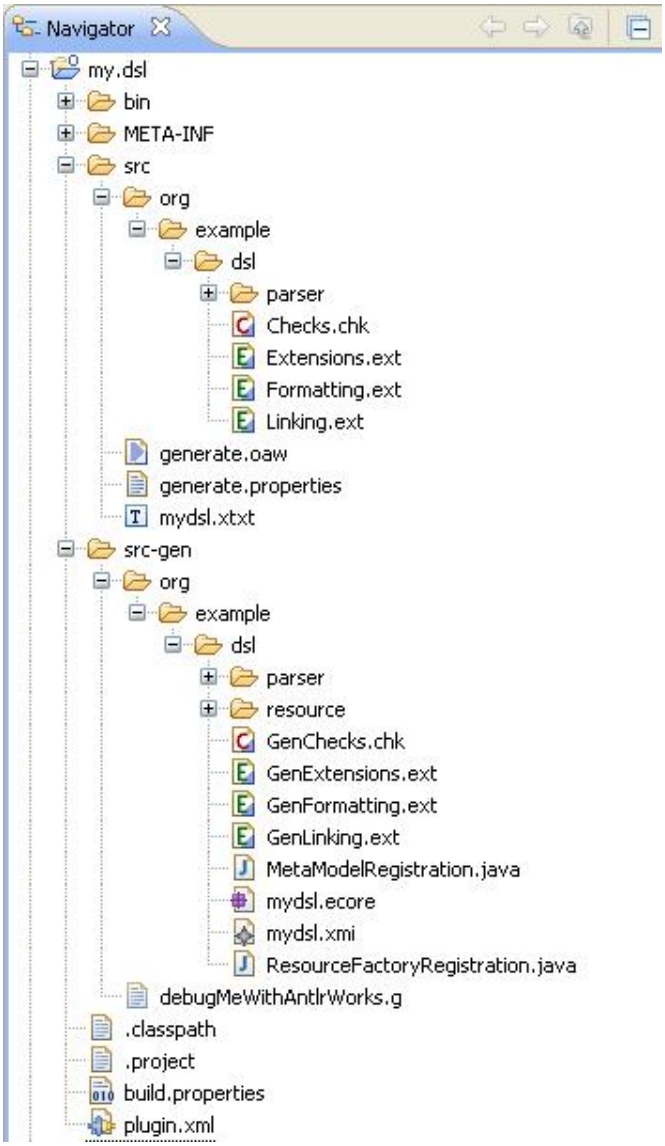
Едитор за креирани језик је генерисан у облику Eclipse plug-in пројекта и обухвата софтверско окружење у ком се основни едитор може проширити додатним функционалностима као што су навигација, односно брзо кретање кроз унети текст (*Navigation*), допуна кода (*Code Completion*), структурни поглед (*Outline*) и иконице које ће бити део екранске форме едитора (*Icons*).

Конкретни фајлови који су генерисани Xtext генератором распоређени су у три пројекта, која су на почетку инстанцирана: *my.dsl*, *my.dsl.editor* и *my.dsl.generator*.

На слици 3.5 приказана је архитектура пројекта *my.dsl* у Eclipse navigator погледу. Пројекат *my.dsl* односи се на сам доменски специфичан језик који се развија и у њему се налази пакет који имплементира парсер за језик, то је пакет *parser*. Поред тога, у њему се могу специфицирати ограничења над

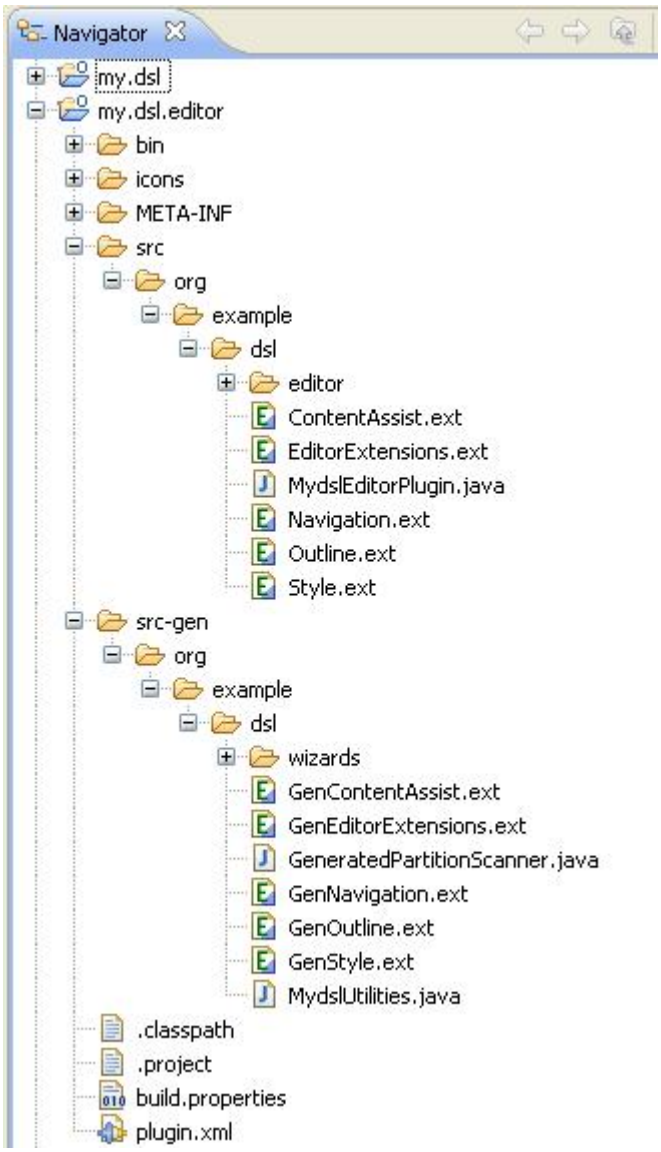
језиком у фајлу *Check.chk*, као и разне врсте проширења, односно помоћних операција над језиком у фајлу *Extensions.ext*.

Фајл *mydsl.xtext* садржи спецификацију граматике језика на основу које су генерисани сви остали фајлови. Све промене над спецификацијом језика се раде у фолдеру *src* док се у фолдеру *src-gen* смештају компајлиране верзије спецификација које се аутоматски генеришу и користе у извршној верзији пројекта. Ту се налазе и генерисани EMF модел у фајлу *mydsl.ecore*



Слика 3.5 Структура пројекта за развој језика

На слици 3.6 приказана је архитектура пројекта *my.dsl.editor* који се односи на развој едитора за унос текста према дефинисаном језику. У њему се специфицирају особине генерисаног едиотора као што су допуна текста у фајлу *ContentAssist.ext*, навигација кроз унети садржај у фајлу *Navigation.ext*, подешавање структурног приказа у фајлу *Outline.ext*.



Слика 3.6 Структура пројекта за развој едитора

Трећи генерисани пројекат садржи темплејте за трансформацију текста који се уноси према специфицираној граматици и ти фајлови су са екстензијом *xpt*.

3.4.3 Генерисање EMF модела

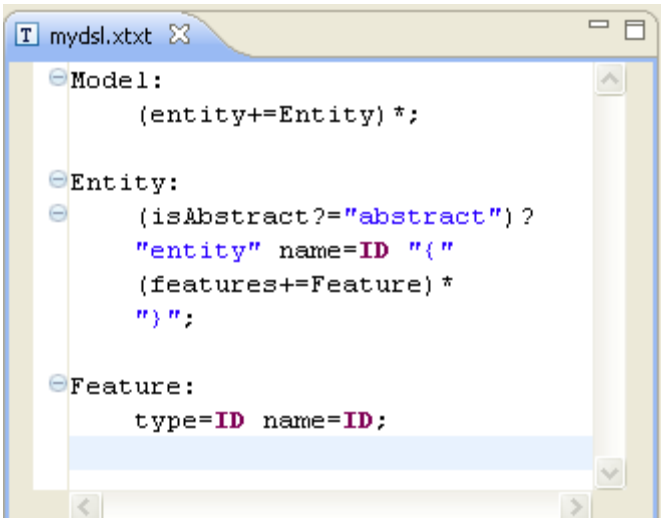
Један од софтверских докумената који се добија као резултат генерисиња Xtext генератора је EMF модел који је садржај фајла са екстензијом *ecore*. Овај фајл садржи спецификацију модела који описује деменски специфичан језик. Наведени EMF модел представља основу за генерисање меморијске структуре доменски специфичног језика, која ће се користити за рад са садржајем едитора, како у оквиру самог Xtext окружења тако и у оквиру система у који се едитор интегрише.

Генерисани EMF модел дефинише модел података којим је описана структура текста написаног у доменски специфичном језику. Овај модел се креира на основу Xtext граматике на следећи начин:

- правило за дефинисање типа постаје класа у EMF моделу – назив правила постаје назив класе, а атрубути тог правила постају атрибути класе,
- ако се за правило специфицира атрибут чији тип представља неко друго правило креира се референца, односно веза асоцијације за класу правила које се специфицира према класи која представља тип атрибута; тако креирана веза ће имати особину *containment* односно представљаће везу композиције,
- набројива правила постају набројиви типови у EMF моделу,
- тип *INT* из Xtext граматика постаје *EInt* у EMF моделу, а сви остали типови постају *EString*,
- атрибути који су специфицирани логичком доделом постају типа *EBoolean*,
- кључне речи које су део граматике не улазе у састав резултујућег EMF модела,
- native правила која су дефинисана у граматици не улазе у састав резултујућег EMF модела.

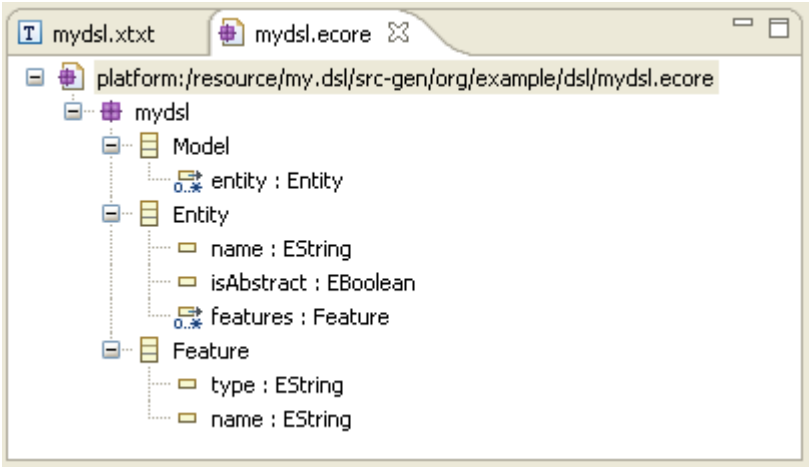
Генерисање EMF модела на основу граматике у наставку ће бити објашњено на једном примеру.

На слици 3.7 дат је пример спецификације једног доменски специфичног језика у облику Xtext граматике. Према овој граматици дефинише се језик за креирање модела, модел се састоји од више ентитета. Ентитет је специфициран правилом *Entity*. Ентитет има логички атрибут *isAbstract*, има своје име и скуп особина (*features*). Једна особина ентитета је представљена правилом *Feature* које има своје име и тип.



Слика 3.7 Пример Xtext граматике

Када се Xtext граматика приказана на слици 3.7 проследи Xtext генератору добија се EMF модел приказан на слици 3.8. EMF модел на слици 3.8 приказан је у оквиру Eclipse Ecore едитора у ком се EMF модел приказује у облику стабла. Сваки елемент стабла представљен је одговарајућом иконицом која графички описује тип елемента модела и лабелом која садржи назив елемента, а у случају атрибута и референци и тип елемента.



Слика 3.8 Пример EMF модела

Коренски елемент овог стабла је URI адреса модела. Коренски елемент садржи елемент *mydsl* којим је представљен EMF пакет чији садржај су класе

креиране на основу Xtext граматике. Пакети су у стаблу EMF модела представљени иконицом  .

EMF модел приказан на слици 3.8 садржи три класе које су креиране на основу правила за дефиницију типа, то су класе *Model, Entity* и *Feature*. Класе су у стаблу EMF модела обележене иконицом  .

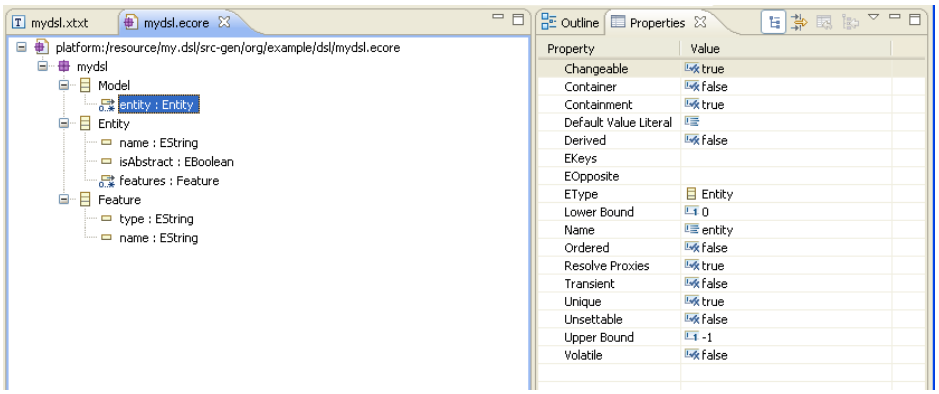
У Xtext граматици дефинисан је тип *Model* који садржи вишеструку појаву инстанци типа *Entity*. Овај случај пресликан је на везу асоцијације у EMF моделу која је у стаблу EMF модела представљена знаком  . Назив ове асоцијације је *entity*. Други крај везе представља класа *Entity*. Кардиналитет везе је *0..** што значи да EMF класа *Model* садржи нула или више инстанци EMF класе *Entity*.

EMF класа *Entity* садржи два атрибута и једну референцу. Атрибути су означени иконицом . Атрибут *name* је типа *EString* добијен на основу типа *ID* који је додељен атрибуту *name* у правилу *Entity* граматике на слици 3.7. Правило *Entity* има још један атрибут, то је атрибут *isAbstract* који је дефинисан логичком доделом. Атрибут *isAbstract* је пресликан у истоимени атрибут EMF класе *Entity*, а додељен му је тип *EBoolean*. Класа *Entity* има и једну референцу на класу *Feature* и та референца је представљена атрибутом *features*.

У EMF моделу који је приказан на слици 3.8 представљена је још једна класа, то је класа *Feature* настала на основу истоименог правила у спецификацији граматике. Ова класа има два атрибута који су типа *EString*.

Поред приказа структуре модела Eclipse Ecore едитор нуди и приказ и едитирање особина (*properties*) појединачних елемената модела. На слици 3.9 приказан је овај едитор у ком је са десне стране отворен прозор за едитирање особина асоцијације *entity* која је дефинисана за класу *Model*. У наведене особине спадају особина *containment* којом се дефинише да ли је ово веза композиције, назив класе која се налази са друге стране асоцијације (*EType*), назив везе асоцијације (*Name*) и слично.

Слично наведеном примеру за везу асоцијације, за сваки од концепата EMF модела могу се специфицирати особине које су карактеристичне за тај концепт.



Слика 3.9 Едитирање особина за елемент модела

3.4.4 Парсирање текста

Xtext граматика која се креира у нотацији сличној EBNF се користи као улаз за парсер генератор и на основу ње се креира ANTLR парсер који представља основу за анализу текста.

Процес анализе текста се састоји из два дела, а то су лексичка анализа и парсирање. Лексичка анализа се састоји у креирању низа токена на основу улазног низа карактера. У ове токене спадају идентификатори, кључне речи, празни карактери (*whitespace)*, и друго.

У Xtext-у постоје уграђена лексичка правила, као што је правило ID, које је коришћено у примеру из претходног одељка и које представља идентификатор.

У процесу парсирања текста, парсер узима низ токена који је резултат лексичке анализе и на основу њега креира стабло парсирања (*parse tree*).

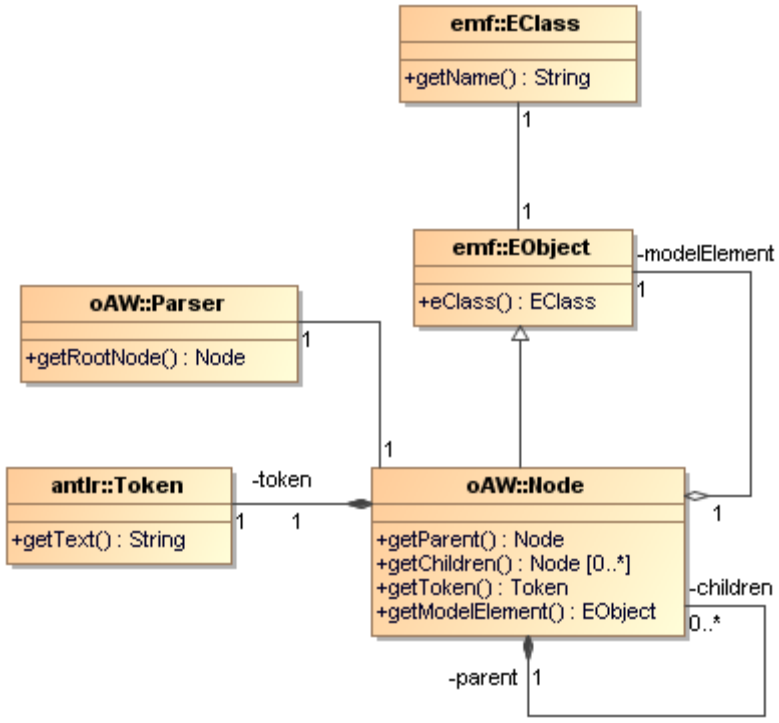
Правила за креирање типа у језику Xtext која су описана у предходном одељку уствари представљају правила парсирања, односно правила на основу којих се креира стабло парсирања.

На основу процеса парсирања генерише се меморијска структура текста која представља основу за даљу обраду текста у оквиру ситема. Опис ове меморијске структуре дат је у наредном одељку.

3.4.5 Меморијска структура текста

На слици 3.10 приказан је дијаграм класа који описује објектни модел који се добија као резултат парсирања текста у едитору који је генерисан на основу граматике. Овај објектни модел је универзални модел који се користи за било

коју граматику у оквиру усвојеног окружења и служи као основа за обраду текста у систему.



Слика 3.10 Објектни модел садржаја едитора

Модел текста који је приказан на слици 3.10 је исти за било коју граматику, али је он повезан за EMF моделом који је генерисан на основу граматике. Ова веза се састоји у томе да се помоћу операција класа посматраног модела могу добити називи и особине EMF модела на основу ког се уноси текст у едитору.

Свака класа на дијаграму са слике 3.10 представљена је својим именом, називом пакета и скупом операција које су релевантне за обраду меморијске структуре текста. Називи пакета укузују на део изабраног софтверског окружења коме припада класа.

Класом *Parser* представљен је парсер који врши анализу текста и креира стабло парсирања. Ова класа је део oAW софтеврског алата.

Стабло парсирања је доступно преко коренског чвора који се може преузети из парсера операцијом *getRootNode()*. Резултат ове операције је инстанца класе *Node* која је такође део алата oAW и представља корински елемент стабла парсирања.

Класом *Node* представљен је један чвор у стаблу парсирања. Ова класа садржи операције за кретање по структури стабла парсирања. То су операције *getChildren()* која враћа све чворове који су деца посматраног чвора и *getParent()* која враћа чвор који је родитељ посматраног чвора.

Класа *Node* је специјализација класе *EObject* из пакета који припада EMF окружењу. Једна инстанца класе *EObject* се јавља и као садржај класе *Node* и та инстанца је обележена као *modelElement*. Овом инстанцом класе *EObject* представљен је објекат класе EMF модела креираног на основу граматике. Посматрана веза агрегације између класе *Node* и класе *EObject* за сваки чвор у стаблу парсирања везује елемент EMF модела на који се тај чвор односи.

Класа *EObject* има операцију *eClass()* која враћа референцу на класу *EClass*. Класа *EClass* представља класу из EMF модела доменски специфичног језика. Резултат операције *getName()* је назив те класе.

Сваки чвор садржи и један токен представљен класом *Token* која је део пакета *antlr*. Инстанца ове класе је конкретан садржај, односно текст који припада једном чвору. Тај текст се може преузети операцијом *getText()*.

3.4.6 Језик Xtend

У оквиру Xtext окружења развијени се текстуални језици који се могу користити у разним аспектима спецификације језика и едитора. У основи свих тих језика је језик Xtend који је сличан језику за дефинисање ограничења над објектима (OCL – Object Constraint Language) [40]. Xtend се у литератури (Kolb и Voalter, 2008) описује као мешавина OCL-а и програмског језика Јава.

Xtend је *expression* језик који се користи за спецификацију великог броја проширења (*extensions*) EMF модела креираног доменски специфичног језика. Та проширења представљају операције EMF модела и формирају се помоћу Xtend израза. У спецификацији проширења могу се позивати и методе написане у програмском језику Јава.

Проширења могу да обухватају различите врсте функција над EMF моделом који представља текст написан у доменски специфичном језику. У те фукције спадају селекција појединих елемената модела, испитавање карактеристика елемената модела, навигација кроз модел, и слично. Поред тога, Xtend изрази се користе и за дефинисање понуде за допуну текста приликом едитирања у генерисаном едитору. Фајлови који садрже Xtend изразе имају екстензију *ext*.

На основу језика Xtend у оквиру Xtext окружења развијени су и језик за спецификацију ограничења над језиком – Check и језик за писање темплејта за трансформацију унетог текста – Xpand.

3.4.7 Изрази

Целокупна спецификације проширења и ограничења доменски специфичног језика заснива се на креирању израза (*expression*) у језику Xtend. Постоје једноставни изрази за приступ атрибутима класе EMF модела језика или позив операција, на пример:

`myTextElement.name`-израз за приступ атрибуту name неког елемента језика.

`myTextElement.doSomething()`– израз за позив проширења doSomething() које је дефисано за елемент *myTextElement*.

У језику Xtend могу се дефинисати и аритметички изрази, као и логички изрази чије вредности могу бити тачно (*true*) или нетачно (*false*). У изразима се могу користити стрингови за које постоји и велики број операција, а међу њима је и конкатенација стрингова помоћу оператора '+'.

Xtend језик има велику подршку за рад са колекцијама вредности, односно листама. Рад са листама у језику Xtend је реализован по угледу на стандардну библиотеку операција OCL-а. Операције над листама које су дефинисане у оквиру OCL-а описане су у [40] у одељку 11.9.

Над листама се у језику Xtend могу вршити следеће операције:

- селекције појединих елемената листе на основу критеријума:
  `collection.select(v|boolean-expression-with-v)`
- преузимање елемента листе на основу индекса:
  `collection.get(index)`
- провера да ли сви елементи листе задовољавају неки критеријум:
  `collection.forAll(v|boolean-expression-with-v)`
- провера да ли у листи постоји елемент који задовољава одређени:
  критеријум:
  `collection.exists(v|boolean-expression-with-v)`
- сортирање листе:
  `collection.sortBy(type|type.property)`

У креирању Xtend израза могу се користити клаузуле *if* и *switch* на исти начин као у програмском језику Јава.

Уколико треба позвати више израза у секвенци они се раздвајају знаком „`->`". Оваква секвенца израза се назива ланац израза (*Chain expression*) . Овај концепт је преузет из језика OCL.

Још један израз који је карактеристичан за језик OCL, а веома се користи за креирање Xtend израза је израз *let* којим се омогућава креирање локалних променљивих. Његова синтакса је:

```
let v = expression : expression-with-v
```

Оно по чему се Xtend израз *let* разликује од овог израза у OCL-у је то што се у Xtend-у тип креиране локалне променљиве аутоматски додељује на основу додељене вредности и не треба га експлицитно наводити.

Још један значајан концепт језика Xtend је преузет из Јаве и то је кастовање објеката чија синтакса је у потпуности иста као у Јави. Под појмом кастовање подразумева се додељивање класе објекту, односно стављање објекта у контекст класе на коју се односи.

3.4.8 Проширења (*extensions*)

Проширења (*extensions*) су специфични изрази написани у језику Xtend који представљају операције над доменски специфичним језиком и користе се у разним аспектима спецификације за унапређивање карактеристика доменски специфичног језика и едитора за тај језик.

Синтакса једноставног проширења је:

```
ReturnType extensionName(ParamType1 paramName1,
ParamType2...): expression-using-params;
```

Прво се наводи тип резултата проширења, затим назив проширења преко кога ће се ово проширење користити у оквиру спецификације, затим у загради листа улазних параметара. Након тога следи знак „:“ за почетак блока у ком се наводи израз који уз помоћ улазних параметара генерише резултат који је типа *ReturnType*.

Пример проширења који враћа стринг у ком је специфициран назив *get* методе креиран на основу имена прослеђеног елемената:

```
String getterName(NamedElement ele) :
'get'+ele.name.firstUpper();
```

Креирана проширења могу се позивати на два начина

- као функција – *getterName(myNamedElement)* или
- као позив методе објекта – *myNamedElement.getterName()*
  при чему се елемент *myNamedElement* мапира на први параметер проширења *getterName()*.

У проширењима се могу користити позиви метода написаних у Јави и то оних које су специфициране као јавне и статичне (*public static*). Проширења у

којима се позивају Јава методе се називају Јава проширења и синтакса ових проширења се може описати следећим уопштеним изразом:

```
Void myJavaExtension(String param) :
JAVA my.Type.staticMethod(java.lang.String);
```

3.4.9 Xtend проширења за допуну текста

У већини Eclipse едитора постоји механизам за добијање помоћи приликом уноса података. Та помоћ се састоји у томе да се кориснику у тренутку уноса притиском на тастер Ctrl+Space на екрану појави падајућа листа са понудама могућности које на том месту у тексту могу да се унесу. Као на пример приликом куцања Јава кода у Eclipse едитору, после унетог назива класе и тачке, генерише се листа свих *static* метода класе који могу да се позову у том делу кода.

Овај механизам се може реализовати и за едиторе који се генеришу у Xtext окружењу и то спецификацијом специјалних проширења у језику Xtend. Проширења за допуну текста се наводе у фајлу *ContentAssist.ext* који је Xtext генератором генерисан у оквиру пројекта за развој едитора за доменски специфичан језик (на пример. *my.dsl.editor*) (слика 3.5).

Спецификација допуне текста се ослања на Xtext граматику доменски специфичног језика која је описана у одељку 3.1. Листа могућих вредности за унос се може дефинисати за атрибуте у оквиру правила којима се дефинише тип. У примеру из одељка 3.1 дефинисан је тип *Entity* који је садржао атрибут *name,* односно назив ентитета, за који се може специфицирати листа могућих вредности.

Проширење за спецификацију листе за допуну текста се наводи у облику:

```
List[Proposal] complete<nazivTipa>_<nazivAtributa>
(emf::Eobject ctx, String prefix):<list_of_Proposals>;
```

Где објекат *ctx* представља објекат типа дефинисаног у граматици и то онај објекат на ком се у едитору налази курсор у тренутку позива помоћи за допуну текста. То је објекат који је „на реду“ да се унесе према специфицираној граматици. Вредност параметра *prefix* је текст који је за посматрани атрибут унет пре позива помоћи за допуну текста.

После заглавља проширења, односно после знака „:“ наводи се листа понуда за допуну текста. Једна понуда се састоји од лабеле која представља текст који се приказује кориснику у падајућој листи, иконице која се испред те лабеле приказује у падајућој листи и текста који се преноси у едитор ако је изабрана дата понуда. Појединачна понуда се креира проширењем *newProposal* које се може позвати са различитим параметрима у којима се специфицирају неки од

или сва три наведена елемента који чине понуду. Постоје три могућности за креирање понуде:

- *newProposal(String proposal)* – параметар *proposal* ће бити и лабела и текст који се преноси, нема иконице,
- *newProposal(String label, String toInsert)* – лабела се наводи као први параметер, а текст који се преноси као други параметар, ни у овом случају неће бити иконице у падајућој листи,
- *newProposal(String label, String toInsert, String image)* – као трећи параметар се наводи релативна путања до слике која ће се користити као иконица.

За назив ентитета у примера из одељка 3.1 могу се на следећи начин навести понуде за допуну текста:

```
List[Proposal] completeEntity_name(emf::EObject ctx,
String prefix) :
{newProposal("Osoba"),
 newProposal("Radnik"),
 newProposal("Vlasnik")};
```

3.4.10 Спецификација ограничења над доменски специфичним језиком

Један од веома важних аспеката едитирања података је провера исправности унетих података и пријављивања грешака кориснику. Софтверски алат Xtext омогућава да се у генерисином едитору специфицирају ограничења над EMF моделом доменски специфичног језика и да се онда на основу те спецефикиције врши провера уноса у реалном времену и пријављивање грешака и упозорења.

Ограничења (*constraints*) се у Xtext-у специфицирају у језику Check који је заснован на језику Xtend и веома је сличан OCL-у. Ограничења се специфицирају у фајлу Check.chk који је генерисан Xtext генератором у оквиру пројекта у ком се развија доменски специфичан језик (на пример *my.dsl*) (слика 3.6).

И спецификација ограничења са ослања на EMF модел који је генерисан на основу граматике језика која је описана у одељку 3.1 и може се односити на разне аспекте, као што су дужина унетог податка, формат унетог податка, број појављивања неког елемента језика, зависност једног елемента језика од неког другог, итд. Приликом спецификације ограничења наводи се у којим условима је унети текст погрешан, као и текст који описује насталу грешку.

На пример, ограничиње на назив ентитета (из примера наведеног у поглављу 3.1) на дужину већу од три карактера би се у Check језику специфицирало на следећи начин:

```
context Entity ERROR "Naziv mora biti dužine vece od 3
karaktera" : this.name.length>3;
```

Овим је специфицирано да се у контексту *Entity* пријављује грешка са текстом који је наведен под знацима навода у случају да није задовољен критеријум *this.name.length>3*. Израз *this* у језику Check и Xtend има исто значење као исти израз у програмском језику Јава, а одговарајући израз у OCL-у је *self* и односи се на инстанцу класе у чијем контексту је позван.

3.4.11 Темплејти за трансформацију

У оквиру Xtext окружења постоји подршка за креирање темплејта којима се текст написан у специфицираном језику може трансформисати у друге облике који се исписују у посебне фајлове. На пример, пошто се Xtext најчешће користи за креирање метамодела, а креирани доменски специфични језици се користе за писање модела у неком домену, веома су корисни темплејти који тако написане моделе трансформишу у класе неког програмског језика.

Ови темплејти се пишу у језику Xpand у коме се користе елементи језика Xtend за селекцију делова унетог садржаја. Фајлови који садрже темплејте за трансформацију имају екстензију *xpt* и пишу се у оквиру пројекта са наставком *generator* (*my.dsl.generator*).

Један пример темплејта за трансформацију који се односи на пример са ентитетом и именом ентитета је:

```
«IMPORT meta::model»
«EXTENSION my::ExtensionFile»
«DEFINE javaClass FOR Entity»
«FILE fileName()»
package «javaPackage()»;
public class «name» {
// implementation
}
«ENDFILE»
«ENDDEFINE»
```

Овим темплејтом се за ентитет генерише Јава класа чије име је име ентитета, односно унета вредност атрибута *name* типа *Entity*.

Темплејт написан у језику Xpand се састоји од произвољног броја *IMPORT* израза, иза кога следи произвољан број *EXTENSION* израза и затим један или више *DEFINE* блокова у којима се специфицира конкретан алгоритам трансформације.

Темплејти за трансформацију се покрећу креирањем токова послова (*workflow*) који представљају XML конфигурационе фајлове. У овим

конфигурационим фајловима се специфицирају параметри неопходни за покретање темплејта, као што су назив фајла који се трансформише, назив фајла који ће садржати трансформисани текст и слично.

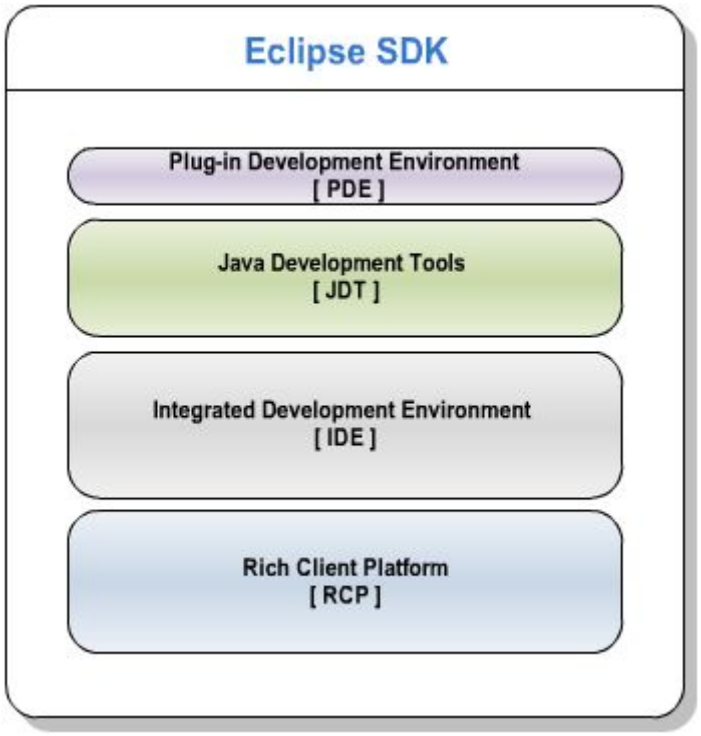
3.5 ECLIPSE PLUG-IN ТЕХНОЛОГИЈА

Eclipse није једна велика Јава апликација, већ се може рећи да је то мала Јава апликација која представља окружење које се може „напунити“ другим Јава програмима, односно plug-in-овима [28]. Plug-in је један део Eclipse платформе који се може развијати одвојено. Целокупну архитектуру Eclipse-а чине plug-in-ови плус Platform Runtime, мали програм који представља језгро платформе.

Језгро Eclipse-а је окружено стотинама и хиљадама plug-in-ова. Plug-in није ништа друго него Јава програм који на неки начин проширује функционалност Eclipse-а. Сваки Eclipse plug-in може или да користи сервисе других plig-in-ова или да прошири њихову функционалност да би се као такав користио од стране других plug-in-ова.

Eclipse је отворена платформа дизајнирана тако да се може једноставно проширити (*extend*) додатним функционалностима. У језгру ове платформе налази се Eclipse SDK (Software Development Kit) око кога се могу креирати различити софтверски производи и алати, који се даље могу проширивати у оквиру Eclipse-а. Архитектура Eclipse-а је нарочито погодна за проширивање, а проширивост се постиже коришћењем plug-in технологије.

На слици 3.11 приказана је архитектура Eclipse-овог SDK-а. На дну ове архитектуре налази се RCP – Rich Client Platform, који обезбеђује архитектуру и окружење за креирање RCP апликација. Следећи ниво је IDE – Integrated Development Environment који представља платформу за развој софтверских алата, а сам по себи је једна RCP апликација. Коришћењем овог нивоа могу се развијати различити софтверски алати као што је на пример алат за рад са базама података. На основу IDE платформе развијена је Јава IDE платформа представљена као ниво JDT - Java Development Tools. Овај ниво обезбеђује окружење за равој софтвера у програмском језику Јава. Њиме се Eclipse-у додају прозори, едитори, визарди, билдери који обезбеђују окружење за развој Јава програма. На овај ниво архитектуре SDK-а се наслања ниво који садржи софтверске алате за развој plug-in-ова и RCP апликација – PDE – Plug-in Development Environment.



Слика 3.11 Архитектура Eclipse SDK-а - преузето из [28]

Сви поменути слојеви SDK архитектуре састављени су од plug-in-ова, односно имплементирани су као скуп међусобно повезаних plug-in-ова.

У наставку ће бити описани основни концепти plug-in технологије на основу (Clayberg and Rubel, 2008).

3.5.1 Основне карактеристике Eclipse plug-in технологије

Цела технологија plug-in-ова заснива се на проширивости Eclipse платформе. Да би се постигла ова проширивост користи се концепт тачака проширења (*extension points*) и проширења (*extensions*). Сваки plug-in дефинише своје тачке проширења које други plug-in-ови могу да проширују. Plug-in који проширује не зна ништа о начину реализације plug-in-а који се проширује. Важно је напоменути да је свака тачка проширења нека врста уговора и прихватају се само она проширења која поштују тај уговор.

Plug-in-ови се још могу окарактерисати као екстерни софтверски алати који обезбеђују додатну функционалност платформе и потпуно су интегрисани са њом. У већини случајева plug-in-ови имплементирају неку специјализовану функционалност која није већ подржана платформом.

Plug-in-ови се имплементирају у програмском језику Јава и развијају се коришћењем Plug-in Development Environment – PDE [41] технологије која је део Eclipse SDK (слика 3.11).

Понашање сваког plug-in-а имплементирано је у програмском коду, али све особине plug-in-а као што су зависности (*dependencies*) и сервиси које обављају (*services*) декларишу се у фајловима *MANIFEST.MF* и *plugin.xml* који су део plug-in пројекта. Поред ова два фајла plug-in садржи и скуп Јава класа којима је имплементирана функционалност plug-in-а, као и иконице, слике и остале статичке елементе који се користе у имплементацији plug-in-а.

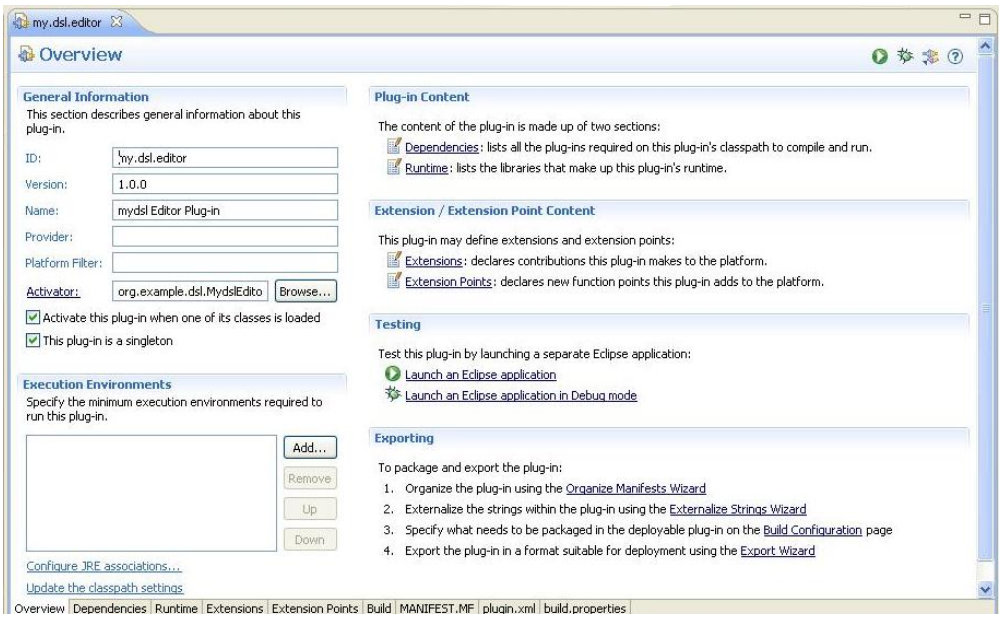
Фајл *MANIFEST.MF* описује plug-in тако што садржи назив, ID, и верзију plug-in-а на који се односи, листу осталих plug-in-ова који су неопходни да би се plug-in извршавао и локацију на којој се налази програмски код којим је реализован plug-in. Фајл *MANIFEST.MF* може се едитирати коришћењем Eclipse-ових алата.

Фајл *plugin.xml* описује проширења која су реализована plug-in-ом као и тачке проширења дефинисане plug-in-ом.

3.5.2 Plug-in Manifest едитор

Eclipse платформа нуди веома ефикасно окружење за развој plug-in-ова чији део је и посебан едитор за спецификацију plug-in-ова, то је Plug-in Manifest едитор. Захваљујући овом едитору развој plug-in-ова је веома олакшан и углавном се своди на спецификацију у окружењу које се може окарактерисати као кориснички-пријатељско (*user friendly*). Двокликом на један од фајлова *MANIFEST.MF* или *plugin.xml* у Eclipse-у отвора се овај едитор. Едитор се састоји од девет картица којима се специфицирају подаци о plug-in-у чији је *MANIFEST.MF* или *plugin.xml* фајл отворен. Картице Plug-in manifest едитора нуде кориснички погодније окружење за едитирање ова два фајла, *MANIFEST.MF* и *plugin.xml*.

На слици 3.12 приказан је plug-in manifest едитор за plug-in *my.dsl.editor*. У овом едитору отворена је прва картица обележена лабелом *Overview*. Са ове картице могу се прочитати и едитирати основни подаци о plug-in-у и специфицирати подаци за експорт plug-in-а. Поред тога, са ове картице може се тестирати plug-in, тако што ће се покренути посебна Eclipse апликацији која ће у скупу plug-in-ова садржати и посматрани plug-in. Подаци који се уносе у овој картици уписују се у фајл *MANIFEST.MF*.

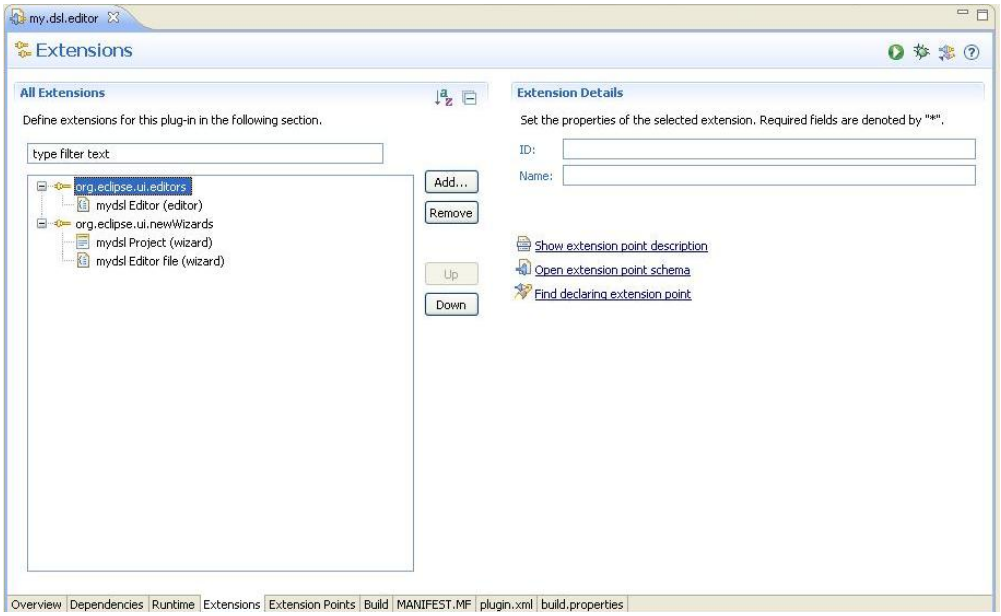


Слика 3.12 Plug-in Manifest едитор – картица Owerview

Још једна картица у Plug-in manifest едитору је веома значајна за развој plug-in-a и коришћена је у развоју система за каталогизацију, то је картица *Extensions*. Ова картица представља графички приказ фајла plugin.xml и њој се специфицирају проширења plug-in-a. Подаци о проширењима се у овој картици уносе без познавања xml-a и синтаксе фајла plugin.xml.

На слици 3.13 приказан је Plug-in manifest едитор са отвореном картицом *Extensions* за plug-in *my.dsl.editor*. Ова картица се састоји из два дела, лева страна форме садржи стабло свих проширења која су дефинисана за plug-in, а десна спецификацију појединачног проширења који је селектован у стаблу са леве стране.

Plug-in manifest едитор обезбеђује кориснику велики број пречица и помоћи приликом едитирања података. Тако на пример, десним кликом на назив тачке проширења (лева страна форме на слици 3.13) отвара се падајући мени са акцијама којима се може креирати ново проширење. Поред тога, десна страна форме са слике 3.13 у којој се уноси спецификација проширења омогућава кориснику кретање кроз пројекат plug-in-a, приказ класа и иконица, опис структуре података који треба да се унесе у текстуална поља, итд.



Слика 3.13 Plug-in manifest едитор – картица Extensions

Реализација сваке додатне функционалности plug-in-а започиње спецификацијом проширења неког од концепата Eclipse-а, а та спецификација врши се у картици *Extension* Plug-in manifest едитора. Захваљујући свим погодностима које пружа овај едитор, велики део реализације тих функционалности састоји од спецификације, док је имплементација сведена на минимум. Поред тога, део саме имплементације ових функционалности реализован је у самом Eclipse окружењу, што представља још једну од погодности предложеног приступа реализације софтверских система.

3.5.3 Тачке проширења

Eclipse окружење нуди могућност ефикасног проширења функционалности, који није својствено само платформи већ и plug-in-овима који су њен саставни део. Сваки plug-in може да дефинише сопствене тачке проширења (*extension points*). Тачке проширења plug-in-а могу се користити интерно, у самом plug-in-у чиме се добија виши степен модуларности система или екстерно од стране неког другог plug-in-а (Clayberg and Rubel, 2008).

Тачке проширења доприносе флексибилности и прилагодљивости система, чиме је омогућена лакша кастомизација, односно лакше прилагођевање система специфичним захтевима. Циљ коришћења тачака проширења је да се омогући кориснику софтвера да прошири софтвер додатним функционалностима које се нису могле предвидети приликом самог развоја.

На концепту тачака проширења заснована је и цела Eclipse платформа. Тачкама проширења реализује се „слабо повезивање“ (*loosely coupling*) модула. Један plug-in дефинише своје тачке проширења у свом манифесту тако што наводи минимални скуп интерфејса и њима придружених класа које други могу да користе. Затим други plug-in-ови дефинишу проширења за ове тачке проширења тако што имплементирају одговарајуће интерфејсе или апстрактне класе.

Свака тачка проширења има свој јединствени идентификатор који се састоји од идентификатора plug-in-а коме припада и локалног идентификатора који је јединствен у оквиру plug-in-а коме припада. За сваку тачку проширења специфицира се XML шема која описује начин коришћења те тачке проширења. Шема је смештена у фајлу са екстензијом *exsd* а назив фајла је локални идентификатор plug-in-а.

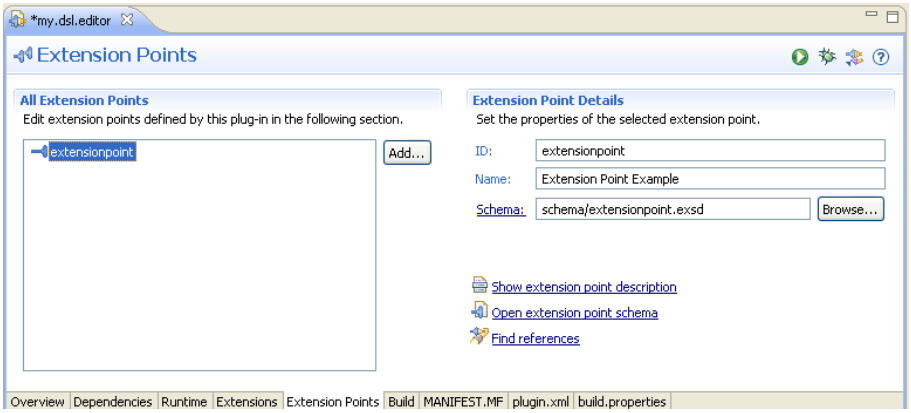
Спецификација тачака проширења је садржај фајла plugin.xml у ком је, поред основних података, наводена и референца на XML шему у којој је описан начин коришћења тачке проширења.

Спецификација тачака проширења може се вршити директно у фајлу plugin.xml, а шема се може едитирати у било ком XML едитору. Међутим, у оквиру Plug-in Manifest едитора постоје екранске форме које нуде графичко окружење за ефикасну спецификацију тачака проширења.

3.5.3.1 Спецификација тачака проширења у Plug-in Manifest едитору

Једна од картица на екранској форми Plug-in Manifest едитора (слика 3.12) има лабелу *Extension Points* и у њој се врши спецификација тачака проширења plug-in-а. На слици 3.14 приказан је изглед ове картице са креираном једном тачком проширења чији је локални идентификатор *extensionpoint*. Ова тачка проширења креирана је за plug-in *my.dsl.editor*

У екранској форми која је приказана на слици 3.14 специфицирају се само основни подаци о тачки проширења, као што су назив, идентификатор и путања до шеме у којој је детаљније специфицирана ова тачка проширења. Сви остали подаци који се односе на начин коришћења ове тачке проширења специфицирају се у XML шеми проширења која је у овом случају наведена у фајлу *extensionpoint.exsd*.



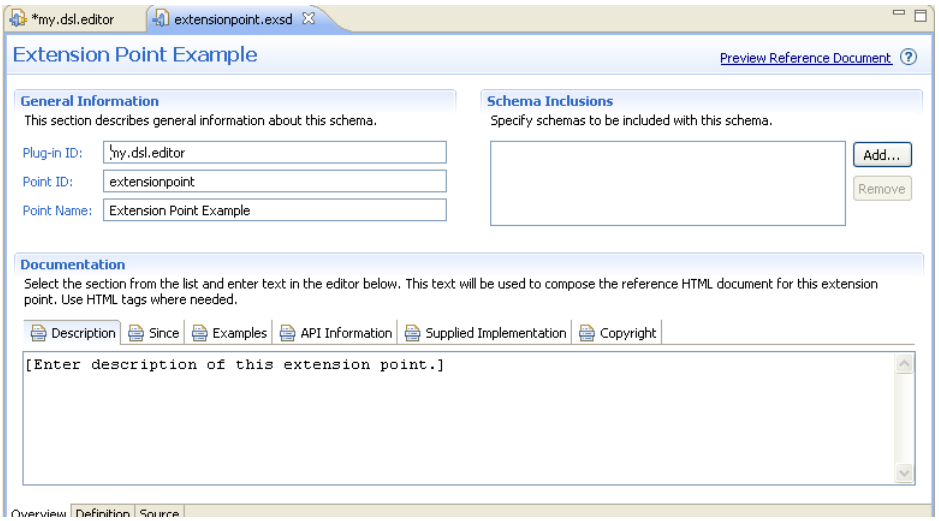
Слика 3.14 Спецификација тачака проширења у Plug-in Manifest едитору

3.5.3.2 Креирање шеме за тачку проширења

Спецификација шеме за тачку проширења у оквиру Eclipse окружења врши се у посебном едитору који је сличан Plug-in Manifest едитору. Изглед овог едитора за шему *extensionpoint.exsd* приказан је на слици 3.15.

Едитор за шему тачке проширења има три картице. У картици *Overview* наводе се основни подаци о тачки проширења, опис тачке проширења у виду документације и списак шема које се укључују у спецификацију ове шеме. На слици 3.15 приказан је изглед ове картице. Картица *Definition* садржи спецификацију елемената тачке проширења на основу којих ће се имплементирати конкретна проширења. Картица *Source* садржи опис шеме у изворном, XML формату и њен садржај је генересан на основу садржаја прве две картице, али се и у њој може вршити едитирање.

Суштински део спецификације тачке проширења односи се на дефинисање елеманата и атрибута који ће представљати модел за креирање проширења посматране тачке проширења. Спецификацијом елемената шеме и њихових атрибута дефиниши су параметри који ће се уносити приликом проширења.



Слика 3.15 Спецификација шеме за тачку проширења

Значење елемената и атрибута шеме тачке проширења у наставку је објашњено на примеру једног специфицираног проширења. Пример проширења које је специфицирано за plug-in *my.dsl.editor* је:

```
<extension
         point="org.eclipse.ui.editors">
      <editor
            class="org.example.dsl.editor.MydslEditor"

contributorClass="org.eclipse.ui.texteditor.BasicTextEditorAc
tionContributor"
            extensions="dsl"
            icon="icons/file.gif"
            id="org.example.dsl.editor.MydslEditor"
            name="mydsl Editor">
      </editor>
 </extension>
```

Ово проширење дефинисано је за тачку проширења чији је идентификатор вредност атрибута *point* елемента *extension*, а то је *org.eclipse.ui.editors*. Елемент *extension* је коренски елемент сваког проширења и он има један обавезни атрибут – *point*. Остатак елемента *extension* зависи од спецификације тачке проширења *org.eclipse.ui.editors*.

На основу приказаног листинга проширења може се закључити да тачка проширења дефинише елемент *editor* који има неколико атрибута. Имена тих атрибута су *class, contributorClass, extensions, icon, id* и *name*. За сваки атрибут елемената тачке проширења специфицира се тип. Тако су на пример атрибути *class* и *contributorClass* типа *java* који представља класу написану у

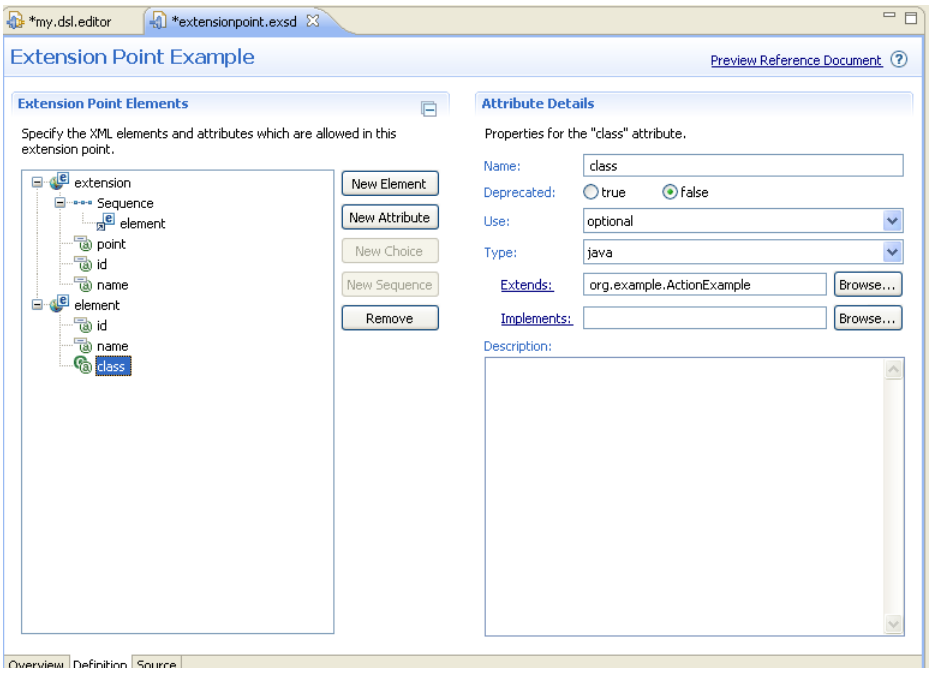
програмском језику Јава. Атрибут *icon* је типа *resource* којим се представљају ресурси, а остали атрибути су типа *string*.

Поред имена и типа, за атрибут елемента тачке проширења могу се дефинисати још неки параметри, а скуп параметара који се дефинишу зависи од изабраног типа. За атрибуте било ког типа специфицира се да ли су обавезни или опциони и ако су обавезни може им се доделити подразумевана (*default*) вредност. За сваки атрибут се специфицира да ли се не саветује његова употреба, односно да ли има особину *deprecated*. Сваком атрибуту се може доделити одговарајући текстуални опис. Атрибутима који су типа *java* додељује се назив класе коју класа која је вредност овог атрибута мора да наследи или назив интерфејса који мора да имплементира. За атрибуте који су типа *string* дефинише се да ли вредност атрибута треба да се преводи, односно да ли је његова вредност намењена човеку (*human readable*) а може се специфицирати и рестрикција вредности на неки дозвољени скуп.

Спецификација елемената тачке проширења и њихових атрибута едитира се у картици *Definition* у едитору за шему тачке проширења. На слици 3.16 приказан је изглед картице *Definition* у којој се специфицирају елементи за тачку проширења *extensionpoint* plug-in-a *my.dsl.editor*.

На левој страни екранске форме приказане на слици 3.16 наведени су сви елементи и атрибути дефинисани за посматрану тачку проширења. Коренски елемент је елемент *extension* и он има дефинисана три атрибута, то су атрибути *point*, *id,* и *name*. Елемент *extension* садржи и секвенцу елемената *element*. Кардиналитет појављивања елемента *element* у елементу *extension* је *1* што значи да се у проширењу дефинише тачно један елемент *element*.

Елементом *element* дефинисано је окружење за спецификацију проширења ове тачке проширења. Овај елемент има три атрибута, два опциона, *id* и *name* који су типа *string*. Елемент *element* има и један обавезан атрибут, то је атрибут *class*. На левој страни екранске форме са слике 3.16 селектован је овај атрибут и његова спецификација је приказана на десној страни екранске форме. Поред имена атрибута, наведено је и да атрибут нема особину *deprecated*, да је његова употреба обавезна и да је његов тип *java*. Пошто је посматрани атрибут типа *java* може се изабрати класа која треба да наследи класу која је садржај овог атрибута или интерфејс који треба да имплементира. Пун назив класе наводи се као вредност параметра *Extends*, а пун назив интерфејса као садржај параметра *Implements*. У примеру са слике 3.16 наведено је да класа која је вредност атрибута *class* елемента *element* мора да наследи класу *org.example.ActionExample*.



Слика 3.16 Спецификација елемената тачке проширења

Овако креирана тачка проширења може се проширити од стране неког другог plig-in-a. Пример спецификације проширења за тачку проширења *extensionpoint* је.

```
<extension
      point="my.dsl.editor.extensionpoint">
  <element
      id="newAction"
      name="My Action "
      class="org.newexample.MyAction ">
  </element>
</extension>
```

Садржај атрибута *point* елемента *extension* је цео идентификатор тачке проширења који се састоји од имена plug-in-a (*my.dsl.editor*), тачке и локалног идентификатора тачке проширења (*extensionpoint*). У елементу *element* специфицирани су сви параметри проширења, назив проширења, идентификатор проширења и класа која наслеђује класу *org.example.ActionExample*.

3.5.3.3 Програмски код као подршка за тачку проширења

Кад је направљена тачка проширења, потребно је имплементирати програмски код који узима у обзир постојање могућих проширења.

Програмски код којим се преузимају сва проширења тачке проширења *extensionpoint* дат је у наставку.

```
IExtension[] extensions =
Platform.getExtensionRegistry()
.getExtensionPoint(Plugin.PLUGIN_ID, "extensionpoint")
.getExtensions();
```

Најпре се преузме регистар проширења из платформе, затим се из тог регистра преузме тачка проширења одговарајућег имена и на крају се за ту тачку проширења преузму сва доступна проширења.

Да би се добила инстанца класе која је садржај атрибута *class* елемента *element*, прво треба преузети конфигурационе елементе (*configuration elements*) проширења. Затим узети конфигурациони елемент са именом *element* и направити инстанцу класе која је садржај атрибута *class*. Креирање одговарајуће инстанце класе за тачку проширења *extensionpoint* реализује се следећим фрагментом програмског кода:

```
ActionExample action = (ActionExample)confElements[i].
    createExecutableExtension("class");
```

Операцијом *createExecutableExtension* креира се инстанца одговарајуће класе и смешта у објекат *action*.

3.6 КОМПОНЕНТЕ КОРИСНИЧКОГ ИНТЕРФЕЈСА ECLIPSE-А

Компоненте корисничког интерфејса Eclipse-а развијају се у SWT – Standard Widget Toolkit [29] технологији. SWT технологија омогућава ефикасан развој савременог корисничког интерфејса. Основна карактеристика ове технологије је да она нуди директан приступ ресурсима оперативног система на ком се извршава. SWT користи концепте оперативног система за креирање корисничког интерфејса. У оквиру SWT технологије доступне су све компоненте за креирање савремених корисничких интерфејса као што су стабла (*tree*), табеле (*table*), лабеле (*label*), дугмад (*button*) и слично.

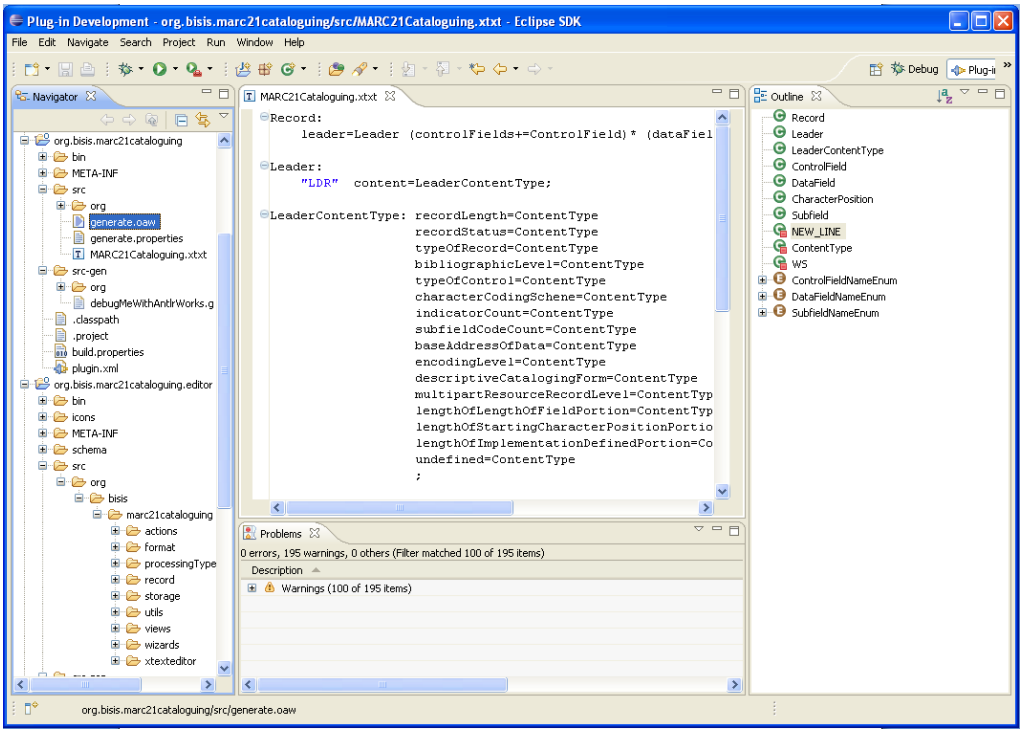
Као додатак SWT технологији, за развој екранских форми Eclipse-а користи се и JFace [30]. JFace се заснива на SWT-у али нуди неке сложеније концепте за развој корисничког интерфејса. Поред тога, JFace уводи концепт повезивања података за графичку компоненту по принципу креирања модела података. Особина JFace технологије је да нуди велики број већ готових компоненти корисничког интерфејса чиме се смањује број линија програмског кода неопходних да се развије сложен кориснички интерфејс апликације.

Основне компоненте корисничког интерфејса Eclipse-а су:

- Погледи (*Views*),
- Едитори (*Editors*),

- Акције (*Actions*),
- Дијалози и визарди (*Dialogs and Wizards*),
- Перспективе (*Perspectives*).

Сваки Eclipse прозор се састоји од једног или више погледа или едитора. Едитори се користе за едитирање ресурса (фајлова), док се погледи користе да би пружили помоћ едиторима. Погледи најчешће служе за хијерархијски приказ података, за отварање едитора, приказ особина активног едитора, структуирани поглед на садржај едитора или на грешке настале приликом едитирања. На слици 3.17 приказан је Eclipse прозор у ком је отворен један едитор (централно место на екранској форми) и три погледа (*Navigator*, *Problems* и *Outline*).



Слика 3.17 Екранска форма Eclipse-а

Акције представљају акције корисника као што су акција за снимање фајла, штампање, покретање апликације и слично. Акције се у оквиру Eclipse корисничког интерфејса могу позвати преко дугмета на главном *toolbar*-у апликације, позивом ставки у програмском менију, преко дугмади у *toolbar*-у сваког погледа и позивом ставки у падајућем менију едитора.

Дијалози су помоћни прозори који су део сваког корисничког интерфејса. Они се могу користити за прослеђивање обавештења кориснику или за преузимање

неког уноса корисника. Специјалан случај дијалога су визарди који се обично састоје од неколико страница којима се реализује нека сложенија функционалност. Један од примера визарда у Eclipse-у је креирање новог пројекта, где се прво бира врста пројекта, затим уноси назив пројекта, а као резултат визарда добија се нови пројекат на основу унетих параметара.

Перспектива се односи на скуп компоненти корисничког интерфејса и њихову организацију у оквиру прозора. Једном перспективом специфицира се скуп погледа и едитор, као и њихова организација на екрану (који поглед ће се налазити лево, који доле, и слично). Перспектива може специфицирати и скуп акција које ће бити доступне на *toolbar*-у и у главном менију апликације.

Унапређење система за каталогизацију које се састоји у проширењу основног едитора додатним функционалностима реализовано је коришћењем поменутих компоненти Eclipse корисничког интерфејса.

Поглавље 4

Библиографски формати

Најпознатији формат за размену библиографских података у машински читљивом облику је MARC [10] (*Machine Readable Cataloging*) развијен пре више од 30 година у Конгресној библиотеци (*Library of Congress*) [11]. На основу овог формата, за потребе различитих држава и библиотечких система развили су се други библиографски формати, на пример: USMARC у Америци, CAN/MARC у Канади, UKMARC у Великој Британији, UNIMARC [13] (*Universal Machine Readable Cataloguing)* у европским државама. Године 1999. настао је формат MARC 21 [12] на основу формата USMARC и CAN/MARC.

У раним фазама развоја XML технологије уочена је повезаност структуре библиографских записа са структуром XML документа. Особина која повезују ове две структуре јесте постојање хијерхијског односа између елемената, елемената који могу бити обавезни или опциони, који се могу понављати или делити на поделементе. Свака XML структура може се формално описати XML шемом која је дефинисана XMLSchema језиком [40]. Правила за креирање XML шеме одговарају правилима за креирање структуре MARC записа. Највећа предност коришћења XML-а за представљање библиографских података огледа се у једноставнијој обради података у отвореним дистрибуираним системима. Томе доприносе и већ развијени софтверски алати за рад са XML документима. Поред велике предности у области описивања MARC записа, уочено је да XML технологије пружају погодно окружење за опис података који се односе на сам формат.

У овом одељку описани су библиографски формати MARC 21 и UNIMARC и формирана је XML шема и објектни модел ових формата. Даље је дат преглед постојећих XML шема за MARC 21 и UNIMARC запис као и објектни модел библиографског записа ових формата.

На основу постојеће XML шеме MARC 21 записа у овом поглављу дат је предлог објектног модела MARC 21 записа који се користи за спецификацију

Xtext граматике на основу које се генерише едитор за MARC 21 записе и ово је описано у поглављу 5.

XML шема MARC 21 формата предложена у овом поглављу и објектни модел креиран на основу ове шеме коришћени су за приказ библиографског формата у систему за каталогизацију и ово је описано у поглављу 6.

4.1 MARC 21 ФОРМАТ

Запис по MARC стандарду је композиција три елемента: структуре записа, ознаке садржаја и података које запис садржи. Структура записа је имплементација Америчког националног стандарда за размену информација (*American National Standard for Information Interchange*) (ANSI/NISO Z39.2) [43], односно његовог еквивалента по ISO стандарду ISO 2709 [44]. Под ознаком садржаја подразумевају се називи поља, потпоља, шифарници и остале конвенције уведене како би се подаци о публикацијама на јединствен начин могли окарактерисати и идентификовати. Наведени елемент записа прописан је посебно сваким MARC форматом. Правила за креирање оног елемента записа који се односи на податке о публикацијама прописани су стандардима ван формата и везана су за установу која врши каталогизацију. Неки од најпознатијих међународних стандарда из ове области су: *International Standard Bibliographic Description* (ISBD) [9], *Anglo-American Cataloguing Rules*, 2nd edition (AACR 2) [45], *Library of Congress Subject Headings* (LCSH), *Holdings Statements Summary Level* (ISO 10324), *American National Standard for Serial Holdings Statements* (ANSI/NISO Z39.44), *Library of Congress Classification* (LCC).

Стандард MARC 21 састоји се од пет документа:

- MARC 21 формат за библиографске податке (*MARC 21 Format for Bibliographic data*) [46],
- MARC 21 формат за нормативне податке (*MARC 21 Format for Authority Data*) [47],
- MARC21 формат за податке о фонду односно локацијске податке (*MARC 21 Format for Holdings Data*) [48],
- MARC 21 формат за класификационе податке (*MARC 21 Format for Classification*) [49],
- MARC 21 формат за информације о заједници (*MARC 21 Format for Community Information*) [50].

Структура MARC записа састоји се из три основна дела, заглавља записа (*Leader*), директоријума (*Directory*) и скупа варијабилних поља (*Variable*

Fields). У наставку ће бити дато кратко објашњење сваког од наведених делова MARC записа.

Заглавље записа јавља се као први елемент записа и фиксне је величине од 24 карактера. Сваки податак у заглављу записа је шифрирана вредност на одговарајућој позицији карактера и носи информацију неопходну за обраду записа.

Директоријум садржи податке о ознаци, дужини и почетној локацији за свако варијабилно поље које се јавља у оквиру записа. Подаци о једном пољу у директоријуму су фиксне дужине од 12 карактера. На почетку директоријума налазе се подаци о контролним пољима записа, поређани по ознаци поља у растућем редоследу. Следе подаци о пољима поређани у растућем редоследу према првом карактеру ознаке поља. Редослед појављивања поља података у запису не мора одговарати редоследу појављивања одговарајућих података у директоријуму. Поља која се више пута понављају у запису у директоријуму се разликују по локацији на којој се јављају у запису. Део записа који се односи на директоријум завршава се карактером који означава крај поља (ASCII 1E hex).

Подаци су у MARC 21 запису организовани у **варијабилна поља** која се идентификују ознаком која садржи три цифре. Поља су у запису одвојена посебним карактером, ASCII 1E hex. MARC запис се завршава ознаком за крај записа, ASCII 1D hex. Разликују се две врста варијабилних поља:

- **контролна поља** (*control field*),
- **поља** (*data field*).

У контролна поља спадају поља која немају индикаторе нити потпоља, могу садржати или јединствен податак о публикацији или скуп података фиксне дужине при чему се сваки појединачни податак јавља на тачно одређеној позицији. Поља садрже индикаторе и потпоља.

У делу записа који садржи податке везане за једно поље резервисана су и два карактера за први и други **индикатор** поља. Улога индикатора је да омогуће прецизнију интерпретацију или ближе одреде податке који се јављају као садржај припадајућег поља. Вредности индикатора интерпретирају се одвојено, односно два индикатора у пољу немају међусобне везе. Вредности индикатора могу бити мала алфабетска слова или нумерички симболи. Бланко карактер (ASCII 20 hex) на месту индикатора означава да индикатор за дато поље није дефинисан, овај случај индикатора често се обележава и знаком #.

За ознаку **потпоља** у запису резервисана су два карактера, први је ознака за долар ($), а други ознака потпоља која може бити мало алфабетско слово или

нумерички симбол. Ознаке потпоља уведене су у сврху јединствене идентификације података. Редослед појављивања потпоља специфицира се правилима каталогизације. Потпоља у оквиру записа представљају основне носиоце података о публикацији на коју се запис односи. По MARC 21 стандарду, интерпункција која подржава стандард за библиографски (каталошки) опис различитих типова публикација укључује се у садржај потпоља записа.

По MARC стандарду поља и контролна поља могу имати особину поновљивости у оквиру записа. Такође и потпоља у оквиру поља могу бити поновљива.

На листингу 4.1 дат је пример библиографског записа по MARC 21 формату. Овај формат приказа записа усвојен је као формат за унос податка у едитору који је описан у поглављу 5.

```
LDR  01142cam**2200301*a*4500
001    92005291
003    DLC
005    19930521155141.9
008    920219s1993    caua   j      000 0 eng
010 ## $a92005291
020 ## $a0152038655 :$c$15.95
040 ## $aDLC$cDLC$dDLC
042 ## $alcac
050 00 $aPS3537.A618$bA88 1993
082 00 $220$a811/.52
100 1# $aSandburg, Carl,$d1878-1967.
245 10 $aArithmetic /$cCarl Sandburg ; illustrated as an anamorphic adventure by Ted Rand.
250 ## $a1st ed.
260 ## $aSan Diego :$bHarcourt Brace Jovanovich,$cc1993.
300 ## $a1 v. (unpaged) :$bill. (some col.) ;$c26 cm.
500 ## $aOne Mylar sheet included in pocket.
520 ## $aA poem about numbers and their characteristics. Features anamorphic, or distorted, drawings which can be restored to normal by viewing from a particular angle or by viewing the image's reflection in the provided Mylar cone.
650 #0 $aArithmetic$xJuvenile poetry.
650 #0 $aChildren's poetry, American.
650 #1 $aArithmetic$xPoetry.
650 #1 $aAmerican poetry.
650 #1 $aVisual perception.
700 1# $aRand, Ted,$eill.
```

Листинг 4.1 Пример записа по MARC 21 стандарду

Први елемент записа је заглавље записа, садржај заглавља записа наведен је иза ознаке *LDR*. Првих пет карактера у заглављу записа резервисано је за логичку дужину записа, док се на шестој позицији налази шифра којом је одређен статус записа. У посматраном примеру шифра статуса записа је *c* што значи да се ради о исправљеном или редигованом запису. На седмој позицији налази се шифра *a* која означава да је запис креиран за текстуалну грађу. Поред текстуалне грађе, на овој позицији дозвољене су и шифре за нотну, картографску грађу, музички запис, компјутерску датотеку и друго. Следећа позиција карактера означава библиографски ниво и у посматраном примеру на овој позицији налази се шифра *m* која означава да се ради о монографској публикацији.

Након заглавља записа наводе се контролна поља у формату: ознака контролног поља, садржај контролног поља. Библиографски запис на листингу 4.1 садржи четири контролна поља, то су поља са ознакама 001, 003, 005 и 008. Поље 001 садржи контролни број и у посматраном примеру то је број 92005291. Поље 003 садржи идентификатор контролног броја односно шифру која одређује организацију чији контролни број записа се налази у пољу 001. Контролно поље 005 садржи датум и време последње измене записа. Контролно поље 008 има сложенију структуру и садржи неколико информација. Првих шест карактера резервисано је за датум креирања записа, затим следи ознака године издавања иза које се наводи година издавања. У посматраном примеру библиографског записа ознака године издавања је s и означава једини познати датум/вероватни датум. Година издавања је 1993. На позицији 35-37 налази се шифра за језик публикације. У примеру је то шифра eng која означава да је публикација написана на енглеском језику.

После контролних поља следи листа поља записа. Сваки ред који се односи на поље садржи ознаку поља, вредности првог и другог индикатора или знак # за индикатор који није дефинисан и листу потпоља која припадају посматраном пољу заједно са садржајем тих потпоља. Библиографски запис приказан на листингу 4.1 садржи поље *245* које се односи на стварни наслов публикације. Вредност првог индикатора поља *245* даје информацију о томе да ли је наслов значајан, односно да ли је наслов одредница. У посматраном примеру вредност првог индикатора је *1* што значи да наслов јесте одредница. Вредност другог индикатора поља *245* означава број карактера у наслову који се не сортирају и у примеру је тај број *0*. Потпоље *a* поља *245* садржи главни стварни наслов, у посматраном примеру наслов публикације је *Arithmetic* и наведен је након ознаке *$a* у реду у ком се налази поље 245. Потпоље *a* поља *520* садржи кратак садржај или синопсис дела. Поље *650* по MARC 21 стандарду има особину поновљивости и садржи податке о тематској

предметној одредници дела. Посматрани библиографски запис садржи пет поља *650*, што значи да је публикацији додељено пет тематских предметних одредница. У потпоље *a* поља *650* уноси се текст предметне одреднице, док се у потпоље *x* уноси предметна пододредница која ближе одређује предметну одредницу.

4.1.1 Ограничења у MARC 21 формату

Детаљном анализом MARC 21 формата могу се уочити ограничења која се односе на структуру и садржај записа. Ограничењима над структуром дефинисано је која поља и потпоља се могу понављати у запису, која је дозвољена дужина елемената записа као и да ли се сви елементи записа дефинисани форматом. Ограничења садржаја записа дефинисана су над садржајем заглавља записа, директоријума, индикаторима, контролних поља и потпоља. Поред тога, постоје ограничења дефинисана између различитих података у истом запису.

Пример структурног ограничења за поље *016* је да постоји први индикатор, други индикатора није дефинисан, а ово поље може да садржи потпоља *a*, *z*, *2* и *8* од којих су *z* и *8* поновљива.

Ограничење садржаја за први индикатор поља *016* је да узима вредност # или *7*. Садржај подпоља *2* поља *016* је шифриран (кодна листа организација). Ако је вредност првог индикатора *7* онда мора да постоји садржај потпоља *2*.

Неке карактерске позиције или потпоља имају тачно одређени формат податка који у њих може да се унесе. Један пример је потпоље a поља *020* у које се уноси ISBN број који се састоји од цифара или знака X и мора бити дужине 10 или 13.

4.2 UNIMARC ФОРМАТ

Структуру библиографског записа по UNIMARC стандарду чини коначан скуп поља, која су груписана у десет блокова:

0 – блок за идентификацију,
1 – блок кодираних информација,
2 – блок главног описа,
3 – блок напомена,
4 – блок за повезивање,
5 – блок сродних наслова,
6 – блок садржајне анализе,
7 – блок података о одговорности,
8 – блок за међународну употребу,

9 – блок за националну употребу.

Идентификатор поља састоји се од три цифре, и прва цифра означава блок коме поље припада. Сви основни концепти MARC 21 формата као што су поља, потпоља и индикатори као и њихове особине постоје и у UNIMARC формату, али се ова два стандарда разликују у начину организације библиографских података.

UNIMARC формат уводи особину *секундарности* за поља. Поља која имају особину секундарности могу се појавити као садржај потпоља поља из блока 4 (Блок за повезивање библиографских записа).

За разлику од MARC 21 формата, интерпункција прописана ISBD стандардом [9], по UNIMARC стандарду не укључује се у садржај потпоља записа.

4.3 XML ШЕМА БИБЛИОГРАФСКИХ ФОРМАТА

У раду (Dimić, et al., 2010) дата је XML шема која описује формате MARC 21 и UNIMARC. Појава ове шеме је XML документ одговарајућег формата. XML документ MARC 21 формата који је инстанца предложене шеме коришћен је у софтверском систему за каталогизацију по MARC 21 формату.

Полазну основу за истраживање у раду (Dimić, et al., 2010) представља рад са конференције IFLA (Curvalho and Cordiero, 2002). У том раду је описана улога XML технологија у библиотечким информационим системима и то у три нивоа, а то су транспорт библиографских података, специфицирање података за валидацију записа у односу на библиографски формат и спецификација сервиса путем којих се ови подаци могу користити. Овакав модел представљања MARC спецификације се у литератури назива TVC модел (Transport, Validation и Services). Демонстрација ових истраживања дата је у [17].

У поменутом раду је критикована идеја о креирању компликованих XML шема или DTD-а којама би се могли представљати само записи који су валидни. Уместо тога, дат је предлог да се приликом коришћења XML-а транспорт раздвоји од спецификације. У њему се каже да је уместо покушаја де се споје структура, синтакса и семантика у једну XML спецификацију боље раздвојити на две спецификације, а то су:

- спецификација формата за транспорт записа, чија је улога да омогући ефикасан транспорт библиографских података и
- спецификација XML формата за представљање семантике MARC-а, односно креирање XML документа који би представљао једну врсту MARC Manual-а.

Тема рада (Dimić, et al., 2010) односи се на други случај примене XML-а. Циљ је био да се изврши спецификација XML документа који садржи све податке који се односе на библиографске формате UNIMARC и MARC 21 и који би се користио као улазна информација у софтверским алатима за каталогизацију.

4.3.1 Структура формата MARC 21 и UNIMARC

У овом одељку описани су концепти библиографских формата који ће бити моделирани у XML шеми. Анализа концепата MARC 21формата извршена је на основу документа *MARC 21Format for Bibliographic Data* који је доступан на [46], анализа концепата UNIMARC формата извршена на основу документа *UNIMARC Manual: Bibliographic Format 1994- IFLA Universal Bibliographic Control and International MARC Core Programme* (*update 3, 1 March 2000*) који је доступан на [13].

Структура UNIMARC и MARC 21 формата је слична. Они се састоје од заглавља записа, директоријума и коначног скупа поља.

Заглавље записа (*leader*) налази се на почетку сваког записа и садржи податке за обраду записа. Састоји се од 24 знака означена са 00 до 23. Позиција знака указује на податак који носи. На пример, у заглављу и UNIMARC и MARC 21 записа, првих 5 цифара (позиција 00-04) означавају дужину записа, тј. број карактера који га чине. На позицији 05 (шеста цифра) налази се код који означава стање записа, итд.

Директоријум се састоји од троцифрене ознаке поља за свако поље, његове дужине и почетног места знака у односу на прво поље података.

Поља могу сама по себи бити носиоци информација или могу да садрже највише два индикатора и коначан скуп потпоља. Индикатори додатно одређују садржај поља, а потпоља су у том случају носиоци информација.

Опис ових библиграфских формата помоћу XML шема језика може се разматрати са два аспекта. Први је формирање XML шеме тако да се описују све информације о формату појединачно. На пример, свако потпоље описано је као елемент XML шеме. На тај начин описан је и сваки део заглавља записа и директоријума.

Други аспект формирања XML шеме је такав да су елементи XML шеме концепти библиографских формата. На пример, концепт је потпоље тако да је елеменат XML шеме потпоље а сва потпоља су појаве тог концепта. XML шема која је креирана на овај начин је једноставнија за коришћење у имплементацији софтвера за каталогизацију и овај принцип је прихваћен у раду (Димић и Милосављевић, 2009).

Основни концепти библиографских формата MARC 21 i UNIMARC су:

- заглавље записа - leader (MARC21), label (UNIMARC),
- директоријум,
- контролна поља,
- поља,
- шифарници.

Наведени концепти су сложени концепти који садрже друге концепте формата. Концепт заглавље записа и контролно поља садрже концепт *позиције карактера*, а поља садрже концепт *потпоља* и *индикатора*. Концепти индикатора, неких позиција карактера и потпоља могу да имају предефинисани скуп дозвољених вредности, то су шифарници који се односе само на један елемент формата.

У наставку су наведене особине и структура наведених концепата који представљају основу за креирање XML шеме.

Заглавље записа (*leader* у MARC 21, односно лабел у UNIMARC-у) се састоји од низа података непроменљиве дужине који се идентификују према релативној позицији карактера. Овај део записа има различите називе у MARC 21 и UNIMARC-у, а може се и превести на неки други језик, зато се за њега везује и назив којим се овај концепт формата идентификује у конкретном систему за каталогизацију. По UNIMARC формату за све концепте формата везује се обавезност, па тако и за заглавље записа.

Заглавље записа садржи низ **позиција карактера**. Позиција карактера је дефинисана индексом почетне и крајње позиције и описом информације коју садржи. Позиција карактера може имати скуп дозвољених вредности у виду референце на неки шифарник или има сопствену листу дозвољених вредности. Податак у позиција карактера може бити компјутерски генерисан, а неке позиције карактера имају и предефинисане вредности.

Директоријум се састоји од низа елемената који садрже три податка у 12 карактера. Сваки податак у директоријуму представља једну позицију карактера која се системски генерише. Директоријум је дефинисан за MARC 21 формат, али не и за UNIMARC.

Контролна поља одређена су својим именом, описом података који се смештају у то поље И особином поновљивости. Спецификација неких контролних поља у MARC 21 формату зависи од врсте грађе (*material type*) или категорије грађе (*category of material*). Нека контролна поља садрже појединачан елемент података. За ова поља потребно је навести да ли се подаци у њима компјутерски генеришу или их уноси корисник. Контролна поља се могу састојати од низа позиција карактера које су описане у оквиру

концепта заглавље записа. По UNIMARC формату не постоје контролна поља под тим именом као што је то случај са MARC 21 форматом, али постоје поља која имају структуру и особине контролних поља и таква поља имају и особину обавезности.

Поља се описују помоћу назива поља, описа и особине поновљивости. По UNIMARC формату за поља се везије још и особина обавезности. Поља могу имати највише два индикатора и коначан скуп потпоља. По UNIMARC формату постоји блок 4 (блок за повезивање). Поља из овог блока у оквиру својих потпоља могу садржати друга поља која имају своје индикаторе и потпоља.

Потпоља су основни носиоци информација и имају свој назив, опис података који се смештају у потпоље, особину поновљивости у оквиру поља. Потпоље може имати скуп предефинисаних вредности као и позиција карактера. Тај скуп дозвољених вредности је или референца на шифарник или скуп кодова који су везани само за то потпоље. Постоје потпоља и у MARC 21 и у UNIMARC формату која имају предефинисани формат података фиксне дужине и садрже низ позиција карактера. По UNIMARC формату потпоља имају особину обавезности у оквиру поља.

Индикатор је везан за поље и може бити први или други. За сваки индикатор се везује његово значење и скуп дозвољених вредности у облику интерног шифарника.

Шифарник је листа кодова која је прописана неким стандардом или је уводи нека институција. Он може бити везан за само један или више елемената формата и састоји од скупа кодова са придруженим значењем. Сваки шифарник има идентификацију преко које се елементи формата могу референцирати, као и свој назив и назив институције или стандарда која га прописује.

4.3.2. XML шема

На основу анализе наведених концепата формата MARC 21и UNIMARC креирана је XML шема. Елементом *format* моделиран је коренски елемент који садржи остале сложене концепте формата. Листинг овог елемента је:

```
<xs:element name="format">
 <xs:annotation>
  <xs:documentation>UNIMARC and MARC21 format</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
```

```
   <xs:element ref="leader"/>
   <xs:element ref="directory" minOccurs="0"/>
   <xs:element ref="controlfield" maxOccurs="unbounded"/>
   <xs:element ref="datafield" maxOccurs="unbounded"/>
   <xs:element ref="externalCodeList" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
```

Елемент *format* је комплексног типа и садржи секвенцу осталих сложених концепата библиографског формата који су моделирани као глобални елементи. Елемент *format* има и два атрибута који садрже назив формата и његов опис.

Појава елемента *leader* у оквиру формата је обавезна јер је овај део формата дефинисан и за MARC 21 и за UNIMARC. Директоријум је дефинисан само за MARC 21 формат и зато је његово појављивање у оквиру елемента *format* опционо, што је представљено атрибутом *minOccurs="0"*. Остали елементи формата, контролна поља, поља и шифарници имају вишеструко појављивање у оквиру елемента формат што је представљено атрибутом *maxOccurs="unbounded"*.

Концепт *заглавље записа*

Елементом *leader* представљен је концепт заглавља записа. Он је комплексног типа и представља низ позиција карактера. Овај елемент има и три атрибута, то су *name* којим се специфицира назив овог дела формата, атрибут *repeatable* којим је представљена поновљивост и коме је додељена фиксна вредност *false* и опциони атрибут за обавезност који ће се користити за представљање UNIMARC формата и он има фиксну вредност *true*. Листинг елемента *leader* је:

```
<xs:element name="leader">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="charposition" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="optional"
default="LEADER"/>
  <xs:attribute name="repeatable" type="xs:boolean" use="required"
fixed="false"/>
```

```
  <xs:attribute name="mandatory" type="xs:boolean" use="optional"
fixed="true"/>
 </xs:complexType>
</xs:element>
```

Елемент *charposition* представља позицију карактера. Сваки елемент *charposition* одређују његови атрибути, а то су индекс почетне позиције (*start)* индекс крајње позиције (*end)*, опис податка који се смешта на ову позицију карактера (*description*), опционално може имати и подразумевану вредност (*default value*) и атрибут који специфицира да ли се вредност за позицију карактера системски генерише. Позиција карактера може опционо имати поделемент *codelist* који представља листу дозвољених вредности за позицију карактера. Листинг елемента *charposition* је:

```
<xs:element name="charposition">
 <xs:complexType>
  <xs:sequence minOccurs="0">
   <xs:element ref="codelist"/>
  </xs:sequence>
  <xs:attribute name="start" type="xs:int" use="required"/>
  <xs:attribute name="end" type="xs:int" use="required"/>
  <xs:attribute name="description" type="xs:string" use="required"/>
  <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
  <xs:attribute name="isSystemGenerated" type="systemGenerated"
use="optional"/>
 </xs:complexType>
</xs:element>
```

Елементом *codelist* моделирана је листа кодова. То је сложени тип који садржи један од два поделемента, листу кодова која се односи само на један елемент формата и која је представљена елементом *internalCodeList* или референцу на шифарник која је садржај елемента *externalCodeListID*. Листинг овог елемента је:

```
<xs:element name="codelist">
 <xs:complexType>
  <xs:choice>
   <xs:element ref="internalCodeList"/>
   <xs:element name="externalCodeListID" type="xs:string"/>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

Листа кодова који се односе на један елемент формата представљена је елементом *internalCodeList*. То је секвенца елемената који представљају ставке у шифарнику и он је у XML шеми представљен на следећи начин:

```
<xs:element name="internalCodeList">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="item" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Свака ставка у шифарнику представљена је елементом *item* који се састоји од шифре и значења те шифре. Листинг овог елемента је:

```
<xs:element name="item">
 <xs:complexType>
  <xs:attribute name="code" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

Атрибут *isSystemGenerated* специфицира да ли се подаци у елементима за које се веже системски генеришу. Овај елемент је типа *systemGenerated* који представља рестрикцију на string и може имати следеће вредности: *never* – податак се никада системски не генерише, *may be* – податак се може системски генерисати, *usually* – податак се обично системски генерише и *always* – податак се увек системски генерише. Листинг типа *systemGenerated* је:

```
<xs:simpleType name="systemGenerated">
 <xs:restriction base="xs:string">
  <xs:enumeration value="never"/>
  <xs:enumeration value="may be"/>
  <xs:enumeration value="usually"/>
  <xs:enumeration value="always"/>
 </xs:restriction>
</xs:simpleType>
```

Концепт *деректоријум*

Елементом *directory* представљен је концепт директоријума. Директоријум је дефинисан само за MARC 21 формат. Елемент *directory* је комплексног типа и састоји се од секвенце позиција карактера, има један атрибут, *repeatable* коме

је исто као код заглавља записа додељена фиксна вредност *false*. Листинг елемента *directory* је:

```
<xs:element name="directory">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="charposition" maxOccurs="3"/>
   </xs:sequence>
   <xs:attribute name="repeatable" type="xs:boolean" use="required"
fixed="false"/>
  </xs:complexType>
 </xs:element>
```

Концепт *контролно поље*

Елементом *controlfield* моделирано је контролно поље формата. Овај елемент је сложеног типа и може садржати секвенцу позиција карактера. Елемент *controlfield* има атрибуте којиме се специфицирају назив контролног поља, његов опис, особине поновљивости и обавезности, два атрибута којима се специфицира врста грађе и категорија грађе на које се поље односи, као и атрибут који одређује да ли се вредност контролног поља системски генерише. Листинг овог елемента је:

```
<xs:element name="controlfield">
  <xs:complexType>
   <xs:sequence minOccurs="0">
    <xs:element ref="charposition" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="description" type="xs:string" use="required"/>
   <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
   <xs:attribute name="mandatory" type="xs:boolean" use="optional"/>
   <xs:attribute name="materialType" type="materialTypeType" use="optional"/>
   <xs:attribute name="materialCategory" type="materialCategoryType"
use="optional"/>
   <xs:attribute name="isSystemGenerated" type="systemGenerated"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Елементи *materialTypeType* и *materialCategoryType* дефинисани су као енумерације свих могућих вредности за врсту односно категорију грађе.

Концепт *поље*

Елементом *datafield* представљено је поље формата. Овај елемент је секвенца потпоља формата и индикатора који се могу појавити од 0 до 2 пута. Од атрибута овај елемент садржи назив, опис, поновљивост и обавезност која се везује за UNIMARC формат. Атрибут *embaddedFieldsAllowed* је такође везан за UNIMARC формат и одређује да ли ово поље у оквиру својих потпоља може да садржи друга, угњеждена поља. Листинг елемента *datafield* је:

```
<xs:element name="datafield">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="subfield" maxOccurs="unbounded"/>
    <xs:element ref="indicator" minOccurs="0" maxOccurs="2"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="description" type="xs:string" use="required"/>
   <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
   <xs:attribute name="mandatory" type="xs:boolean" use="optional"/>
   <xs:attribute name="embaddedFieldsAllowed" type="xs:boolean"
use="optional"/>
  </xs:complexType>
 </xs:element>
```

Референцом на елемент *subfield* представљено је потпоље које се вишеструко јавља у оквиру поља. Потпоље формата је одређено атрибутима елемента *subfield*, а то су назив потпоља, опис и поновљивост потпоља у оквиру поља, а постоји и опциони атрибут за подразумевану вредност за потпоље. Потпоља могу опционо имати или низ позиција карактера или листу шифара што је представљено елементом *choice* који се састоји од елемената *charposition* и *codelist*. Листинг елемента *subfield* је:

```
<xs:element name="subfield">
  <xs:complexType>
   <xs:choice minOccurs="0">
    <xs:element ref="charposition" maxOccurs="unbounded"/>
    <xs:element ref="codelist"/>
   </xs:choice>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="description" type="xs:string" use="required"/>
   <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
   <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
  </xs:complexType>
```

```
</xs:element>
```

Индикатор је представљен елементом *indicator*. То је сложени елемент и садржи елемент *internalCodeList* којим је представљена листа дозвољених вредности за индикатор. Индикатор је одређен атрибутима *description* који садржи опис информације коју индикатор носи и атрибут *index* који је рестрикција на *intiger* и може имати вредност 1 или 2 и одређује да ли се ради о првом или другом индикатору за поље коме индикатор припада. Индикатор опционо може имати и подразумевану вредност. Листинг елемента *indicator* је:

```
<xs:element name="indicator">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="internalCodelist"/>
  </xs:sequence>
  <xs:attribute name="description" type="xs:string" use="required"/>
  <xs:attribute name="index" use="required">
   <xs:simpleType>
    <xs:restriction base="xs:int">
     <xs:enumeration value="1"/>
     <xs:enumeration value="2"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
```

Концепт *шифарник*

Елемент *externalCodeList* представља шифарник формата. Он је секвенца кодова, моделираних елементом *item*, а од атрибута има свој id преко ког ће се елементи формата референцирати на њега, назив екстерног шифарника и извор који представља институцију или стандард који је извор овог шифарника. Листинг елемента *externalCodeList* је:

```
<xs:element name="externalCodeList">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="item" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="name" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="source" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
```

4.3.3 XML документ MARC 21 формата

XML документ који је инстанца шеме описане у претходном одељку садржи појаве концепата библиографских формата. Коренски елемент XML документа за MARC 21 формат изгледа овако:

```
<format name="MARC 21 format" description="MARC 21 format for
bibliographic data">.
```

Први поделемент елемента *format* је *леадер*. У наставку је дат листинг елемента *leader* са прве две позиције карактера. Обе ове позиције карактера се обично системски генеришу. Листинг је:

```
<leader repeatable="false" name="LEADER">
 <charposition start="00" end="04" description="Record length
isSystemGenerated="usually"/>
 <charposition start="05" end="05" description="Record status"
isSystemGenerated="usually">
  <codelist>
   <internalCodeList>
    <item code="a" value="Increase in encoding level"/>
    <item code="c" value="Corrected or revised"/>
    <item code="d" value="Deleted"/>
    <item code="n" value="New"/>
    <item code="p" value="Increase in encoding level from prepublication"/>
   </internalCodeList>
  </codelist>
 </charposition>
```

По MARC 21 формату постоје контролна поља која садрже слободан текст, нека имају дефинисане позиције карактера, а нека контролна поља, као што су поља са ознаком 006 и 008 су различито дефинисана за различите врсте грађе. Пример спецификације неколико контролних поља и поља 006 за књиге је:

```
<controlfield repeatable="false" description="Control Number" name="001"
isSystemGenerated="may be"/>
 <controlfield repeatable="false" description="Control Number Identifier"
name="003" isSystemGenerated="usually"/>
 <controlfield repeatable="false" description="Date and Time of Latest
Transaction" name="005" isSystemGenerated="always"/>
```

```
<controlfield repeatable="false" description="Fixed-Length Data Elements-
Additional Material" name="006" materialType="BK">
 <charposition start="00" end="00" description="Form of material">
  <codelist>
   <internalCodeList>
    <item code="a" value="Language material"/>
    <item code="t" value="Manuscript language material"/>
   </internalCodeList>
  </codelist>
 </charposition>
 <charposition start="01" end="04" description="Illustrations">
  <codelist>
   <internalCodeList>
    <item code="#" value="No illustrations"/>
    <item code="a" value="Illustrations"/>
    <item code="b" value="Maps"/>
    <item code="c" value="Portraits"/>
    <item code="d" value="Charts"/>
    <item code="e" value="Plans"/>
    <item code="f" value="Plates"/>
    <item code="g" value="Music"/>
    <item code="h" value="Facsimiles"/>
    <item code="i" value="Coats of arms"/>
    <item code="j" value="Genealogical tables"/>
    <item code="k" value="Forms"/>
    <item code="l" value="Samples"/>
    <item code="m" value="Phonodisc, phonowire, etc."/>
    <item code="o" value="Photographs"/>
    <item code="p" value="Illuminations"/>
    <item code="|" value="No attempt to code"/>
   </internalCodeList>
  </codelist>
 </charposition>
 <charposition start="05" end="05" description="Target audience"/>
```

Поље 010 садржи 4 потпоља и његов листинг је:

```
<datafield repeatable="false" description="Library of Congress Control Number"
name="010">
 <subfield repeatable="false" description="LC control number" name="a"/>
 <subfield repeatable="true" description="NUCMC control number" name="b"/>
```

```
 <subfield repeatable="true" description="Canceled/invalid LC control number"
name="z"/>
 <subfield repeatable="true" description="Field link and sequence number"
name="8"/>
 </datafield>
```

На крају XML документа формата наведени су сви екстерни шифарници, а један од њих је и шифарник за земље који прописује *Network Development and MARC Standards Office, Library of Congress*. Назив институције која прописује шифарник наводи се као садржај атрибута *source*. Листинг једног дела овог шифарника је:

```
<externalCodeList id="marccountry" name="MARC Code List for Countries"
source="Network Development and MARC Standards Office, Library of
Congress">
 <item code="af" value="Afghanistan"/>
 <item code="alu" value="Alabama"/>
 <item code="aku" value="Alaska"/>
 <item code="aa" value="Albania"/>
 <item code="abc" value="Alberta"/>
 <item code="ae" value="Algeria"/>
 <item code="as" value="American Samoa"/>
 <item code="an" value="Andorra"/>
 <item code="ao" value="Angola"/>
 <item code="am" value="Anguilla"/>
 <item code="ay" value="Antarctica"/>
 <item code="aq" value="Antigua and Barbuda"/>
 <item code="ag" value="Argentina"/>
 <item code="azu" value="Arizona"/>
```

4.3.4 XML документ UNIMARC формата

Појава приказане XML шеме за UNIMARC формат има следећи коренски елемент:

```
<format name="UNIMARC format" description="UNIMARC Manual :
Bibliographic Format 1994">
 <leader repeatable="false" name="LABEL" mandatory="true">
 <charposition start="0" end="4" description="Record length"/>
 <charposition start="5" end="5" description="Record status"/>
 <charposition start="6" end="6" description="Type of record"/>
 <charposition start="7" end="7" description="Bibliographic level"/>
```

UNIMARC формат садржи поља која немају потпоља и индикаторе, и то су поља која су у XML документу представљена као контролна поља. У наставку су дати XML елементи који представљају поља 001 и 005:

```
<controlfield  repeatable="false" description="RECORD IDENTIFIER"
name="001" mandatory="true"/>
<controlfield repeatable="false" description="VERSION IDENTIFIER"
name="005" mandatory="false">
```

Као што је већ наведено постоје поља у UNIMARC формату која као садржај својих потпоља могу имати друга поља. Једно такво поље је и поље 464 чија спецификација је:

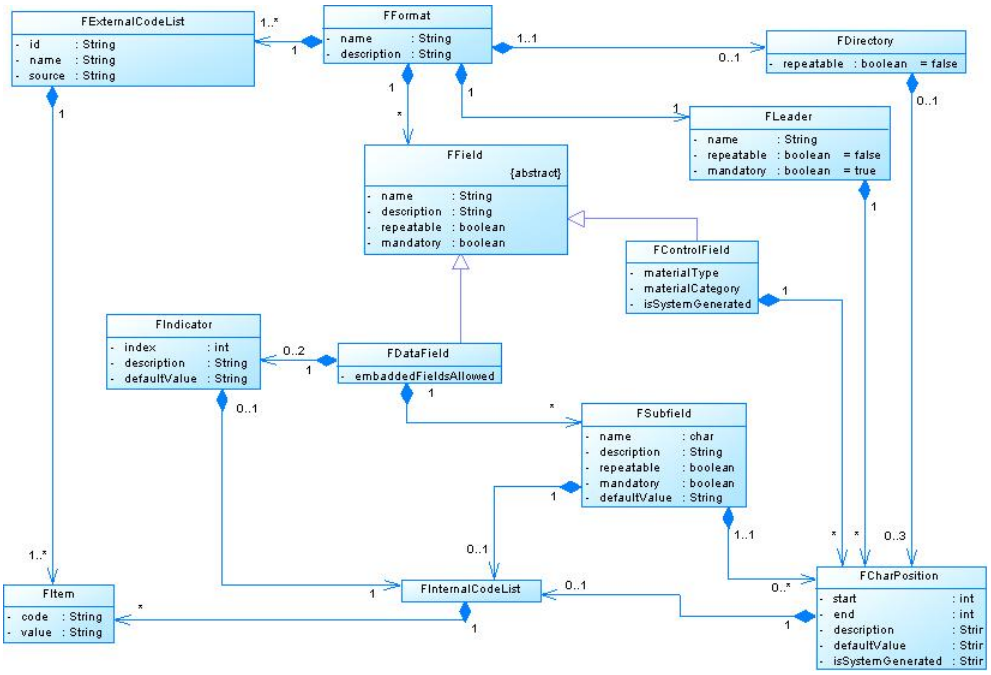
```
<datafield repeatable="true" description="PIECE-ANALYTIC" name="464"
embaddedFieldsAllowed="true" mandatory="false">
  <subfield repeatable="true" description="Embedded tag, indicators and subfields"
name="1"></subfield>
 </datafield>
```

Сви елементи који су илустровани у оквиру XML документ за MARC 21 су исти и за UNIMARC формат.

4.4 ОБЈЕКТНИ МОДЕЛ БИБЛИОГРАФСКИХ ФОРМАТА

XML документ библиографског формата може се користити у разним софтверским алатима за рад са библиографским подацима. У оквиру система БИСИС развијена је софтверска компонента за рад са подацима о библиографским форматима. Ова софтверска компонента имплементирана је у Јава окружењу.

Један од пакета ове софтверске компоненте садржи класе којима је представљен објектни модел библиографског формата. Дијаграм класа овог објектног модела приказан је на слици 4.1. Овај дијаграм класа креиран је на основу шеме која је дата у овом раду и сваки елемент шеме представљен је посебном класом. Називи класа одговарају називима из XML шеме са префиксом *F* који означава да се ради о формату. На дијаграму класа са слике 4.1 приказани су атрибути класа који одговарају атрибутима из XML шеме. Операције класа састоје се од *set* и *get* метода за сваки атрибут помоћу којих је могуће ажурирати податке о формату у оквиру објектог модела.



Слика 4.1. Дијаграм класа објектног модела библиографских формата

4.5 XML ШЕМЕ БИБЛИОГРАФСКОГ ЗАПИСА ПО MARC 21 СТАНДАРДУ

У овом одељку је дат преглед постојећих XML шема библиографских записа по MARC 21 стандарду. Описане су две шеме и два различита приступа у моделирању структуре библиогарфских записа по овом формату.

4.5.1 MARC XML шема

У оквиру MARC XML [51] пројекта који води служба за развој MARC стандарда Конгресне библиотеке развијена је XML шема за MARC записе [52]. То је XML формат чија дефиниција у XML шеми одређује називе тагова XML елемената, који одговарају заглављу записа, пољима, потпољима и индикаторима MARC записа, и све њихове могуће вредности и комбинације. Назив тага XML елемента који одговара заглављу записа је *leader*. Називи тагова XML елемената који одговарају контролним пољима су *controlfield*. XML елементи за контролна поља садрже атрибут *tag* са идентификатором контролног поља. Називи тагова XML елемената који одговарају пољима су *datafield*. XML елементи за поља садрже атрибуте *ind1* и *ind2* са првим и другим индикатором поља, и атрибут *tag* са идентификатором поља. Називи тагова XML елемената који одговарају потпољима су *subfield*. XML елементи

за потпоља садрже атрибут *code* са идентификатором потпоља. У наставку је дат листинг ове XML шеме.

```
<?xml version="1.0"?>
<xsd:schema xmlns="http://www.loc.gov/MARC 21/slim"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.loc.gov/MARC 21/slim"
elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.1"
xml:lang="en">
      <xsd:element name="record" type="recordType"
nillable="true" id="record.e">
      </xsd:element>
      <xsd:element name="collection" type="collectionType"
nillable="true" id="collection.e">
            <xsd:annotation>
                  <xsd:documentation>collection is a top
level container element for 0 or many
records</xsd:documentation>
            </xsd:annotation>
      </xsd:element>
      <xsd:complexType name="collectionType"
id="collection.ct">
            <xsd:sequence minOccurs="0"
maxOccurs="unbounded">
                  <xsd:element ref="record"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="idDataType"
use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="recordType" id="record.ct">
            <xsd:sequence minOccurs="0">
                  <xsd:element name="leader"
type="leaderFieldType"/>
                  <xsd:element name="controlfield"
type="controlFieldType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="datafield"
type="dataFieldType" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="type" type="recordTypeType"
use="optional"/>
            <xsd:attribute name="id" type="idDataType"
use="optional"/>
      </xsd:complexType>
      <xsd:simpleType name="recordTypeType" id="type.st">
```

```
            <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="Bibliographic"/>
                  <xsd:enumeration value="Authority"/>
                  <xsd:enumeration value="Holdings"/>
                  <xsd:enumeration value="Classification"/>
                  <xsd:enumeration value="Community"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="leaderFieldType" id="leader.ct">
            <xsd:annotation>
                  <xsd:documentation>MARC 21 Leader, 24
bytes</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                  <xsd:extension base="leaderDataType">
                        <xsd:attribute name="id"
type="idDataType" use="optional"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
      <xsd:simpleType name="leaderDataType" id="leader.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
                  <xsd:pattern value="[\d ]{5}[\dA-Za-z
]{1}[\dA-Za-z]{1}[\dA-Za-z ]{3}(2| )(2| )[\d ]{5}[\dA-Za-z
]{3}(4500|    )"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="controlFieldType"
id="controlfield.ct">
            <xsd:annotation>
                  <xsd:documentation>MARC 21 Fields 001-
009</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                  <xsd:extension base="controlDataType">
                        <xsd:attribute name="id"
type="idDataType" use="optional"/>
                        <xsd:attribute name="tag"
type="controltagDataType" use="required"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
```

```
      <xsd:simpleType name="controlDataType"
id="controlfield.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="controltagDataType"
id="controltag.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
                  <xsd:pattern value="00[1-9A-Za-z]{1}"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="dataFieldType"
id="datafield.ct">
            <xsd:annotation>
                  <xsd:documentation>MARC 21 Variable Data
Fields 010-999</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence maxOccurs="unbounded">
                  <xsd:element name="subfield"
type="subfieldatafieldType"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="idDataType"
use="optional"/>
            <xsd:attribute name="tag" type="tagDataType"
use="required"/>
            <xsd:attribute name="ind1"
type="indicatorDataType" use="required"/>
            <xsd:attribute name="ind2"
type="indicatorDataType" use="required"/>
      </xsd:complexType>
      <xsd:simpleType name="tagDataType" id="tag.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
                  <xsd:pattern value="(0([1-9A-Z][0-9A-
Z])|0([1-9a-z][0-9a-z]))|(([1-9A-Z][0-9A-Z]{2})|([1-9a-z][0-
9a-z]{2}))"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="indicatorDataType" id="ind.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
                  <xsd:pattern value="[\da-z ]{1}"/>
```

```
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="subfieldatafieldType"
id="subfield.ct">
            <xsd:simpleContent>
                  <xsd:extension base="subfieldDataType">
                        <xsd:attribute name="id"
type="idDataType" use="optional"/>
                        <xsd:attribute name="code"
type="subfieldcodeDataType" use="required"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
      <xsd:simpleType name="subfieldDataType"
id="subfield.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="subfieldcodeDataType"
id="code.st">
            <xsd:restriction base="xsd:string">
                  <xsd:whiteSpace value="preserve"/>
                  <xsd:pattern value="[\da-
z!&amp;quot;#$%&amp;amp;'()*+,-./:;&amp;lt;=&amp;gt;?{}_^`~\[\]\\]{1}"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="idDataType" id="id.st">
            <xsd:restriction base="xsd:ID"/>
      </xsd:simpleType>
</xsd:schema>
```

Основни недостаци приказане шеме односе се на структуру XML библиографског записа који она описује. На пример, заглавље записа је пренето у један XML елемент *leader* и не у више XML елемената који зависе од позиције. Иста примедба односи се и на контролна поља. Овакво решење би захтевало додатну обраду података да би се правилно обрадио XML библиографски запис.

4.5.2 OAI MARC шема

Једно решење за моделирање библиографских записа по MARC 21 стандарду у XMLSchema језику предложила је организација *Open Archives Initiative* [53].

То је организација која развија и промовише стандарде за интероперабилност софтверских система.

XML шема представљена у овом одељку [54] подржава све концепте MARC 21 записа и слична је шеми описаној у претходном одељку. Разлика између ове две шеме тиче се назива елемената записа, али најбитнија разлика односи се на структуру XML документа који ове шеме описују. Према OAI шеми заглавље записа не постоји као посебан елемент већ су неки његови подаци издвојени као вредности атрибута коренског елемента библиографског записа. На овај начин олакшана је обрада података из заглавља записа, али недостатак овог решења односи се на одсуство неких података који су дефинисани у заглављу записа, а нису моделирани одговарајућим атрибутима коренског елемента библиографског записа.

Листинг XML шеме предложен од стране ове организације дат је у наставку.

```
<!-- edited with XMLSpy v2006 sp1 U (http://www.altova.com)
by bojana (EMBRACE) -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:oai_marc="http://www.openarchives.org/OAI/oai_marc"
targetNamespace="http://www.openarchives.org/OAI/oai_marc"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<element name="oai_marc">
   <complexType>
      <sequence>
       <element ref="oai_marc:fixfield"
maxOccurs="unbounded"/>
        <element ref="oai_marc:varfield" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="status" type="string" use="optional"/>
      <attribute name="type" type="string" use="required"/>
      <attribute name="level" type="string" use="required"/>
      <attribute name="ctlType" type="string"
use="optional"/>
      <attribute name="charEnc" type="string"
use="optional"/>
      <attribute name="encLvl" type="string" use="optional"/>
      <attribute name="catForm" type="string"
use="optional"/>
      <attribute name="lrRqrd" type="string" use="optional"/>
    </complexType>
</element>
<element name="fixfield">
```

```
    <complexType>
      <simpleContent>
        <extension base="oai_marc:fixfieldType">
          <attribute name="id" type="oai_marc:idType"
use="required"/>
        </extension>
      </simpleContent>
     </complexType>
</element>
<simpleType name="fixfieldType">
   <restriction base="string">
      <pattern value='".*"'/>
            <!-- fixfield must be enclosed between quotes
because spaces are meaningfull -->
            </restriction>
      </simpleType>
      <element name="varfield">
            <complexType>
                  <sequence>
                        <element ref="oai_marc:subfield"
maxOccurs="unbounded"/>
                  </sequence>
                  <attribute name="id" type="oai_marc:idType"
use="required"/>
                  <attribute name="i1" type="oai_marc:iType"
use="required"/>
                  <attribute name="i2" type="oai_marc:iType"
use="required"/>
            </complexType>
      </element>
      <element name="subfield">
            <complexType>
                  <simpleContent>
                        <extension base="string">
                              <attribute name="label"
type="oai_marc:subfieldType" use="required"/>
                        </extension>
                  </simpleContent>
            </complexType>
      </element>
      <simpleType name="subfieldType">
            <restriction base="string">
                  <pattern value="[0-9 | a-z]"/>
```

```
                  <!-- MARC subfield (the leading $ i not
used)
         may be any lowercase alphabetic or numeric character
-->
            </restriction>
      </simpleType>
      <simpleType name="idType">
            <restriction base="string">
                  <pattern value="[0-9]{1,3}"/>
                  <!-- MARC tags are 1 to 3 digits -->
            </restriction>
      </simpleType>
      <simpleType name="iType">
            <restriction base="string">
                  <pattern value="[0-9 | a-z | \s]"/>
                  <!-- MARC indicator may be any lowercase
alphabetic or numeric character or a blank  -->
            </restriction>
      </simpleType>
</schema>
```

4.5.3 XML шема библиографског записа по UNIMARC стандарду.

Креирање XML шеме библиографских записа по UNIMARC стандарду своди се на допуну једне од две описане XML шеме MARC 21 библиографских записа увођењем елемената који моделирају постојање секундарних поља, онако како је овај случај решен на примеру YUMARC библиографских записа.

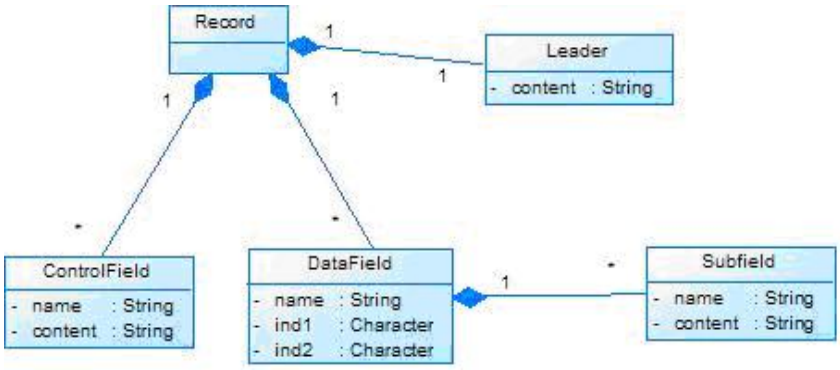
4.6 ОБЈЕКТНИ МОДЕЛИ MARC ЗАПИСА

Основна улога објектних модела библиогарфских записа јесте репрезентација података о запису у апликацији. Подаци о запису се могу учитати у објектни модел из XML докумената чија спецификација је дата у другом поглављу ове тезе. Такође, предвиђено је да се подаци о запису складиште у те XML докуманте. Да би се извршило правилно мапирање података из XML докумената на објектни модел, и обрнуто, потребно је да објектни модел одговара моделу XML шеме према којој се креирају XML библиографски записи.

4.6.1 Објектни модел библиогарфског записа по MARC 21 стандарду

На слици 3.4 дат је дијаграм класа објектног модела по MARC 21 стандарду. Овај дијаграм класа креиран је према XML шеми библиографског записа по MARC 21 формату која је описана у одељку 4.5.1.

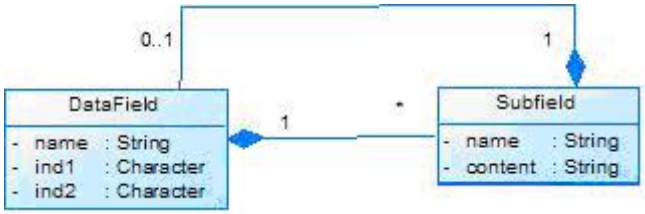
Библиографски запис представљен је класом *Record*. Један запис садржи тачно једно заглавље, представљено класом *Leader* чији атрибут *content* представља садржај заглавља записа и дужине је 24 карактера. Библиографски запис по MARC 21 стандарду садржи две врсте поља, контролна поља и поља која садрже потпоља. Контролна поља су моделирана класом *ControlField* која има два атрибута, атрибут *name* представља ознаку поља коју чине три цифре и *content* који представља садржај контролног поља. Класом *DataField* престављена су поља записа која садрже први и други индикатор, *ind1* и *ind2* и скуп потпоља представљених класом *Subfield* која је везом композиције повезана са класом *DataField*. Потпоље је одређено својом ознаком (атрибут *name*) и садржајем (атрибут *content*).



Слика 4.2 Дијаграм класа објектног модела библиографског записа по MARC 21 стандарду

4.6.2 Објектни модел библиогарфског записа по UNIMARC стандарду

Структура библиогарфског записа по UNIMARC формату је слична структури записа по MARC 21 формату. У оба формата појављују се заглавље записа, контролна поља, поља која садрже потпоља. Поред тога, у UNIMARC формату су дефинисана секундарна поља, то су поља која се појављују као садржај потпоља. Објектни модел UNIMARC записа може се добити на основу објектног модела MARC 21 записа додавањем још једне везе између класе *DataField* и *Subfield* којом је представљено секундарно поље. На слици 4.3 приказана је ова веза.



Слика 4.3 Секундарно поље

Поглавље 5

Спецификација система за каталогизацију по MARC 21 формату

У овом поглављу описан је обједињени процес развоја система за каталогизацију. Развој овог система је вођен случајевима коришћења који су описани у одељку 5.1. Развој се састоји од три итерације које су описане у одељку 5.2. Софтверска архитектура система за каталогизацију описана је у одељку 5.3.

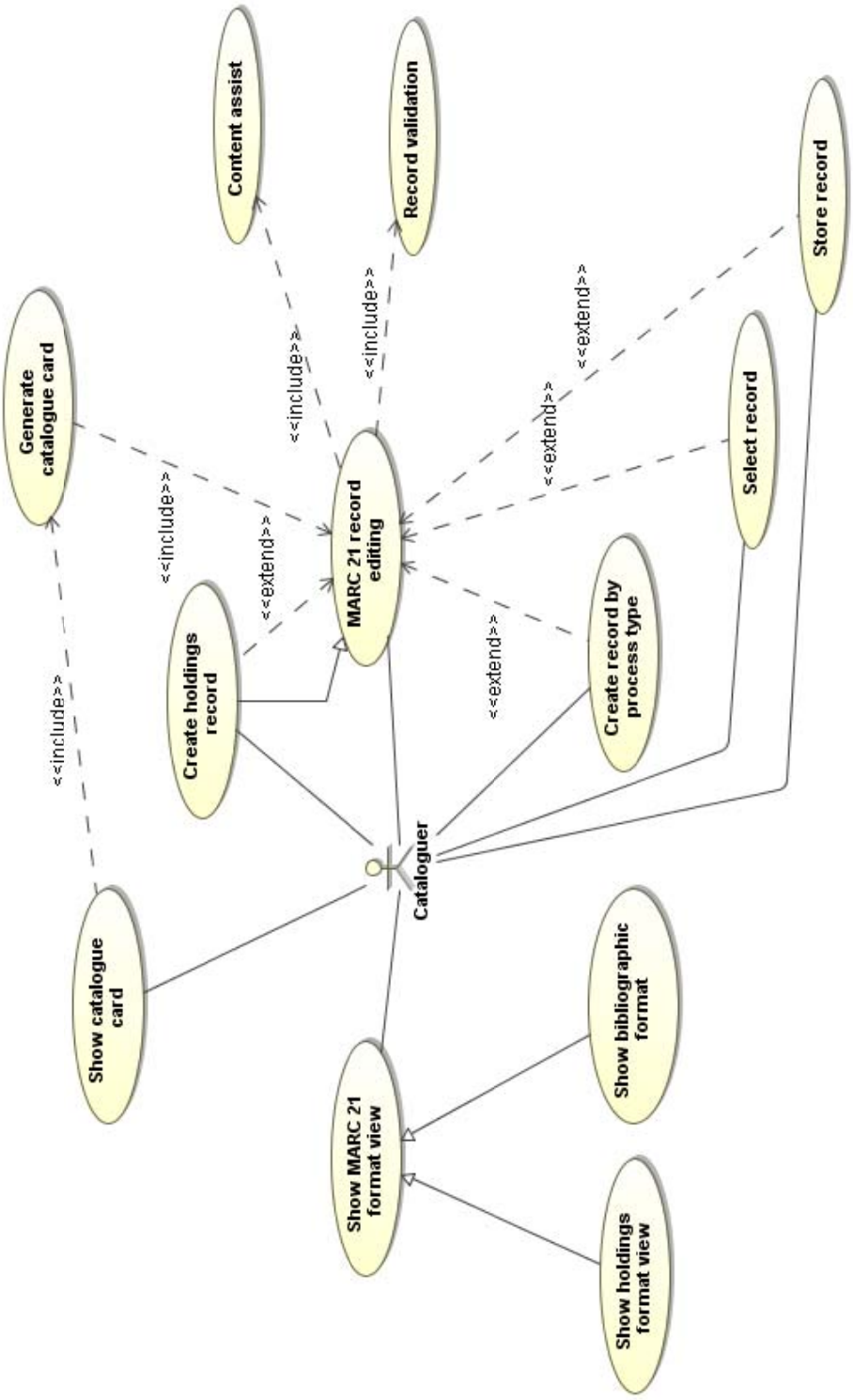
Моделирање система приказано је у облика дијаграма пакета, дијаграма класа и дијаграма секвенци. Ови дијаграми креирани су коришћењем CASE алата MagicDraw верзија 16.0 [55]. Дакле, нотација на приказаним дијаграмима је из овог CASE алата.

5.1 СЛУЧАЈЕВИ КОРИШЋЕЊА

Систем за каталогизацију по MARC 21 формату реализован је коришћењем обједињеног процеса. Прва тачка у реализацији система за каталогизацију је спецификација случајева коришћења. На слици 5.1 приказан је дијаграм случајева коришћења целокупног система за каталогизацију.

Учесник у свим случајевима коришћења система за каталогизацију је библиотекар који врши каталогизацију, односно каталогизатор (*cataloguer*).

Централни случај коришћења на дијаграму случаја коришћења који је приказан на слици 5.1 је *MARC 21 record editing* којим је представљено едитирање MARC 21 записа. Овај случај коришћења укључује два случаја коришћења, то су случај коришћења *Content assist* којим се реализује помоћ за едитирање записа у виду понуде предефинисаних вредности за унос и случај коришћења *Record validation* којим се реализује контрола исправности библиографског записа који се едитира.



Слика 5.1 Дијаграм случајева коришћења система за каталогизацију

Случај коришћења *MARC 21 record editing* проширују четири случаја коришћења, а то су: *Create record by process type*, *Select record*, *Store record*, *Create holdings record.* Случај коришћења *Create record by process type* представља функционалност креирања новог записа на основу изабраног типа обраде, после чега се тај новокреирани запис даље едитира у оквиру сличаја коришћења *MARC 21 record editing*. Случај коришћења *Select record* представља функционалност селектовања и учитавања записа за едитирање. Случај коришћења *Store record* представља функционалност складиштења записа који се едитира у случају коришћења *MARC 21 record editing*. Случај коришћења *Create holding record* представља функционалност креирања записа за локацијске податке на основу записа који се едитира у случају коришћења *MARC 21 record editing*. Функционалност креирање записа за локацијске податке представља такође процес едитирања записа и слична је функционалности која је представљена случајем коришћења *MARC 21 record editing*.

Случај коришћења *Show cataloguing card* представља функционалност приказа каталошког листића и његова реализацију укључује случај коришћења за генерисање каталошког листића, а то је *Generate catalogue card*. Генерисање каталошког листића врши се на основу записа који је креиран у случају коришћења *MARC 21 record editing*.

Случај коришћења *Show MARC 21 format view* представља приказ података о MARC 21 формату и он има две специјализације, то су приказ MARC 21 библиографског формата који је представљен случајем коришћења *Show bibliographic format view* и приказ MARC 21 формата за локацијске податке који је представљен случајем коришћења *Show holdings format view*.

5.2 ИТЕРАЦИЈЕ У РАЗВОЈУ СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

Систем за каталогизацију чији је дијаграм случајева коришћења приказан на слици 5.1 развијен је у три итерације.

Прва итерација

У првој итерацији реализован је случај коришћења *MARC 21 record editing*. У овој итерацији креиран је едитор за унос библиографског записа који је генерисан на основу граматике написане у језику Xtext која представља модел MARC 21 записа.

Ова итерација реализована је коришћењем развоја софтвера вођеним моделима. Модел MARC 21 записа представљен је граматиком на основу које

је генерисан EMF модел записа и програмски код основног едитора за каталогизацију.

Резултат ове итерације је тзв. основни едитор за каталогизацију који подржава основни унос MARC 21 записа.

Друга итерација

У другој итерацији реализовани су случајеви коришћења *Content assist*, *Record validation* и *Generate catalogue card*. У овој итерацији основни едитор је проширен функцијама за генерисање помоћи приликом едитирања, за контролу унетих података, као и трансоформацијом унетог записа у форму каталошког листића.

Ова итерација такође је реализована коришћењем развоја вођеног моделима. Генерисање помоћи за унос реализовано је као проширење EMF модела MARC 21 записа написано у језику Xtend, а контрола унетих података врши се на основу спецификације ограничења над EMF моделом записа написано у језику Check. Генерисање каталошких листића реализовано је као трансформација EMF модела библиографског записа у HTML документ каталошког листића. Ове трансформације специфициране су у виду темплејта написаних у језику Xpand.

Резултат друге итерације је проширени основни едитор који подржава напредне функције едитирања MARC 21 записа.

Трећа итерација

У трећој итерацији реализовани су случајевим коришћења *Show catalogue card*, *Create holdings record*, *Store record*, *Select record*, *Create record by processing type*, *Show MARC 21 format view, Show bibliographic format view* и *Show holdings format view*. У овој итерацији реализоване су све додатне функционалности система за каталогизацију у Eclipse plug-in технологија, а креирани plug-in-ови издвојени су у независну RCP апликацију.

У трећој итерацији коришћен је развој софтвера заснован на компонентама, а за реализацију случајева коришћења за приказ MARC 21 формата *Show MARC 21 format view, Show bibliographic format view* и *Show holdings format view* и развој заснован на XML-у. Софтверске компоненте су plug-in-ови Eclipse платформе који међусобно комуницирају преко концепата проширења и тачака проширења. Тачка проширења plug-in-а представља интерфејс компоненте преко кога та компонента комуницира са другим компонентама које проширују ту тачку проширења.

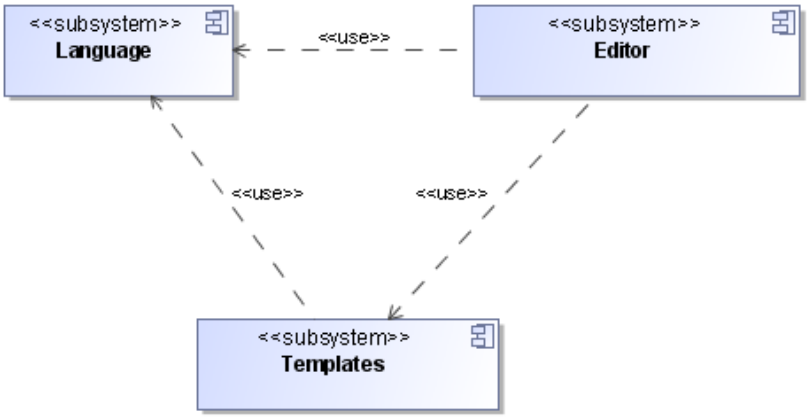
Приказ података о MARC 21 формату реализован је коришћењем XML технологије. Подаци о MARC 21 форматима смештени су XML документе формата из којих се учитавају и приказују у компоненти стабла на екранској форми.

Резултат треће итерације је RCP апликација система за каталогизацију који подржава све функционалности представљене на дијаграму случајева коришћења на слици 5.1.

5.3 АРХИТЕКТУРА СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

У овом одељку описана је архитектура основног едитора за каталогизацију који се генерише на основу спецификације у Xtext окружењу. Архитектура је приказана дијаграмом компоненти и дијаграмима пакета који се креирани коришћењем CASE алата MagicDraw верзија 16.0 [55].

Као што је описано у поглављу 3 у Xtext алату креирају се три пројекта за спецификацију језика и едитора за тај језик. На слици 5.2 приказан је дијаграм који илуструје зависност између три пројекта од којих је састављен систем за каталогизацију. Пројекти су представљени као подсистеми система за каталогизацију.



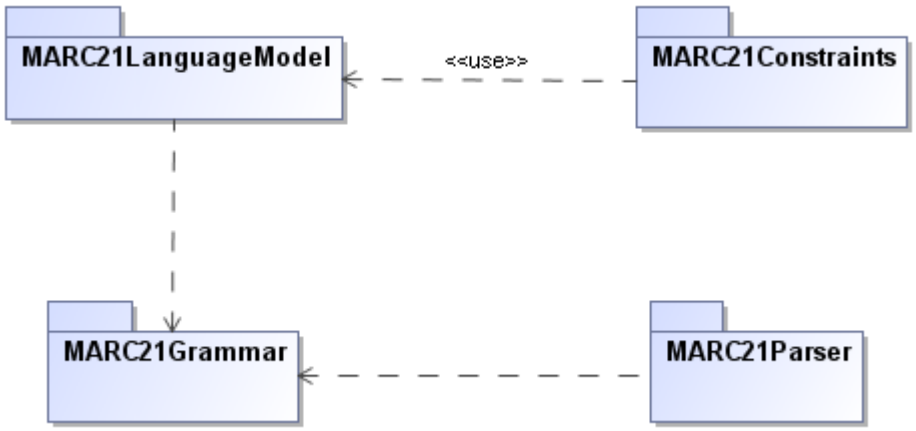
Слика 5.2 Архитектура система за каталогизацију

На дијаграму са слике 5.2 су приказани следећи подсистеми:

- *Language* - подсисем у ком је специфицирана граматика MARC 21 библиографског записа, проширења и ограничења над библиографским записом, као и EMF модел за језик специфициран граматиком и парсер за тај језик;
- *Templates* - подсистем за спецификацију темплејта за трансформацију библиографских записа који се креирају у едитору;

- *Editor* - подсистем за спецификацију и имплементацију едитора за каталогизацију, представља централни подсистем у реализацији система за каталогизацију.

На слици 5.3 приказан је дијаграм пакета који описује статички модел подсистема *Language* односно пројекта којим је специфициран језик за каталогизацију по MARC 21 формату. Основу овог система чини граматика језика која је представљена пакетом *MARC21Grammar*. Од граматике језика зависе, посредно или непосредно сви остали елементи подсистема за спецификацију језика. На основу граматике креира се модел језика представљен пакетом *MARC21LanguageModel* као и парсер за тај језик који је представљен пакетом *MARC21Parser*. Пакет *MARC21Constraints* садржи елементе за спецификацију ограничења над језиком, а за ту спецификацију користи се модел језика, односно пакет *MARC21LanguageModel*.



Слика 5.3 Архитектура подсистема за спецификацију језика

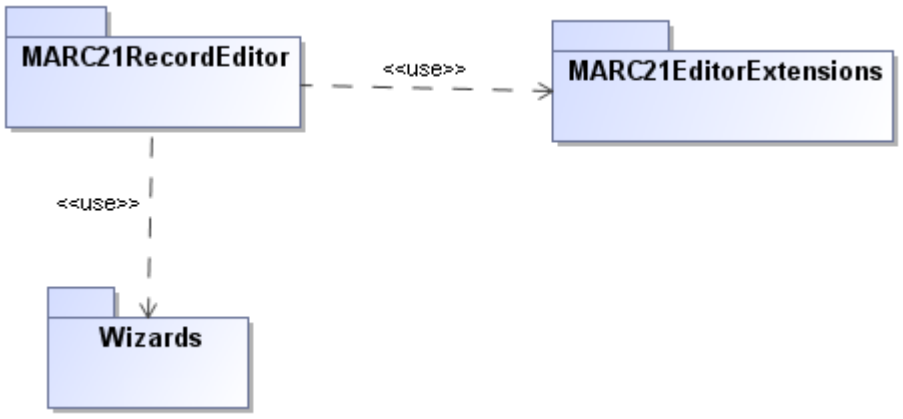
Подсистем за спецификацију темплејта за трансформацију библиографских записа, *Templates* садржи темплејте којима се библиографски запис може трансформисати у различите формате који се складиште у фајлове. У случају система за каталогизацију, овај подсистем се користи за спецификацију генерисања разних врста каталошких листића и XML докумената на основу библиографског записа који се уноси у едитору. Архитектуру овог подсистема чини скуп дефинисаних темплејта и генератор којим се извршавају ови темплејти.

Подсистем *Editor* заузима централно место у систему за каталогизацију и у њему је реализована целокупна функционалност система за каталогизацију

коришћењем остала два подсистема. Подсистем *Editor* је реализован у три корака односно три итерације које су описане у одељку 5.2:

1. аутоматско генерисање основног едитора на основу граматике специфициране у Xtext окружењу,
2. унапређење едитора спецификацијом његових карактеристика у Xtext окружењу,
3. имплементација додатних функционалности едитора у plug-in технологији.

На слици 5.3 приказана је архитектура основног едитора за каталогизацију, односно архитектура подсистема *Editor* након прве две итерације његове реализације. Архитектуру основног едитора чине три пакета. Пакет *MARC21RecordEditor* садржи едитор који је генерисан на основу граматике језика и који подржава бојење синтаксе и проверу исправности уноса. Пакетом *MARC21EditorExtensions* реализују се напредније функционалности едитора које се односе на помоћ приликом уноса података. Основни едитор садржи и пакет *Wizards* којим су представљени визарди за креирање нових докумената у едитору.



Слика 5.3 Архитектура основног едитора

5.4 МОДЕЛИРАЊЕ ДОДАТНИХ ФУНКЦИОНАЛНОСТИ СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

У одељку 5.3 описана је архитектура система за каталогизацију која је добијена генерисањем компоненти на основу граматике језика за унос библиографског записа. Спецификација граматике и генерисање система за каталогизацију реализовани су коришћењем софтверског алата Xtext.

У овом одељку дат је опис моделирања додатних функционалности система за каталогизацију који се развијају у трећој итерацији обједињеног процеса. Ово унапређење састоји се од имплементације додатних функционалности едитора које су представљене на дијаграму случајева коришћења на слици 5.1 у које спадају:

- приказ података о MARC 21 формату (случајеви коришћења *Show MARC 21 format view, Show bibliographic format view* и *Show holdings format view*),
- унос локацијских података (случај коришћења *Create holdings record*),
- експорт и импорт MARC 21 записа (случајеви коришћења *Store record*, *Select record*),
- библиотечко окружење (случај коришћења *Create record by processing type*) и
- приказ каталошких листића (случај коришћења *Show catalogue card*).

Опис моделирања унапређеног система за каталогизацију садржи опис архитектуре овог система који је дат у облику UML дијаграма пакета.

За сваку од наведених функционалности која представља унапређење система за каталогизацију дат је статички модел у облику дијаграма класа којима се реализује посматрана функционалност. Сваки дијаграм класа садржи класе из различитих пакета, а истакнити су само они елементи класе који су релевантни за реализацију посматране функционалност. Поред тога, за сваку од наведених функционалности дат је динамички модел у облику дијаграма секвенци којима се описује динамика извршавања тих функционалности.

5.4.1 Архитектура комплетног система за каталогизацију

На слици 5.4 приказан је дијаграм пакета који описује архитектуру унапређеног система за каталогизацију. Сви пакети који су приказани на дијаграму приказани су у оквиру подсистема коме припадају.

Подистем *Language* није промењен у односу на подсистем који је добијен генерсањем, и он се и у унапређеном систему за каталогизацију састоји од четири пакета. Основни пакет је *MARC21Grammar* у ком је описана граматика језика. На основу граматике креира се модел језика представљен пакетом *MARC21LanguageModel* као и парсер за тај језик који је представљен пакетом *MARC21Parser*. Пакет *MARC21Constraints* садржи елементе за спецификацију ограничења над језиком, а за ту спецификацију користи се модел језика, односно пакет *MARC21LanguageModel*. Пакети из овог подистема користе се од стране пакета из осталих подистема.

Подсистем *Templates* садржи један пакет у ком се налазе темплејти за трансформацију библиографских записа. За спецификацију темплејта користи се модел језика из пакета *MARC21LanguageModel*.

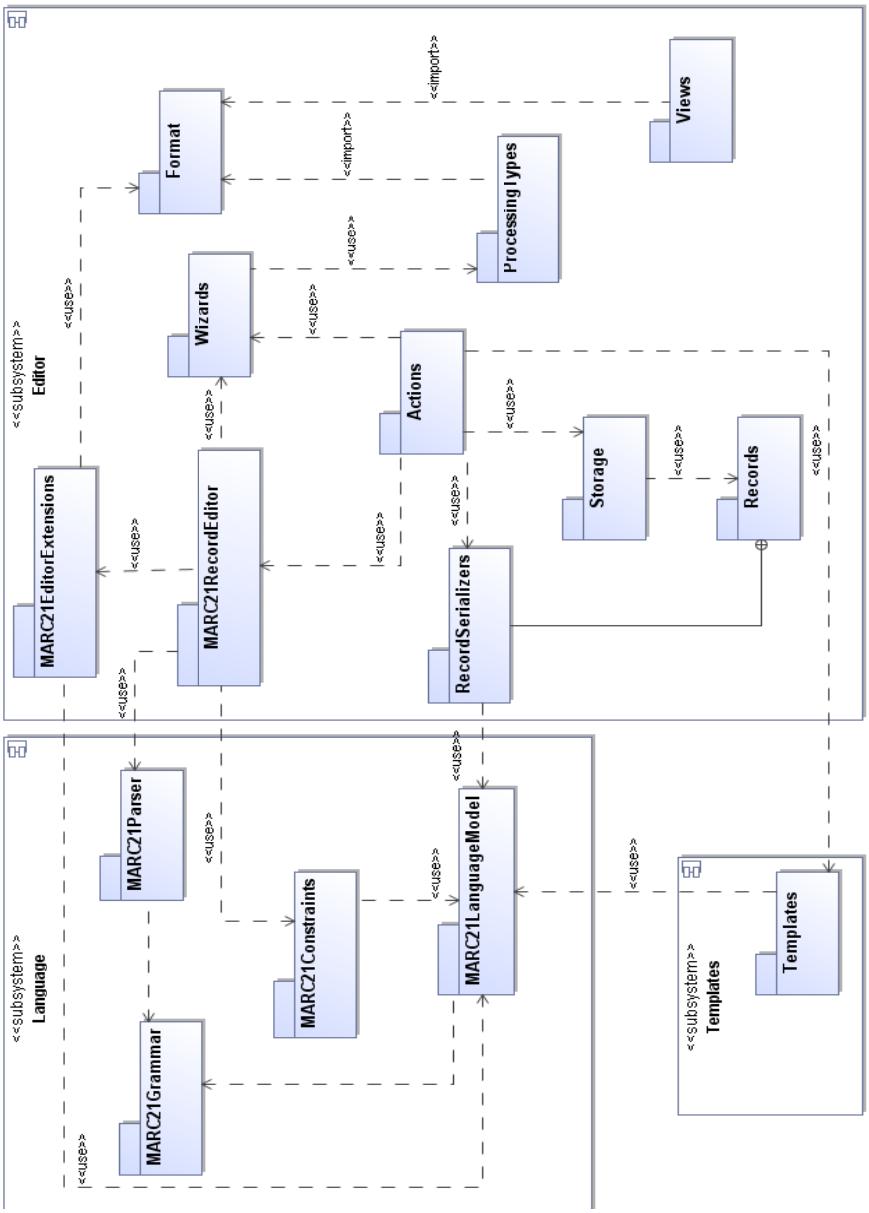
Подсистемом *Editor* реализоване су све функционалности система за каталогизацију. За реализацију ових функционалности користе се пакети из остала два подистема система за каталогизацију.

Пакети *MARC21RecordEditor, MARC21EditorExtensions* и *Wizards* који су део подистема *Editor* постоје и у архитектури основног едитора описаној у одељку 5.3. Остали пакети подистема *Editor* приказани на слици 5.4 представљају проширење основног едитора додатним функционалностима. Поред тога, пакет *Wizards* је проширен додатним визардима у унапређеном едитору.

Пакет *MARC21RecordEditor* у ком је реализован сам едитор за каталогизацију користи елементе MARC 21 језика за каталогизацију из подсистема *Language*. Едитор користи парсер за обраду текста који се уноси у едитор, као и ограничења језика на основу којих ће се генерисати поруке за корисника приликом едитирања.

Пакет *MARC21EditorExtensions* у ком се реализују додатне функционалности едитора као што су генерисање помоћи приликом уноса користи модел MARC 21 језика за каталогизацију.

Поред визарда за креирање новог документа за унос библиографског записа, пакет *Wizard* садржи и визард за креирање новог записа, на основу изабраног типа обраде. За реализацију овог визарда користи се пакет *ProcessingTypes* у ком се налазе објектни модел типа обраде и механизам за импорт типова обраде. Елементи пакета *Wizard* користе се у пакету *Actions* у ком се налазе све акције које корисник може да изврши у систему, а неке од њих захтевају покретање одговарајућих визарда. Поред тога, пакет *Actions* користи елементе пакета *MARC21RecordEditor* јер се већина акција које су специфициране у овом пакету извршава над едитором за каталогизацију.



Слика 5.4 Архитектура унапређеног система за каталогизацију

Постоје још и акције које се односе на експорт и импорт библиографских записа из едитора што је реализовано у пакету *Storage*. За експорт и импорт записа из едитора користи се објектни модел записа (описан у одељку 4.6.1) који је садржај пакета *Records*. Серијализација објектног модела записа у разне облике његове репрезентације, као и креирање објектног модела на основу разних врста репрезентације записа обавља се у пакету *RecordSerializers*. Серијализација објектног модела записа користи се у акцијама експорта и импорта библиографских записа. Један од облика у који се објектни модел записа серијализује и из кога се креира је и модел дефинисан у пакету *MARC21LanguageModel* из подистема *Language*. Пакет *RecordSerializers* је угњеждени пакет пакета *Records* (што је графички приказано на слици 5.4).

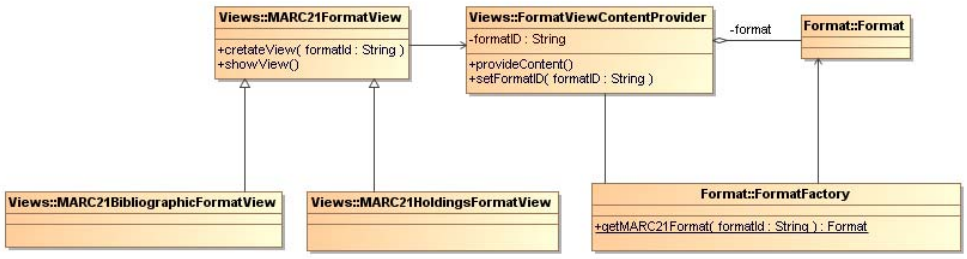
Пакет *Actions* садржи и реализацију функционалности приказа каталошких листића за шта се користе елементи пакета *Templates* из подсистема *Templates*.

Поред свих наведених пакета, у реализацији додатних функционалности едитора учествују и специјални елементи за приказ података на екранској форми који су специфицирани пакетом *Views*. Класама које припадају овом пакету реализован је приказ података о библиографском формату и у њима се користи објектни модел библиографског формата (описан у одељку 4.4) који је садржај пакета *Format*. Елементи пакета *Format* се користе и у формирању типова обраде, односно у елементима пакета *ProcessingType*.

Пакет *MARC21EditorExtensions* је део основног едитора и у њему се специфицира помоћ за унос података у едитору. За генерисање помоћи приликом уноса користе се информације из спецификације библиографског формата (списак поља, потпоља, шифре за индикаторе и сл.). У унапређеном систему за каталогизацију пакет *MARC21EditorExtensions* користи спецификацију формата која је садржај пакета *Format.*

5.4.2 Моделирање приказа података о MARC 21 формату

На слици 5.5 приказан је дијаграм класа које учествују у реализацији функционалности приказа података о MARC 21 формату. За сваку класу на овом дијаграму приказан је њен назив, пакет коме припада и скуп атрибута и операција релевантних за реализацију приказа библиографског формата.



Слика 5.5 Дијаграм класа за приказ библиографског формата

У оквиру система за каталогизацију реализована су два погледа за приказ MARC 21 формата, а то су поглед у ком се приказују подаци MARC 21 библиографског формата и поглед у ком се приказују подаци MARC 21 формата везани за обраду локацијских података. Поглед у ком се приказује библиографски формат реализован је класом *MARC21BibliographicFormatView*. Поглед за приказ оног дела формата који се односи на обраду локацијских података реализован је класом *MARC21HoldingsFormatView*. Обе поменуте класе представљају специјализацију класе *MARC21FormatView* која је оквир за креирање приказа MARC 21 формата.

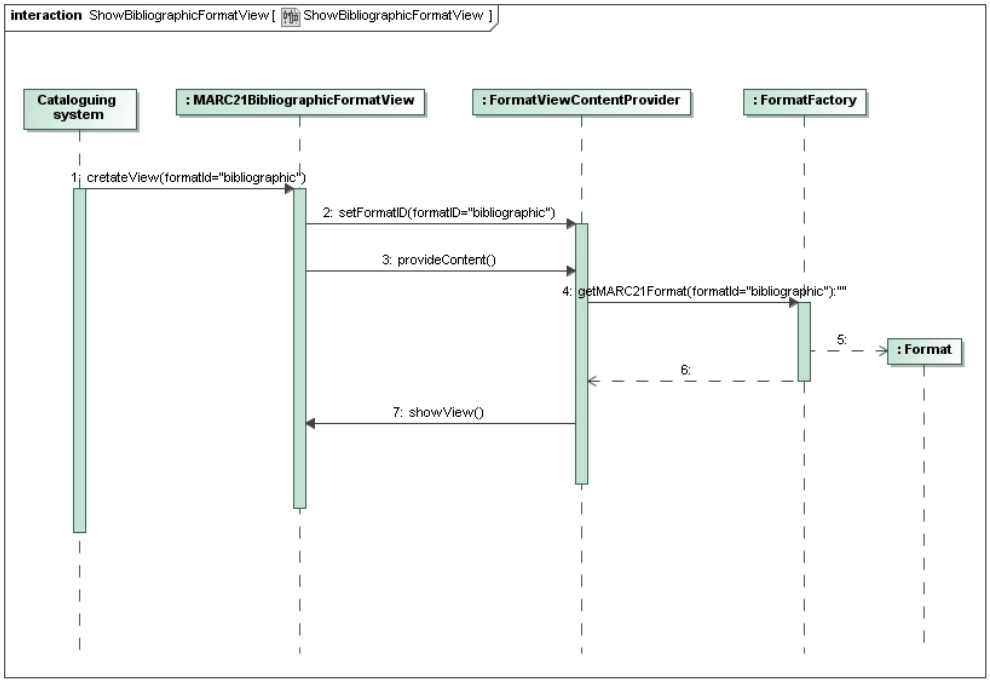
Класа *MARC21FormatView* има две операције. Операција *createView(formatId:String)* служи за креирање погледа, а операцијом *showView()* креирани поглед се приказује на екранској форми.

Операција *createView(formatId:String)* као улазни параметар узима *id* формата за који се креира поглед. У тренутној верзији система за каталогизацију овај параматер може имати две вредности, то су *„bibliographic“* и *„holdings“*. Уколико се ова операција позове са параметром *„bibliographic“* инстанцира се класа за креирање погледа за библиографске податке, односно ако се позове са параметром *„holdings“* инстанцира се класа за приказ формата за унос локацијских података.

За прибављање садржаја који ће се приказивати на екранској форми задужена је класа *FormatViewContentProvider*. Њена операција *provideContent()* прибавља податке о MARC 21 формату који ће се приказивати. Операцијом *setFormatID(formatID:String)* поставља се вредност атрибута *formatID* који даје информацију о томе који формат треба учитати.

Подаци о MARC 21 форматима учитавају се из XML докумената операцијом *getMARC21Format(formatID:String)* којом се креира инстанца класе *Format*.

Параметар операције *formatID* одређује из ког XML документа се учитавају подаци.

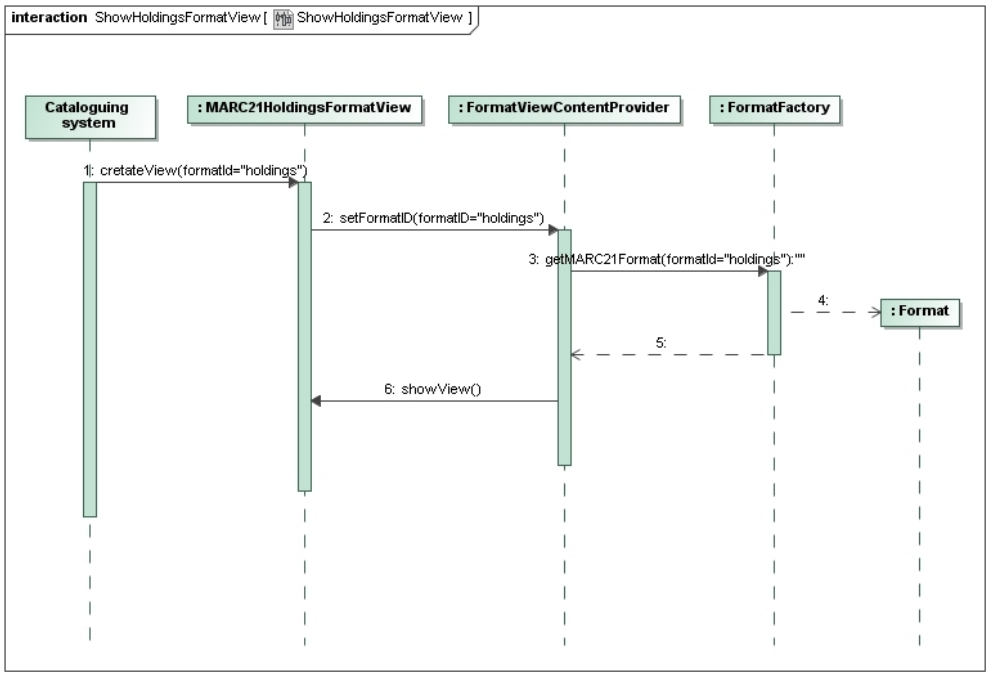


Слика 5.6 Дијаграм секвенци *ShowBibliographicFormatView*

На слици 5.6 приказан је дијаграм секвенци *ShowBibliographicFormatView* који описује динамику креирања погледа за приказ MARC 21 библиографског формата (случај коришћења *Show bibliographic format*). Формирање приказа покреће систем за каталогизацију у тренутку када је потребно на екранској форми приказати податке о формату, а то је у већини случајева приликом покретања едитора за унос библиографских записа. Формирање приказа започиње позивом операције *createView* класе *MARC21BibliographicFormatView* којој се прослеђује параметар *„bibliographic“* јер се креира приказ библиографског формата. Следећа порука је постављање ID вредности формата у класи *FormatViewContentProvider* која је задужена за прибављање података о формату. Након што је постављена вредност за ID формата позива се операција *provideContent()* исте класе и тиме започиње процес учитавања података о формату. Следећи корак је да објекат класе *FormatViewContentProvider* позове операцију *getMARC21Format („bibliographic“)* која ће учитатати податке о MARC 21 формату. Како је прослеђени параметар једнак „*bibliographic*“ учитаће се XML документ библиографског формата у објекат класе *Format* и проследити назад као одговор класи *FormatViewContentProvider*. Након тога остаје још само да се

прикаже прибављени садржај позивом операције *showView* класе *MARC21BibliographicFormatView*.

На слици 5.7 приказан је дијаграм секвенци *ShowHoldingsFormatView* који описује процес креирања приказа података о MARC 21 формату за обраду локацијских података (случај коришћења *Show holdings format view*). Овај процес је веома сличан приказу MARC 21 библиографских података, а разлика се састоји у томе да се у случају локацијских података за креирање погледа користи класа *MARC21HoldingsFormatView* и да се као параметар *formatId* у одговарајућим порукама прослеђује *„holdings“*.



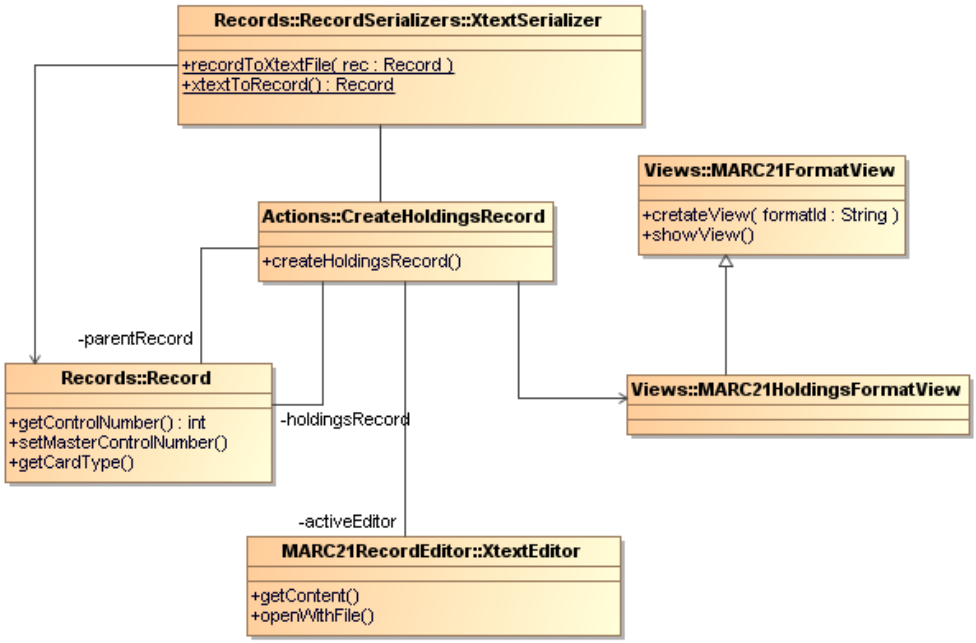
Слика 5.7 Дијаграм секвенци *ShowHoldingsFormatView*

5.4.3 Моделирање уноса локацијских података

На слици 5.8 приказан је дијаграм класа којима се реализује унос локацијских података у систему за каталогизацију. Унос локацијских података реализован је према правилима које прописује MARC 21 стандард тако што се за сваки библиографски запис креира нови MARC 21 запис и то запис за локацијске податке (*holdings data*).

Централна класа за реализацију акције уноса локацијских података је класа *CreateHoldingsRecord* из пакета *Actions*. Она има операцију *createHoldingsRecord()* којим се покреће процес креирања записа за локацијске податке.

Класа *CreateHoldingsRecord* је повезана са класом *XtextEditor* из пакета *MARC21RecordEditor* која представља компоненту едитора за унос података у запис. Класа *MARC21RecordEditor* има две операције то су операција *getContent()* којом се преузима садржај едитора, и операција *openWithFile()* којом се едитор отвара за специфичан фајл чији садржај се појављује у едитору.



Слика 5.8 Дијаграм класа за унос локацијских података

Са друге стране, класа *CreateHoldingsRecord* повезана је и са класом *XtextSerializer* из пакета *RecordSerializers*. Класа *XtextSerializer* садржи операције за трансформацију MARC 21 записа из објектног модела у Xtext репрезентацију и обрнуто. Трансформација објектног модела MARC 21 записа у Xtext репрезентацију реализује се операцијом *recordToXtextFile()*. Ова трансформација састоји се у томе да се елементи записа из објектног модела упишу у фајл са екстензијом *marc21* према правилима која су специфицирана Xtext граматиком библиографског записа. Трансформација записа у супротном смеру, односно из Xtext репрезентације којом је запис представљен у едитору у објектни модел реализује се операцијом *xtextToRecord()*. Ова операција трансформише меморијску структуру текста у генерисаном едитору, која је описана у одељку 3.6 у објектни модел записа.

У креирању записа за локацијске податке користи се и класа *Record*. У реализацији ове функционалности користиће се две операције класе *Record*. Једна је операција за преузимање контролног броја записа (садржај контролног поља 001) – *getControlNumber()*. Друга операције је операција *setMasterControlNumber()* којом се запису за локацијске податке додељује контролни број библиогарфског записа на који се односи ти локацијски подаци (садржај контролнох поља 004 у запису за локацијске податке).

Преостале две класе на дијаграму са слике 5.8 су класе за креирање погледа за приказ MARC 21 формата за локацијске податке.

На слици 5.9 дат је дијаграм секвенци *EnterHoldingsData* којим је илустрован унос локацијских података који је представљен случајем коришћења *Create holdings record*.

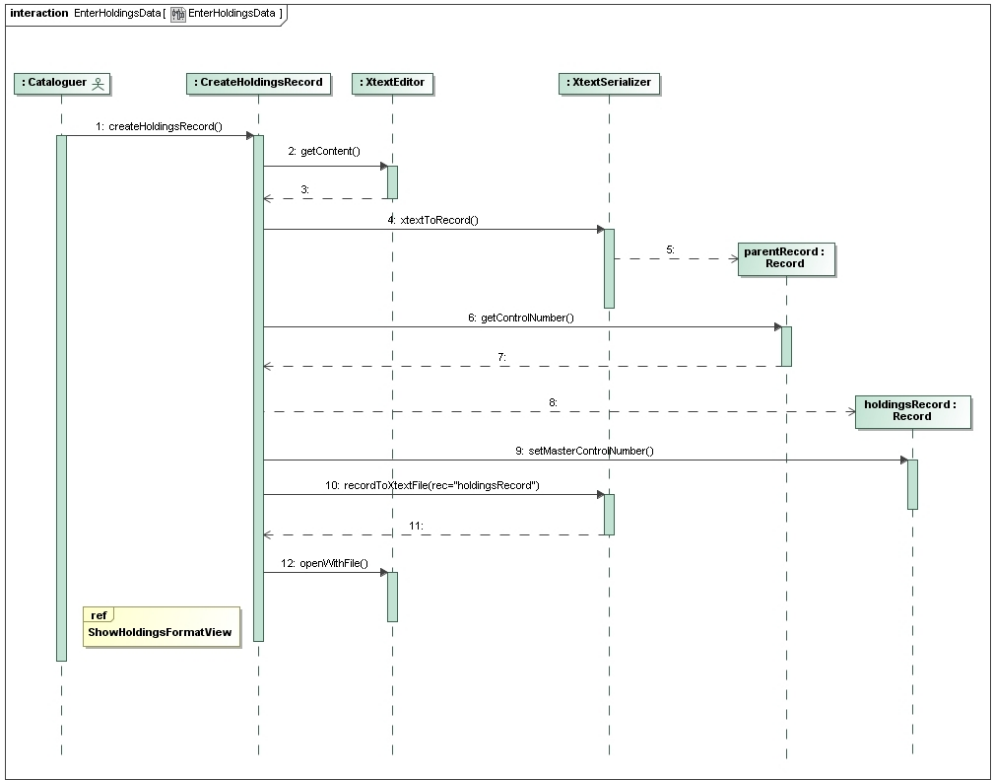
Акцију за унос локацијских података покреће библиотекар, односно каталогизатор позивом операције *createHoldingsRecord()* класе *CreateHoldingsRecord*. Позивом ове операције покреће се процес креирања записа за локацијске податке који ће бити повезан са библиографским записом који се тренутно обрађује у едитору. Повезивање ова два записа релизовано је преко контролног броја.

Први корак је да се из едитора преузме садржај позивом операције *getContent()*. Тај преузети садржај је у Xtext формату и треба га трансформисати у објектни модел што се извршава позивом операције *xtextToRecord()* класе *XtextSerializer*. На овај начин креиран је објекат класе *Record* који представља библиографски запис који се тренутно обрађује у едитору и додељено му је име *parentRecord*. Овом објекту библиографског записа треба још преузети контролни број позивом операције *getControlNumber()*.

Следећи корак је креирање нове инстанце класе *Record* која ће представљати запис за локацијске податке. Овај корак је реализован поруком *createHoldingsRecord* којом је креиран објекат *holdingsRecord*. Објекту *holdingsRecord* се додељује број матичног библиографског записа операцијом *setMasterControlNumber()*.

Када је креиран објектни модел записа за локацијске податке потребно га је отворити за едитирање, односно поставити као садржај едитора. Да би се ово реализовало прво се запис за локацијске податке из објектног модела трансформише у Xtext фајл, односно фајл са екстензијом *marc21*који задовољава правила специфициране граматике, а затим се тај фајл отвара у едитору позивом операције *openWithFile()* класе *XtextEditor*.

На крају још треба отворити поглед за приказ MARC 21 формата за локацијске податке који је описан дијаграмом секвенци *ShowHoldongsFormatView* чиме би се кориснику обезбедила помоћ за унос локацијских података.



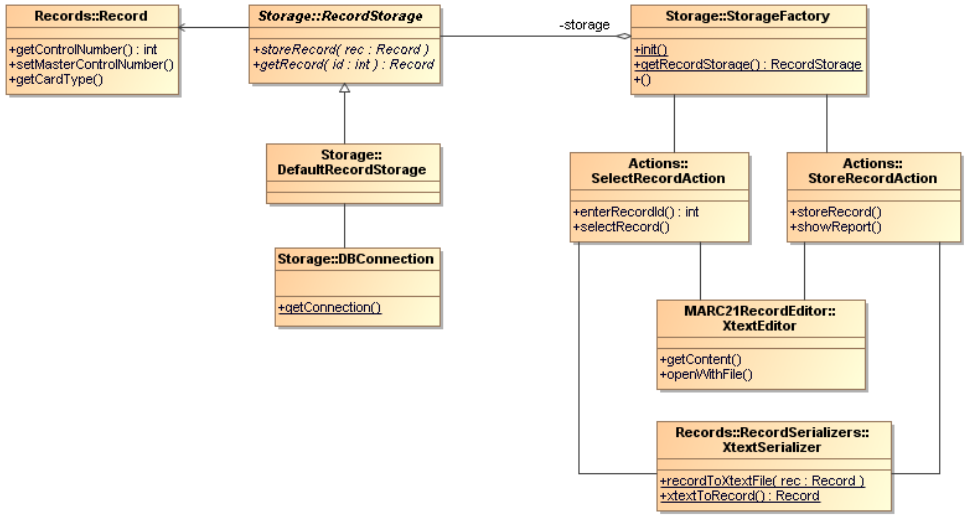
Слика 5.9 Дијаграм секвенци *EnterHoldingsData*

Резултат описане секвенце акција је креирање записа за локацијске податке и његово отварање. Даља обрада овог записа се не разликује од обраде било ког библиографског записа, осим што запис за локацијске података има сопствени скуп поља и потпоља а веза са библиографским записом је остварена преко контролног броја.

5.4.4 Моделирање експорта и импорта MARC 21 записа

На слици 5.10 приказан је дијаграм класа којима се реализује процес експорта записа из едитора и импорта записа у едитор. Експорт записа из едитора састоји се у томе да се на позив одређене акције запис који је креиран у едитору складишти у неку базу података, индекс за претраживање или фајл. Импорт записа у едитор је обрнути процес и представља учитавање записа из неког складишта у едитор за каталогизацију. Импорт записа ће бити приказан

као селекција записа на основу његовог ID-а и учитавање селектованог записа у едитор.



Слика 5.10 Дијаграм класа експорта и импорта MARC 21 записа

За складиштење и учитавање записа реализоване су две акције у оквиру пакета *Actions*. Класом *StoreRecordAction* реализована је акција за складиштење записа. Она има две операција, то су операција за покретање процеса складиштења *storeRecord()* и операција за приказ извештаја о успешности складиштења *showReport()*. Класом *SelectRecordAction* представљена је акција селекције записа на основу ID броја. Ова класа има две операције, то су операција *enterRecordId()* за унос ID броја записа и операција *selectRecord()* којим се покреће процес учитавања записа на основу ID броја.

Обе наведене акције за своју реализацију користе класу *XtextEditor* за проступ садржају едитора и класу *XtextSerialuzer* за трансформацију записа из објектног модела у Xtext репрезентацију и обрнуто. Поред тога, користи се и класа за комуникацију са складиштем података *StorageFactory* из пакета *Storage*.

Архитектура система за каталогизацију омогућава да се складиштење и учитавање записа реализује у оквиру библиотечког система у који се систем за каталогизацију интегрише. Библиотечки систем треба да креира класу која ће наследити апстрактну класу *RecordStorage* и да у тој класи имплементира комуникација са конкретним складиштем (базом података, индексом за претраживање и слично).

Класа *RecordStorage* има две апстрактне операције:

- *storeRecord(rec:Record)* – складиштење записа који је креиран у едитору, позиваће се на одређену акцију за складиштење записа;
- *getRecord(id:int)* – учитавање библиографског записа у едитор, позваће се на акцију учитавања записа на основу његовог идентификационог броја

Имплементација ове две операције зависи од интеграције система за каталогизацију и његовог повезивања са конкретном базом података. Библиографски запис се у систему складишти односно учитава у облику инстанце класе *Record* из пакета *Records* која је објектни модел записа.

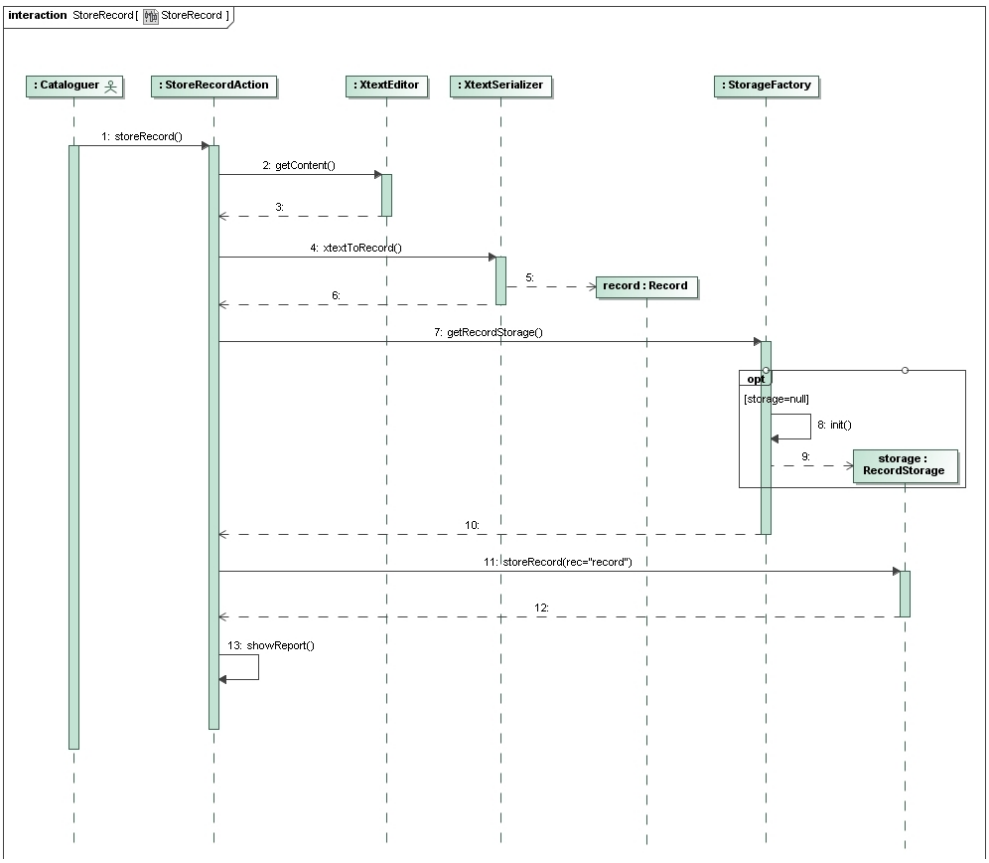
У оквиру самог система за каталогизацију реализована је једна подразумевана имплементација складиштења која ће се користити у случају да нема проширења, односно да се систем за каталогизацију користи самостално. Та подразумевана имплементација проширења представљена је класом *DefaultRecordStorage* и она за реализацију складиштења записа у базу користи класу *DBConnection* у којој је специфицирана конекција на конкретну базу података.

Улога класе *StorageFactory* у систему је да иницијализује инстанцу класе која представља имплементацију апстрактне класе *RecordStorage*. Ова иницијализација се обавља операцијом *init()*. Основна идеја реализације операције *init()* састоји се у томе да се прво провери да ли има проширења система са неком имплементацијом класе *RecordStorage* и ако има прави се инстанца те класе која и смешта у атрибут *storage*, који представља везу агрегације са класом *RecordStorage*. Овај атрибут ће се затим враћати као резултат операције *getRecordStorage()*, и користиће се за складиштење и учитавање записа.

Динамички модел складиштења записа

На слици 5.11 приказан је дијаграм секвенци *StoreRecord* који представља динамички модел складиштења записа у систему за каталогизацију, односно случаја коришћења *Store record*.

Складиштење записа покреће каталогизатор позивом операције *storeRecord()* класе *StoreRecordAction*. Затим се преузима садржај едитора позивом операције *getContent()* и тај садржај се враћа објекту класе *StoreRecordAction*. Овај преузети садржај је у Xtext формату и потребно га је трансформисати у објектни модел, односно инстанцу класе *Record* што се реализује операцијом *xtextToRecord* из класе *XtextSerializer*. Након ове трансформације креиран је објекат *record*, инстанца класе *Record* који треба да се складишти, односно експортује из едитора.



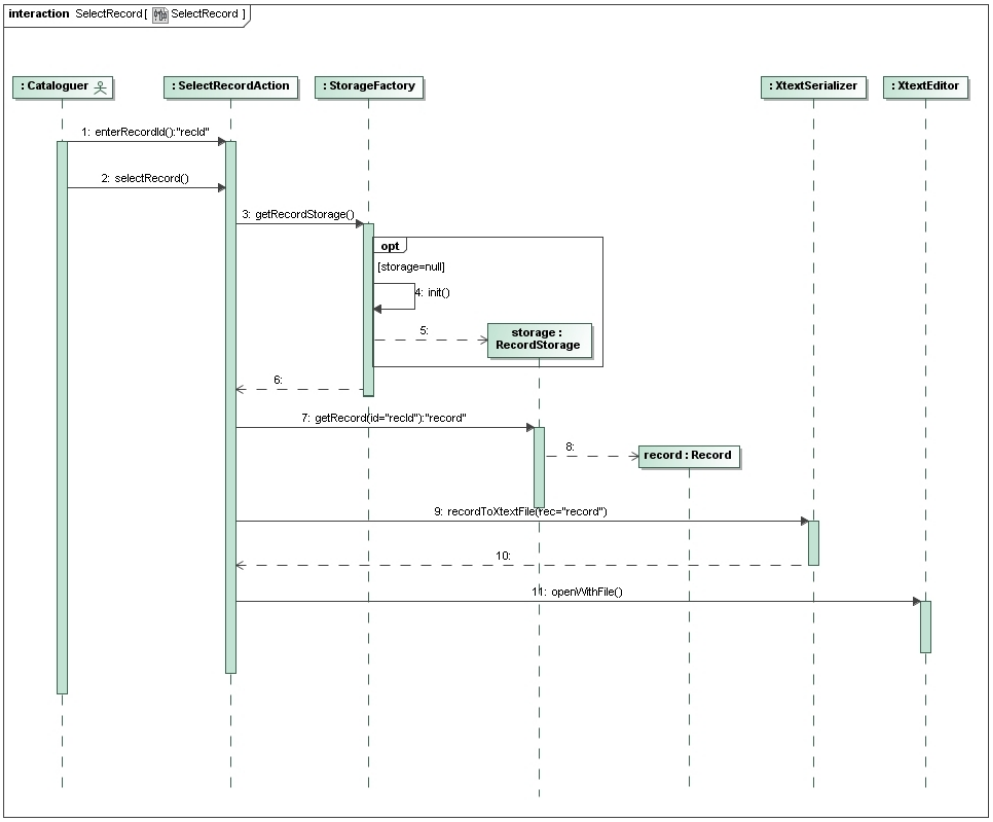
Слика 5.11 Дијаграм секвенци *StoreRecord*

Процес складиштења записа почиње операцијом *getRecordStorage()* класе *StorageFactory*. Уколико објекат *storage* у ком се чува инстанца класе *RecordStorage* преко које се врше складиштење записа није креиран, позива се операција *init()* у којој се врши иницијализација складишта. О операцији *init()* креира се објекат *storage* на основу проширења система, уколико постоје, или иницијализацијом класе *DefaultRecordStorage* уколико нема проширења. Новокреирани или постојећи објекат *storage* враћа се као резултат операције *getRecordStorage* објекту класе *StoreRecordAction*.

На крају још преостаје да се изврши складиштење позивом операције *storeRecord()* објекта *storage*. Као параметар ове операције прослеђује се објекат *record* који је добијен трансформацијом Xtext садржаја едитора. На крају се кориснику приказује порука о успешности складиштења записа операцијом *showReport()* класе *StoreRecordAction*.

Динамички модел учитавања записа

На слици 5.12 приказан је дијаграм секвенци *SelectRecord* који представља динамички модел селекције записа преко ID броја и његово учитавање у едитор. Овим дијаграмом илустрован је процес импорта записа у систем за каталогизацију.



Слика 5.12 Дијаграм секвенци *SelectRecord*

Процес селексије записа започиње каталогизатор уносом ID броја записа за селекцију, односно позивом операције *enterRecordID()*. Као резултат ове операције добија се вредност *recId* на основу које ће се извршити учитавање записа. Након уноса ID броја записа, каталогизатор покреће процес учитавања записа позивом операције *selectRecord()* класе *SelectRecordAction*.

Први корак приликом учитавања записа је прибављање складишта из класе *StorageFactory*. Исто као приликом складиштења записа, које је описано у претходном одељку, проверава се да ли је атрибут *storage* креиран и ако није креира се у операцији *init()*. Новокреирано или постојеће складиште се враћа објекту класе *SelectRecordAction* као резултат операције *getRecordStorage()*.

У следећем кораку се за објекат *storage* позива операција *getRecord(recId)* која враћа запис из складишта и смешта га у објекат *record*. Добијени објекат се затим трансформише у Xtext репрезентацију, односно у Xtext фајл операцијом *recordToXtextFile()* класе *XtextSerializers*.

У последњем кораку се фајл који је добијен као резултат трансформације објекта *record* отвара у едитору позивом операције *openWithFile()* класе *XtextEditor*.

5.4.5 Моделирање библиотечког окружења

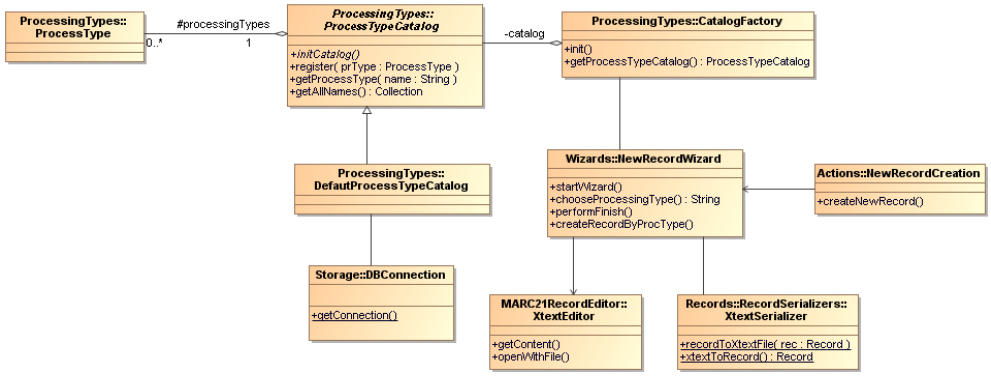
Библиотечко окружење представља значајан аспект библиотечких информационих система и односи се на кастомизацију система, односно прилагођавање система специфичним потребама библиотекара. У случају каталогизације библиотечко окружење подразумева коришћење различитих типова обраде за каталогизацију. Типом обраде дефинише се скуп елемената записа, односно поља и потпоља која се појављују у запису који се креира према том типу обраде. Поред тога, тип обраде дефинише и подразумеване (*default*) вредности за поједине елементе записа.

У случају система за каталогизацију, реализован је аспект библиотечког окружења у ком се приликом креирања новог записа бира тип обраде по ком ће се креирати запис. Тип обраде се бира из неког постојећег скупа типова обраде. На основу изабраног типа обраде креира се иницијални запис и тај запис се отвара у едитору и може се даље обрађивати.

На слици 5.13 приказан је дијаграм класа којима се у систему за каталогизацију реализује креирање новог записа на основу изабраног типа обраде. Класом *NewRecordCreation* из пакета *Actions* реализована је акција којом се покреће процес креирања новог записа на основу типа обраде. Ова класа има једну операције *createNewRecord()* коју ће корисник позвати да би креирао нови запис.

Избор типа обраде реализован је у облику визарда који је представљен класом *NewRecordWizard* из пакета *Wizards*. Ова класа садржи четири операције:

- *startWizard()* – покреће визард за избор типа обраде,
- *chooseProcessingType()* – реализује избор типа обраде,
- *performFinish()* – позива се приликом завршетка рада визарда и
- *createRecordByProcType()* – покреће процес креирања иницијалног записа на основу изабраног типа обраде.



Слика 5.13 Дијаграм класа библиотечког окружења

За реализацију својих операција класа *NewRecordWizard* користи класу *XtextSerializers* за трансформацију објектног модела записа у Xtext репрезентацију и класу *XtextEditor* за отварање новог записа у едитору. За прибављање свих постојећих типова обраде у систему користи се класа *CatalogFactory* из пакета *ProcessingTypes*.

Слично као у примеру експорта и импорта записа, учитавање типова обраде у великој мери зависи од система у који се систем за каталогизацију интегрише. Због тога је архитектура система за каталогизацију реализована тако да се приликом његове интеграције и овај део функционалности може прилагодити систему у који се интегрише. Пролагођавање се састоји у томе да се имплементира операција која врши иницијализацију листе доступних типова обраде у систему.

Класа *ProcessTypeCatalog* представљена је колекција типова обраде који су доступни у систему. Ова класа има једну апстрактну операцију, то је операција *initCatalog()* у којој се врши иницијализација колекције типова обраде. Ова операција се може имплементирати на различите начине у различитим софтверским системима. Наиме, приликом интеграције система за каталогизацију потребно је креирати класу која наслеђује апстрактну класу *ProcessTypeCatalog* и имплементирати њену апстрактну операцију *initCatalog()* у зависности од базе података која се користи. Имплементација операције *initCatalog()* састоји се од учитавања типова обраде из базе података, мапирање сваког учитаног типа обраде на објекат класе *ProcessingType* и његово смештање у каталог типова обраде.

Класа *ProcessTypeCatalog* садржи и помоћне операције за рад са типовима обраде из каталога типа обраде. Помоћу операције *register(procType:ProcessingType)* додаје се нови тип обраде у каталог типова обраде. Операција *getProcessType()* враћа тип обраде на основу његовог

имена, операција *getAllNames()* враћа имена свих типова обраде који су регистровани у каталогу.

Као и у случају експорта и импорта записа у оквиру система за каталогизацију реализована је и једна подразумевана имплементација класе *ProcessTypeCatalog* која је представљена класом *DefaultProcessingTypeCatalog*. Ова класа ће се користити у случају да не постоји друга имплементација. У класи *DefaultProcessTypeCatalog* користиће се специфична комуникација са базом која је представљена класом *DBConnection*.

Улога класе *CatalogFactory* је иста као и улога класе *StorageFactory* у експорту и импорту записа. И овде се операцијом *init()* врши провера да ли је систем проширен са одговарајућом класом која наслеђује апстрактну класе *ProcessTypeCatalog*. Уколико постоји одговарајуће проширење, инстанцира се класа проширења. У случају да не постоји проширење, инстанцира се класа *DefaultProcessTypeCatalog*. Класа која је у овој операцији инстанцирана биће резултат операције *getProcessTypeCatalog()* сваки пут када се у систему за каталогизацију затражи каталог типова обраде.

Динамички модел креирања новог записа на основу типа обраде

Динамички модел креирања новог записа на основу изабраног типа обраде (случај коришћења *Create record by process type*) представљен је дијаграмом секвенци *NewRecordByProcessType* који је приказан на слици 5.14.

Први корак у креирању новог записа је позив операције *createNewRecord()* класе *NewRecordCreation*, после чега се покреће визард за избор типа обраде позивом операције *startWizard()* класе *NewRecordWizard*.

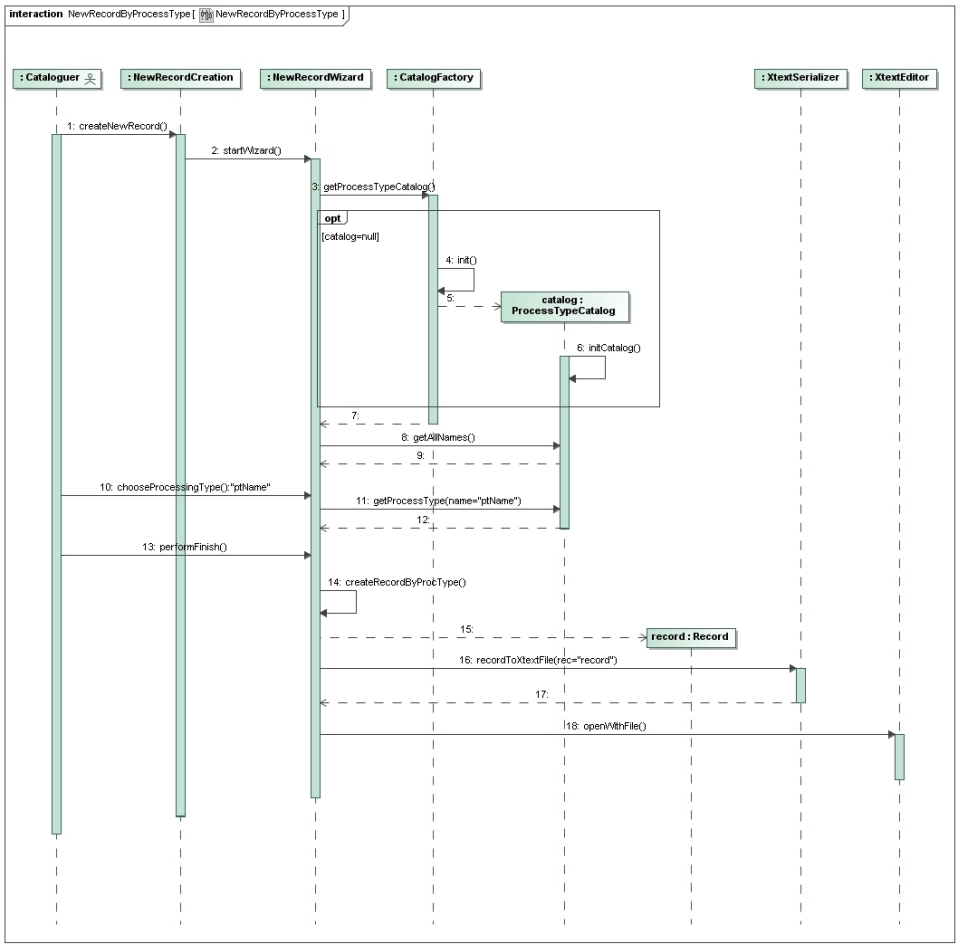
Прва операција која се позива приликом креирање визарда односи се на учитавање постојећих типова обраде и то је операција *getProcessTypeCatalog()* класе *CatalogFactory*. Уколико се први пут позова ова операција вредност објекта *catalog* је *null* и позваће се операција *init()* класе *CatalogFactory*. У овој операцији ће се извршити провера да ли постоји проширење посматраног система и на основу тога ће се креирати одговарајући објекат *catalog*. Прва операција која се позива приликом креирања овог објекта је *initCatalog()* у којој ће се скуп типова обраде учитати у меморију система.

За креирање визарда користиће се имена свих типова обраде новокреираног или постојећег објекта *catalog*. Листа свих типова обраде добија се позивом операције *getAllNames()*. Добијена листа имена типова обраде приказује се у визарду и каталогизатор бира један тип обраде позивом операције *chooseProcessingType()*. Резултат ове операције је назив типа обраде који се смешта у променљиву *ptName*. На основу назива изабраног типа обраде из

каталога типа обраде узима се изабрани тип обраде операцијом *getProcessType(ptName)*.

Након избора типа обраде каталогизатор позива операцију за завршетак визарда *performFinish()* и на тај начин покреће процес креирања новог записа. Операција *createRecordByProcType()* креира нови запис на основу параметара који су специфицирани у изабраном типу обраде. Резултат ове операције је инстанца класе *Record* која је представљена објектом *record*.

Након што је креиран одговарајући објекат којим је представљен запис, потребно је тај запис отворити у едитору. Ово се постиже у два корака, прво се објекат *record* трансформише у Xtext фајл операцијом *recordToXtextFile()* а затим се креирани фајл отвара у едитору операцијом *openWithFile()*.

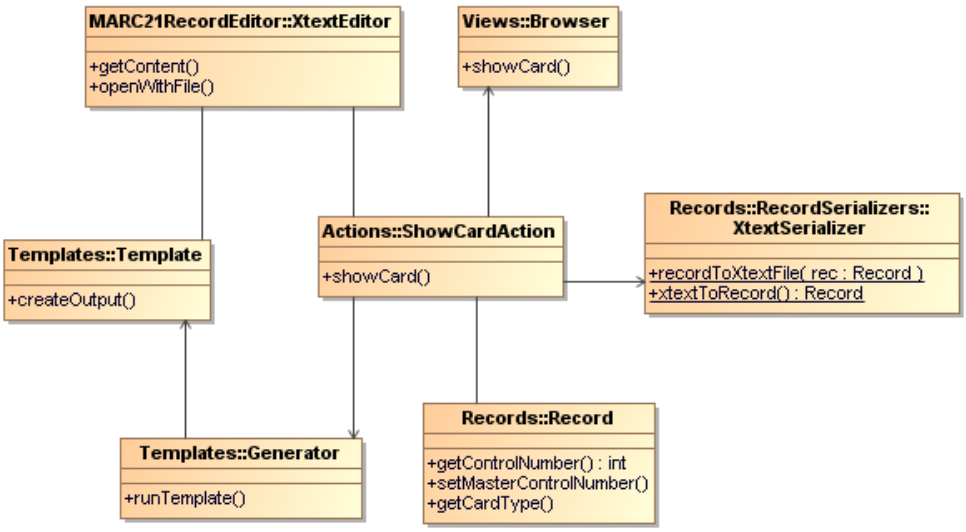


Слика 5.14 Дијаграм секвенци *NewRecordByProcessType*

5.4.6 Моделирање приказа каталошких листића

На слици 5.15 приказан је дијаграм класа којима се реализује приказ каталошког листића за библиографски запис који се креира у едитору. Акција креирања листића представљена је класом *ShowCardAction* из пакета *Actions* а покреће се позивом операције *showAction()*.

Класа *ShowCardAction* за креирање и приказ листића користи класу *XtextEditor* за преузимање записа из едитора, класу *XtextSerializer* за трансформацију записа, класу *Record* за преузимање информације о типу листића који се генерише, класу *Generator* за покретање темплејта за трансформацију и класу *Browser* за приказ каталошког листића.



Слика 5.15 Дијаграм класа за приказ каталошког листића

Један од података у запису је и врста грађе публикације на коју се запис односи. На основу врсте грађе одређује се и тип каталошког листића који ће се креирати за запис. Тип каталошког листића за запис добија се операцијом *getCardType()* класе *Record*.

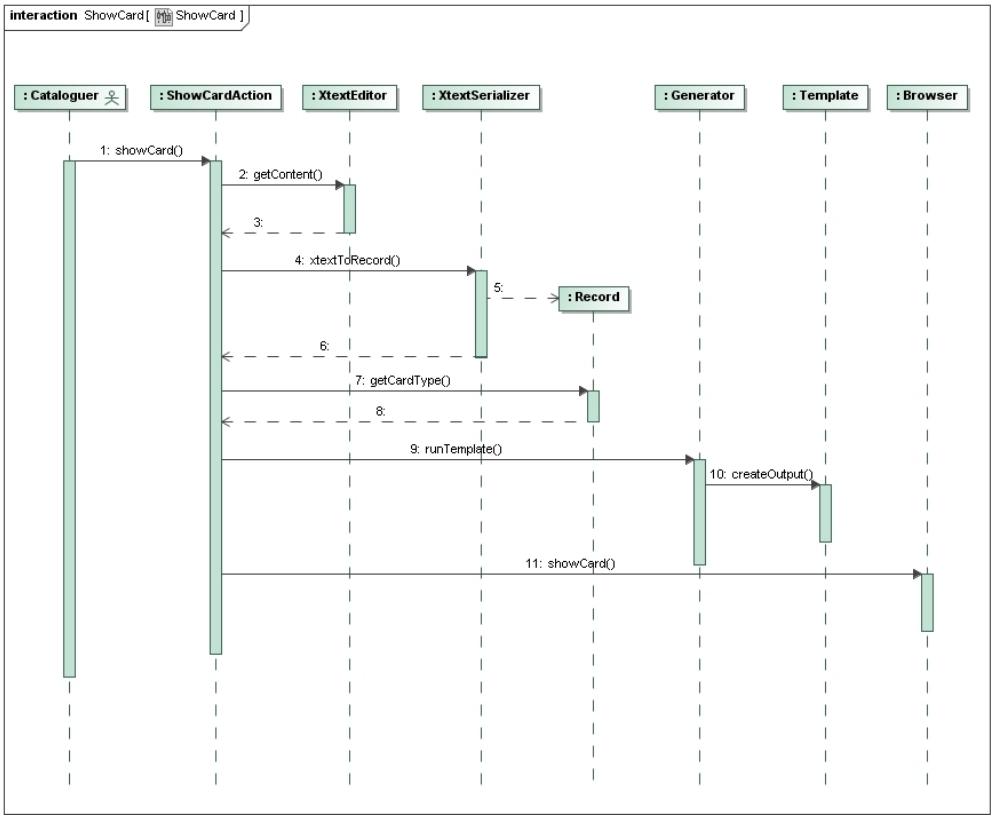
Операцијом *runTemplate()* класе *Generator* из пакета *Templates* покреће се процес генерисања каталошког листића на основу садржаја едитора. Генерисање каталошког листића извршава се на основу темплејта који је представљен класом *Template* из пакета *Templates*. Класа *Template* има једну операцију *createOutput()* којом се креира излаз у облику фајла који ће садржати каталошки листић.

Класом *Browser* из пакета *Views* представљен је поглед за приказ *html* садржаја и њеном операцијом *showCard()* приказује се каталошки листић на екрану.

Динамички модел приказа каталошких листића

На слици 5.16 приказан је дијаграм секвенци *ShowCard* који представља динамички модел креирања и приказа каталошког листића у систему за каталогизацију (случај коришћења *Show catalogue card*).

Каталогизатор покреће процес креирања каталошког листића позивом операције *showCard()* класе *ShowCardAction*. Прва порука која се извршава у процесу креирања листића је преузимање садржаја едитора операцијом *getContent()* класе *XtextEditor*. Следећи корак је трансформација садржаја едитора у објектни модел записа, операцијом *xtextToRecord()* класе *XtextSerializers*. Резултат ове операције је објекат класе *Record* из кога се преузима тип листића операцијом *getCardType()*.



Слика 5.16 Дијаграм секвенци *ShowCard*

Након што је преузет садржај едитора и тип листића који треба да се креира позива се операција *runTemplate()* класе *Generator* која затим позива операцију

createOutput() класе *Template*. Операцијом *createOutput()* генерише се излаз који представља каталошки листић.

Остаје још да се креирани излаз прикаже на екрану што се постиже позивом операције *showCard()* класе *Browser*.

Поглавље 6

Имплементација система за каталогизацију по MARC 21 формату

Имплементација система за каталогизацију по MARC 21 формату извршена је на основу спецификације дате у претходном поглављу. За имплементацију основног едитора коришћен је поступак развоја софтвера заснован на моделима у софтверском окружењу Xtext. Додатне функционалности система за каталогизацију имплементиране су коришћењем развоја система заснованог на компонентама у Eclipse plug-in технологији.

Описана је спецификација MARC 21 библиографског записа и едитора за каталогизацију у софтверском алату Xtext. Као што је наведено у поглављу 3, Xtext је првенствено намењен креирању метамодела у виду доменски специфичних језика и едитора за моделе тог метамодела. Идеја је да се метамодел опише граматиком у нотацији сличној EBNF и да се на основу те граматике генеришу остале софтверске компоненте што је описано у поглављу 3. Међутим, у овом поглављу је показано да се Xtext може користити и за креирање модела на основу кога се могу генерисати парсер и едитор за унос података на основу тог модела.

Креирана је граматика која описује модел библиографског записа по MARC 21 формату чиме је, у ствари, према Xtext концепту MARC 21 библиографски запис посматран као доменски специфичан језик. Затим је на основу те граматике генерисан EMF модел библиографског записа, ANTLR парсер као и Eclipse едитор за унос библиографског записа по MARC 21 формату.

Језик Check је искоришћен за спецификацију ограничења над MARC 21 библиографским записом, а језик Xpand је искоришћен за трансформацију библиографског записа у XML документ и разне врсте листића.

Основни едитор који је генерисан у Xtext технологији доступан је у облику plug-in-а за Eclipse. Основна идеја у реализацији додатних функционалности система за каталогизацију састоји се у томе да се генерисани plug-in прошири

тим функционалностима коришћењем plug-in технологије. Додатне функционалности система за каталогизацију су:

- приказ податка о библиографском формату,
- експорт и импорт библиографских записа,
- унос локацијских података,
- библиотечко окружење,
- приказ каталошких листића.

6.1 ГЕНЕРИСАЊЕ ОСНОВНОГ ЕДИТОРА ЗА КАТАЛОГИЗАЦИЈУ

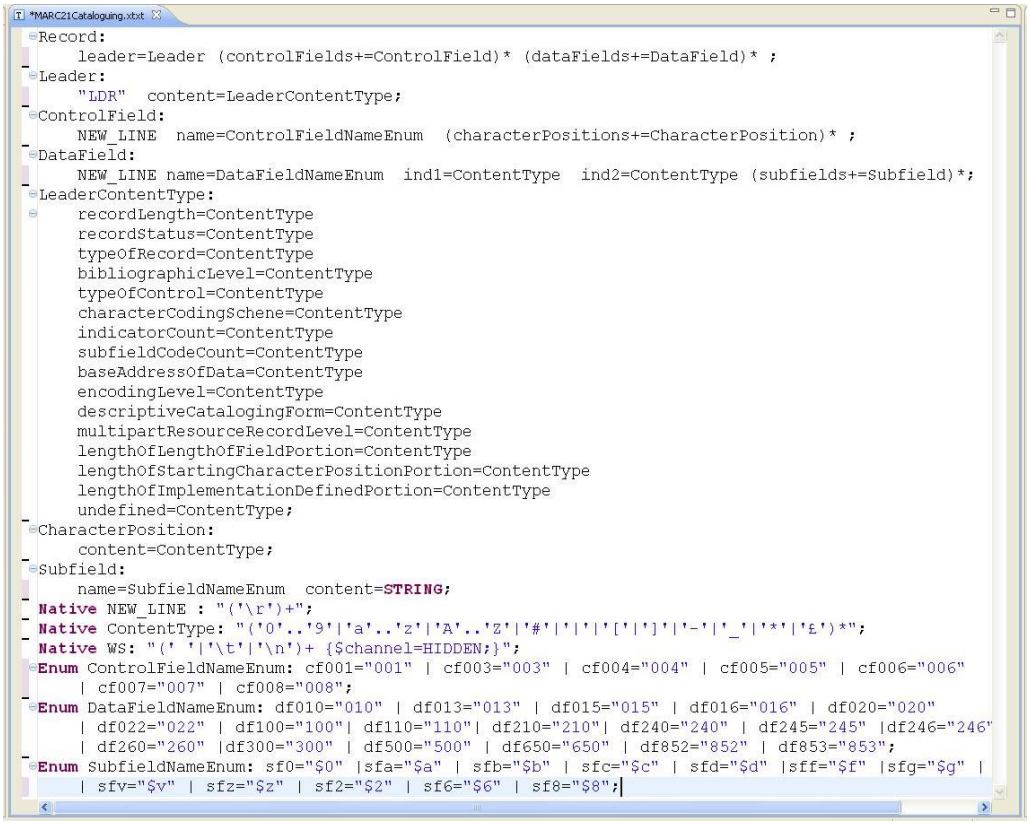
Прва итерација у развоју система за каталогизацију је креирање основног едитора за унос библиографског записа коришћењем развоја заснованог на моделима. У овој итерацији имплементира се случај коришћења *MARC 21 record editing*.

6.1.1 Спецификација библиографског записа у Xtext окружењу

У развоју едитора за каталогизацију по MARC 21 формату циљ је да се развије едитор који подржава унос библиографских записа по формату који је приказан на листингу 4.1. Структура овог формата је прописана стандардом MARC 21.

Полазна тачка је креирање спецификације библиографског записа у облику граматике у Xtext окружењу. Ова граматика креирана је на основу објектног модела MARC 21 записа који је дат у одељку 4.6.1. На основу ове граматике генерише се EMF модел језика библиографског записа и парсер за тај језик. На овај начин се коришћењем развоја софтевра заснованог на моделу добија основни едитор који је резултат прве итерације развоја система за каталогизацију.

На слици 6.1 дата је граматика којом је описан библиографски запис по MARC 21 формату, чија структура је описана у одељку 4.1. Граматика је приказана у оквиру Xtext едитора за Eclipse окружење.



Слика 6.1 Граматика библиографског записа по MARC 21 формату

У наредном тексту су описана појединачна правила спецификације.

Правило *Record*. Овим правилом описана је структура библиографског записа.

```
Record:
   leader=Leader (controlFields+=ControlField)*
(dataFields+=DataField)* ;
```

Према овом правилу библиографски запис чине следећи елементи:

- заглавље записа представљено атрибутом *leader* који је типа *Leader*,
- листа контролних поља представљена атрибутом *controlFields* која садржи елементе типа *ControlField* којим је представљено контролно поље,
- листа поља представљена атрибутом *dataFields* која садржи елементе типа *DataField* којима је представљено поље.

Редослед у ком су наведени атрибути у граматици је редослед којим се ти елементи морају појавити у библиографском запису.

Правило *Leader*. Овим правилом је описао је заглавље записа.

```
Leader:
   "LDR"  content=LeaderContentType;
```

Заглавље записа започиње кључном речи “LDR”, иза које следи садржај заглавља који је типа *LeaderContentType*.

Правило *ControlField*. Правило *ControlField* описује једно контролно поље библиографског записа.

```
ControlField:
   NEW_LINE  name=ControlFieldNameEnum
(characterPositions+=CharacterPosition)* ;
```

Контролно поље започиње native правилом *NEW_LINE* које представља ознаку за почетак реда, следи име представљено атрибутом *name* набројивог типа *ControlFieldNameEnum*, затим листа карактерских позиција представљена атрибутом *characterPositions*. Елементи листе *characterPositions* су типа *CharacterPosition* којим је представљена једна карактерска позиција у запису.

Правило *DataField*. Правило *DataField* описује поље библиографског записа.

```
DataField:
  NEW_LINE name=DataFieldNameEnum  ind1=ContentType
ind2=ContentType (subfields+=Subfield)*;
```

Поље записа такође почиње ознаком за почетак реда (правило *NEW_LINE*), има своје име које је набројивог типа *DataFieldNameEnum*. Иза имена поља следе први и други индикатор поља, а затим и листа потпоља посматраног поља. Индикатори су представљени атрибутима *ind1* и *ind2* који су типа *ContentType*. Листа потпоља представљена је атрибутом *subfields*, а елемент листе, односно потпоље је дефисано типом *Subfield*.

Правило *LeaderContentType*. Садржај заглавља записа је представљен правилом *LeaderContentType*.

```
LeaderContentType:
  recordLength=ContentType
  recordStatus=ContentType
  typeOfRecord=ContentType
  bibliographicLevel=ContentType
  typeOfControl=ContentType
  characterCodingSchene=ContentType
  indicatorCount=ContentType
  subfieldCodeCount=ContentType
  baseAddressOfData=ContentType
  encodingLevel=ContentType
  descriptiveCatalogingForm=ContentType
  multipartResourceRecordLevel=ContentType
  lengthOfLengthOfFieldPortion=ContentType
```

```
  lengthOfStartingCharacterPositionPortion=ContentType
  lengthOfImplementationDefinedPortion=ContentType
  undefined=ContentType;
```

Садржај заглавља записа чини 16 библиографских података који се у датом редоследу појављују у оквиру заглавља. Сви елементи заглавља записа су native типа *ContentType*.

Правило *CharacterPosition*. Карактерска позиција представљена је правилом *CharacterPosition*.

```
CharacterPosition:
   content=ContentType;
```

Карактерска позиција има свој садржај представљен атрибутом *content* који је native типа *ContentType*.

Правило *Subfield*. Потпоље записа дефинисано је правилом *Subfield*.

```
Subfield:
  name=SubfieldNameEnum  content=STRING;
```

Спецификација потпоља се састоји од назива потпоља представљеног атрибутом *name* који је набројивог типа *SubfieldNameEnum* и садржаја представљеног атрибутом *content* који је типа *STRING*.

На крају Xtext фајла који је приказан на слици 5.4 наведене су дефиниције native и набројивих правила која се користе у спецификацији појединачних елемената записа.

Native правило *NEW_LINE*. Native правило *NEW_LINE* дефинише ознаку за нови ред.

```
Native NEW_LINE : "('\n')+";
```

Нови ред дефинисан је најмање једним појављивањем скривеног знака за крај реда који је у програмском језику Јава означен карактером ‘\n’.

Native правило *ContentType*. Native правило *ContentType* дефинише садржај карактерских позиција у заглављу записа и контролним пољима, као и садржај индикатора.

```
Native ContentType:
"('0'..'9'|'a'..'z'|'A'..'Z'|'#'|'|'|'['|']'|'-
'|'_'|'*'|'£')*";
```

Овај елемент дефинисан је као вишеструко појављивање слова, цифара и различитих знакова који се могу јавити као садржај карактерских позиција у заглављу записа, контролним пољима и потпољима записа.

Native правило *WS*. Native правило *WS* представља редефинисање истоименог уграђеног правила.

```
Native WS: "(' '|'\t'|'\r')+ {$channel=HIDDEN;}";
```

Овим правилом специфициран је сепаратор између токена без ознаке за крај ред „\n“ која има другу функцију у граматици, а то је сепаратор између заглавља записа, контролних поља и поља и искоришћено је у првилу *NEW_LINE*.

Набројиво правило *ControlFieldNameEnum*. Набројиво правила *ControlFieldNameEnum* садржи листу свих контролних поља која се могу јавити у оквиру записа.

```
Enum ControlFieldNameEnum: cf001="001" | cf003="003" |
cf005="005" | cf006="006" | cf007="007" | cf008="008";
```

Набројиво правило *DataFieldNameEnum*. Набројиво правило *DataFieldNameEnum* садржи називе свих поља која се могу јавити у оквиру записа (овде је ради простора наведен само део поља).

```
Enum DataFieldNameEnum: df010="010" | df013="013" |
df015="015" | df016="016" | df020="020" | df022="022" |
df100="100"| df110="110"| df210="210"| df240="240" |
df245="245" |df246="246" | df260="260" |df300="300" |
df500="500" | df650="650" | df852="852" | df853="853";
```

Пун назив атрибута *name* за контролна поља и поља ће садржати и префикс *cf* односно *df*. На пример, пољу *245* ће се у изразима приступати са именом *df245* али ће у едитору стајати кључна реч *245* за назив поља.

Набројиво правило *SubfieldNameEnum*. *SubfieldNameEnum* садржи називе свих потпоља која се могу јавити у оквиру записа у било ком пољу (овде је ради простора наведен само део потпоља) .

```
Enum SubfieldNameEnum: sf0="$0" |sfa="$a" | sfb="$b" |
sfc="$c" | sfd="$d" |sff="$f" |sfg="$g" | sfh="$h"|
sfv="$v" | sfz="$z" | sf2="$2" | sf6="$6" | sf8="$8";
```

Испред назива потпоља у MARC 21 формату наводи се знак “$” иза кога следи карактер који представља сам назив.

6.1.2 Генерисање система за каталогизацију

На основу граматике описане у претходном одељку покретањем Xtext генератора генеришу се фајлави у три пројекта којима се дефинишу три plug-in-а за Eclipse окружење. Сваки од ових пројеката одговара једном подсистему који је представљен на дијаграму са слике 5.2 То су следећи пројекти:

- *org.bisis.marc21cataloguing* – имплементација подсистема *Language*,
- *org.bisis.marc21cataloguing.editor* – имплементација подсистема *Editor*,
- *org.bisis.marc21cataloguing.generator* – имплементација подсистема *Templates*.

У оквиру пројекта *org.bisis.marc21cataloguing* који представља имплементацију подсистема за спецификацију доменски специфичног језика за унос MARC 21 библиографског записа генерисане су следеће компоненте које представљају имплементацију пакета са дијаграма на слици 5.3:

- EMF модел за MARC 21 библиографски запис који представља модел генерисаног доменски специфичног језика и на њему се заснива спецификација свих осталих концепата језика и едитора. Овај модел је на дијаграму пакета на слици 5.2 представаљен пакетом *MARC21LanguageModel*.
- Окружење за спецификацију ограничења креираног доменски специфичног језика које представља имплементацију пакета *MARC21Constraints*. Ограничења се у генерисаном окружењу специфицирају у језику Check над генерисаним EMF моделом језика.
- ANTLR парсер за креирани језик који представља имплементацију пакета *MARC21Parser*.

У оквиру пројекта *org.bisis.marc21cataloguing.editor* који представља имплементацију подсистема за спецификацију основног едитора за каталогизацију генерисане су следеће компоненте које представљају имплементацију пакета са дијаграма на слици 5.4:

- Едитор у облику plug-in-а за Eclipse окружење који подржава унос MARC 21 библиогарфских записа према правилима специфициране граматике. Ова компонента представља имплементацију пакета *MARC21RecordEditor*.
- Окружење за спецификацију додатних функционалности генерисаног едитора у које спадају спецификација ограничења над моделом и спецификација помоћи за унос података. Ово окружење је имплементација пакета *MARC21EditorExtension*.
- Визарди за креирање новог прије кта и фајла за унос библиографског записа који чине имплементацију пакета *Wizards*.

Пројекат *org.bisis.marc21cataloguing.generator* садржи окружење за спецификацију темплејта за трансформацију библиографских записа и састоји се од генератора за трансформацију записа.

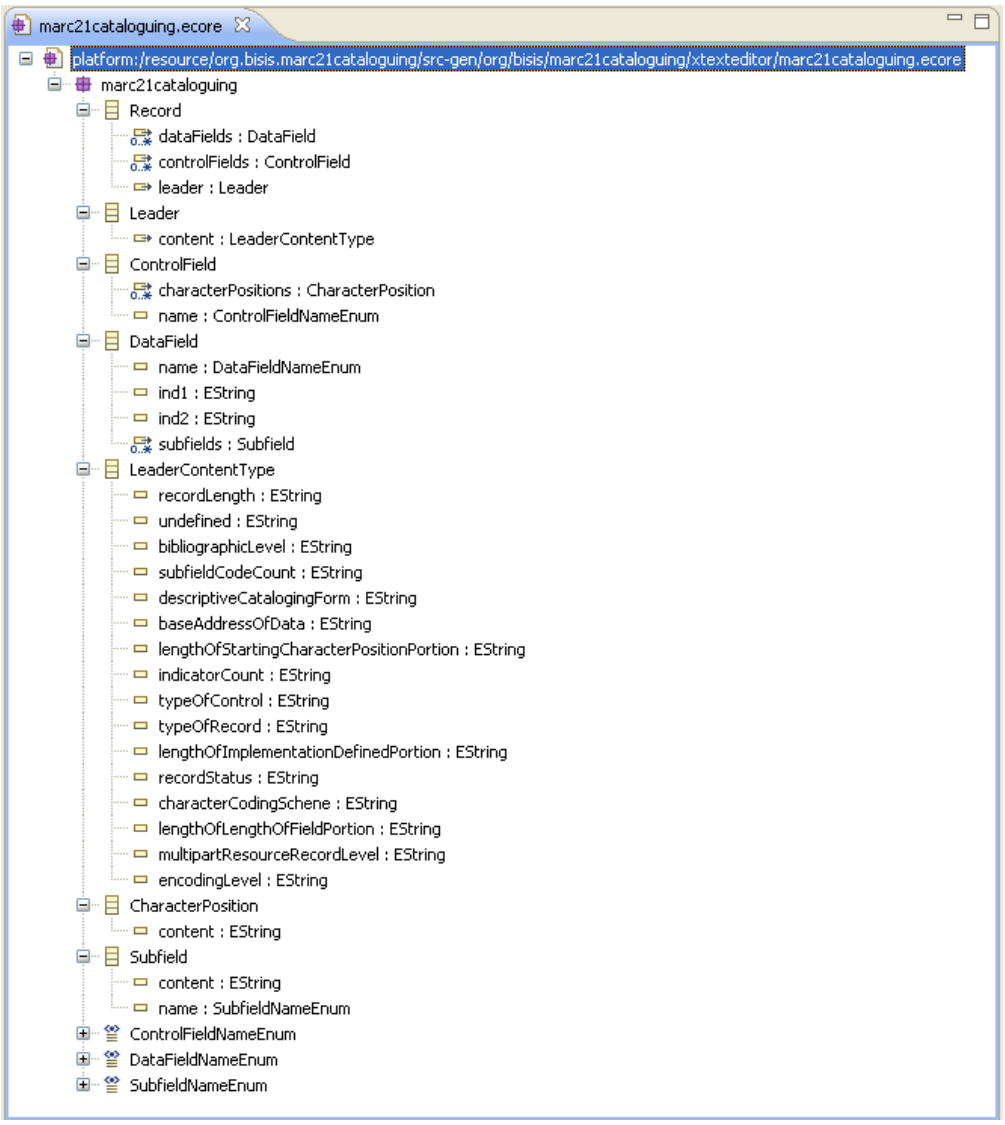
6.1.3 EMF модел MARC 21 библиографског записа

EMF модел библиографског записа заузима централно место у спецификацији система за каталогизацију, јер се на њему заснивају све остале спецификације језика и едитора као што су ограничење над језиком и помоћ приликом уноса текста у едитору. Поред тога, меморијска репрезентација текста у едитору је заснована на овом моделу, тако да се познавањем EMF модела записа може вршити програмска обрада текста који се едитира.

EMF модел библиографског записа креиран је на основу спецификације граматике библиографског записа која је описана у одељку 6.1.1. Правила на основу којих се граматика трансформише у EMF модел описани су у одељку 3.2.3.

У овом одељку дат је детаљан опис EMF модела библиографског записа. На слици 6.2 дат је графички приказ овог модела у оквуру Eclipse Ecore едитора у ком је EMF модел приказан у облику стабла.

Коренски елемент стабла на слици 6.2 садржи URI адресу на којој се налази фајл *marc21cataloguing.ecore* у ком је специфициран EMF модел MARC 21 библиографског записа. Поделемент коренског елемента је пакет *marc21cataloguing* који садржи EMF модел MARC 21 библиографског записа. У оквиру овог пакета налазе се класе и набројиви типови који се део EMF модела записа. У наставку је дат детаљан опис сваке класе EMF модела библиографског записа, а набројиви типови су објашњени на примеру једног набројивог типа. Значење иконица испред назива елемената модела на слици 6.2 описан је у одељку 3.2.3.



Слика 6.2 EMF модел MARC 21 библиографског записа

Класа *Record*

Класа *Record* је генерисана на основу истоименог правила у граматици. Ова класа представља модел целог MARC 21 библиографског записа. Класа *Record* садржи три везе асоцијације.

Прва веза асоцијације је веза *dataFields* која представља везу са класом *DataField*, има особину *containment* што значи да се ради о вези композиције и кардиналитет *0..**. Ова веза композиције означава да запис садржи нула или више поља.

Друга веза асоцијације је веза са класом *ControlField* и њен назив је *controlFields*, и такође има особину *containment* и кардиналитет *0..**. Ова веза означава да запис садржи од нула до више контролних поља.

Последња веза асоцијације је веза са класом *Leader* и њен назив је *leader*. Ова веза је такође веза композиције са кардиналитетом *0..1*, што значи да према посматраном моделу запис има највише једно заглавље.

Класа *Leader*

Класом *Leader* представљено је заглавље записа и она је генерисана на основу истоименог правила граматике. Ова класа има једну везу асоцијације чије је име *content*, а други крај асоцијације је класа *LeaderContentType*. Посматрана асоцијација представља везу композиције са кардиналитетом *0..1* која означава да заглавље записа садржи највише једну инстанцу класе *LeaderContentType* односно један садржај.

Класа *ControlField*

Класом *ControlField* представљено је контролно поље записа. Ова класа има атрибут *name* који је генерисан на основу истоименог атрибута правила *ControlField* у граматици и представља назив контролног поља. Овај атрибут је набројивог типа *ControlFieldNameEnum*. Поред атрибута *name*, класа *ControlField* има и везу асоцијације *characterPositions* са класом *CharacterPosition*. Ова веза представља везу композиције и означава да контролно поље садржи нула или више карактерских позиција.

Класа *DataField*

Класом *DataField* представљено је поље записа. Ова класа има атрибуте *name*, *ind1* и *ind2* и везу асоцијације *subfields*. Атрибут *name* је набројивог типа *DataFieldNameEnum* и представља назив поља. Атрибути *ind1* и *ind2* су типа *EString* који је изгенерисан на основу типа *ContentType* који је у спецификацији граматике дефинисан native правилом као низ карактера. Атрибути *ind1* и *ind2* класе *DataField* представљају први и други индикатор поља.

Веза асоцијације *subfields* класе *DataField* је веза ове класе са класом *Subfield* и представља везу композиције којом је специфицирано да поље садржи нула или више потпоља.

Класа *LeaderContentType*

Класом *LeaderContentType* представљен је тип садржаја заглавља записа. Ова класа садржи 16 атрибута који представљају 16 библиографских података у заглављу записа. Атрибути класе *LeaderContentType* генерисани су на основу

атрибута истоименог правила у граматици. Атрибути правила *LeaderContentType* у граматици су типа *ContentType* који је у EMF модел пресликан у тип *EString*.

Класа *CharacterPosition*

Класом *CharacterPosition* представљена је карактерска позиција записа. Ова класа има један атрибут, то је атрибут *content* који је типа *EString* и представља садржај карактерске позиције.

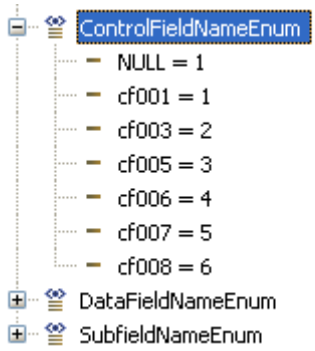
Класа *Subfield*

Класа *Subfield* представља потпоље записа и она има два атрибута, то су атрибут *name* који је набројивог типа *SubfieldNameEnum* и представља назив потпоља и атрибут *content* који представља садржај потпоља.

Набројови тип *ControlFieldNameEnum*

На слици 6.3 представљена је спецификација набројивог типа *ControlFieldNameEnum* која садржи називе свих контролних поља који могу да се јаве у запису. Овај тип генерисан је на основу истоименог набројивог правила у граматици. Набројиви тип се састоји од литерала који се генеришу на основу литерала специфицираних за набројиво правило у граматици.

Литерали који су специфицирани у граматици пренети су и овде с тим што је додат још један литерал *NULL* који ће бити подразумевани (*default)* литерал за овај тип. Сваки литерал се састоји од свог имена и вредности која је аутоматски генерисана за све литерале. На пример, назив контролног поља *001* представљен је литералом чије је има *cf001* а вредност *1*.

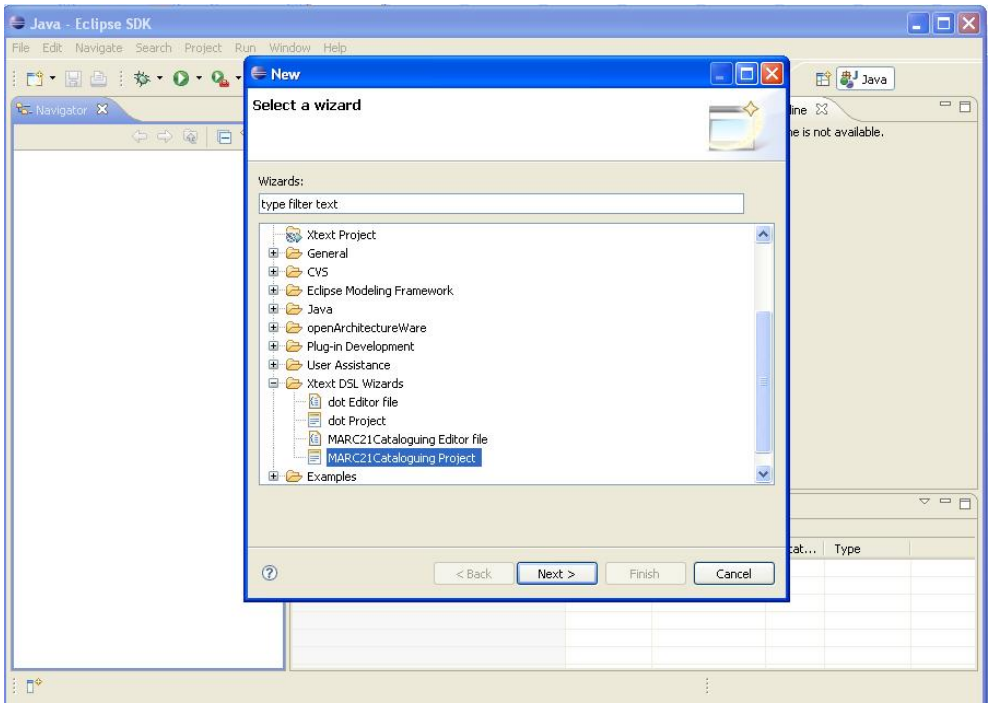


Слика 6.3 Спецификација EMF набројивог типа

На сличан начин су специфицирана и остала два набројива типа, *DataFieldNameEnum* за назив поља и *SubfieldNameEnum* за назив потпоља.

6.1.4 Основни едитор

Ако се plug-in-ови који су генерисани на основу граматике додају у скуп plug-in-ова Eclipse-а добија се резултат прве итерације развоја система за каталогизацију, а то је основни едитор за унос библиографских записа у оквиру Eclipse платформе. Додавањем наведених plug-in-ова у оквиру Eclipse-а постају доступни визарди за креирање новог пројекта за каталогизацију, као и новог фајла за унос библиографског записа. На слици 6.4 приказан је изглед екранске форме за креирање новог ресурса у оквиру Eclipse у ком се може изабрати један од визарда за основни едитор.

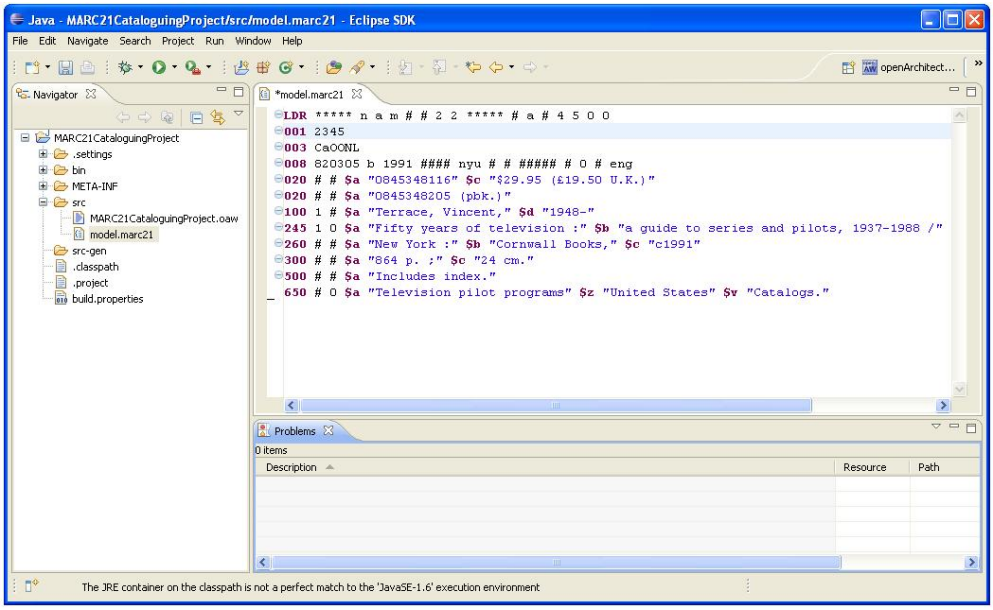


Слика 6.4 Избор визарда за креирање новог пројекта за каталогизацију

Да би се покренуо генерисани едитор треба прво направити пројекат за MARC 21 каталогозацију. На слици 6.5 приказан је основни едитор у оквиру Eclispe окружења. У едитору је отворен фајл *model.marc21* који је креиран у оквиру пројекта *MARC21CataloguingProject*. На овом нивоу развоја система за каталогизацију, коришћење едитора за каталогизацију је слично развоју софтвера у Eclipse окружењу. Наиме, лева страна екранске форме на слици 6.5 резервисана је за често коришћену Eclipse Navigator форму која омогућава увид у ресурсе са којима се ради у едитору. Доњи део екранске форме резервисан је за такође често коришћену Eclipse Problems форму у којој се

наводи извештај о грешкама које постоје у доступним ресурсима (на пример грешке у Јава коду које компјалер пријављује).

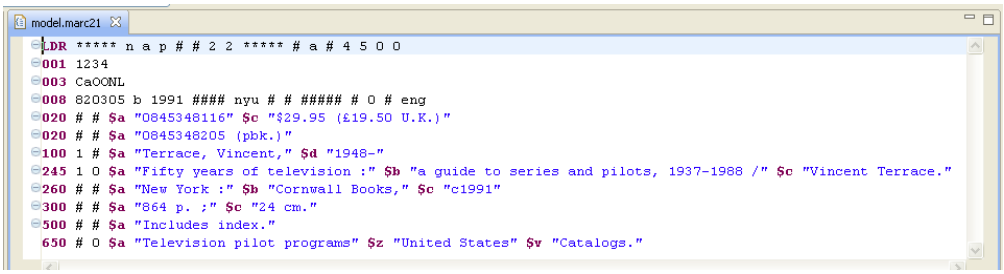
Централни део Eclipse екранске форме која је приказана на слици 6.5 заузима едитор за унос библиографског записа који представља имплементацију случаја коришћења *MARC 21 record editing* који је приказан на дијаграму случајева коришћења на слици 5.1. У овај едитор унети су подаци о једном библиографском запису. Опис приказа библиографског записа у едитору дат је у наставку.



Слика 6.5 Основни едитор за MARC 21 библиографски запис

6.1.4.1 Приказ библиографског записа у едитору

На слици 6.6 приказан је један библиографски запис у генерисаном едитору за каталогизацију. Овај библиографски запис је инстанца EMF модела који је описан у одељку 6.1.3. Поред тога библиографски запис приказан на слици 6.6 задовољава сва правила специфициране граматике која је описана у одељку 6.1.1.



Слика 6.6 Приказ библиографског записа у едитору

Први ред библиографског записа приказаног на слици 6.6 резервисан је за заглавље записа. Заглавље записа започиње кључном речи „LDR“ иза које следи 16 библиографских података раздвојених празним карактером. Значење библиографског податка дефинисано је његовом позицијом у оквиру заглавља записа. Тако на пример, као трећи податак у заглављу уноси се тип записа. Овај податак је у EMF моделу (одељак 6.3) представљен као атрибут EMF класе *LeaderContentType* под називом *typeOfRecord*. У запису који је приказан на слици 6.6 на овом месту унет је карактер „a“ који представља шифру за текстуалну грађу.

Следећа три реда библиографског записа представљају контролна поља. Свако контролно поље наводи се у новом реду што одговора дефинисаној граматици, односно приказу MARC формата. Контролно поље започиње својим именом које је једно од имена наведених у оквиру набројивог типа *ControlFieldNameEnum*. Називи контролних поља су такође кључне речи и зато имају истакнуту синтаксу (обојену јачом бојом). Иза назива контролног поља следе њихов садржај који је низ карактерских позиција. У примеру са слике 6.6 постоје контролна поља која имају један податак, односно једну карактерску позицију, као у случају поља 001 и 003 и поље које има више карактерских позиција, то је поље 008. У случају појављивања више карактерских позиција оне су раздвојене празним карактером.

После контролних поља у запису су наведена поља која се такође јављају у новом реду. И поља започињу својим именом које је кључна реч и има истакнуту синтаксу. После имена следе вредности првог и другог индикатора за поље а затим и листа потпоља. Сви концепти који се јављају у оквиру поља су раздвојени празним карактером.

Потпоља која припадају пољу су наведена у једном реду и започињу називом потпоља. Назив потпоља се састоји од знака „$“ и слова или цифре. И називи потпоља су кључне речи и зато имају истакнуту синтаксу. Поред назива потпоља, потпоље има и свој садржај који је према граматици типа *String* и зато је у запису наведен под знацима навода.

Основни едитор који се генерише на основу Xtext граматике се може у датом софтверском алату проширити додатним функционалностима које се односе на спецификацију ограничења над елементима библиографског записа на основу којих ће се вршити контрола исправности унетих података (одељак 6.5) и генерисање помоћи приликом уноса у виду понуде могућих вредности за унос (одељак 6.6). Ове додатне функционалности реализоване су у другој итерацији развоја система за каталогизацију.

6.1.4.2 Објектни модел библиографског записа у генерисаном едитору

Библиографски запис који је приказан као садржај генерисаног едитора на слици 6.6 представља инстанцу EMF модела библиографског записа описаног у одељку 6.1.3. Све додатне функције над едитором специфицираће се над овим EMF моделом записа. Проширења у језику Xtend се специфицирају као проширења EMF модела записа, ограничења над библиографским записом специфицирају се као ограничења његовог EMF модела, помоћ за допуну текста специфицира се за елементе EMF модела, а темплејти за трансформацију записа специфицирају се као трансформације тог модела. Поред тога, напредније коришћење библиографског записа из едитора, у имплементацији система за каталогизацију вршиће се на основу његове репрезентације у овом EMF моделу.

Објектни модел библиографског записа у наставку је објашњен на примеру записа који је приказан на слици 6.6 у оквиру генерисаног едитора.

Цео библиографски запис који је креиран у едитору представља инстанцу EMF класе *Record*. Вредност његове референце *leader* добија вредност инстанце класе *Leader* која се креира на основу токена унетих у прву линију записа у едитору (слика 6.6) резервисану за заглавље записа. Сваки од токена у овом реду мапира се на одговарајући атрибут класе *LeaderContentType* на основу позиције на којој је унет. На овај начин креирана инстанца класе *LeaderContentType* постаје вредност атрибута *content* класе *Leader*. Кључна реч *LDR* не улази у модел записа.

Вредност референце *controlFields* класе *Record* постаје листа контролних поља која су унета у запис. За свако контролно поље креира се инстанца EMF класе *ControlField*. Ова инстанца класе *ControlField* креира се на основу токена који су унети у једну линију записа у едитору која је резервисана за контролно поље. Први токен у тој линије постаје име контролног поља, односно вредност атрибута *name* класе *ControlField*. Име контролног поља је набројивог типа *ControlFieldEnum* тако да ће вредност атрибута *name* бити назив литерала у овом набројивом типу. На пример, за контролно поље *001* вредност

одговарајућег атрибута *name* је *cf001*. Сви остали подаци унети у линију за контролно поље постају инстанце класе *CharacterPosition* и смештају се у листу *characterPositions* у класи *ControlField*.

Вредност референце *dataFields* класе *Record* постаје листа поља која је унета у запис. За свако поље креира се инстанца EMF класе *DataField* на основу токена који се налазе у једној линији записа која је резервисана за поље. Први токен у тој линији постаје име поља односно вредност атрибута *name*. И у случају поља овај атрибут добија вреднот литерала у набројивом типу *DataFieldNameEnum*. Други токен у линији за поље постаје вредност првог а трећи токен вредност другог индикатора поља. Остали подаци у пољу мапирају се на инстанце класе *Subfield* и смештају се у листу потпоља *subfields* у класи *DataField*. Ова листа се формира тако што се узимају два суседна токена и на основу њих се креира једна инстанца класе *Subfield* и то тако да први токен постаје назив потпоља односно вредност атрибута *name*, а други постаје садржај односно вредност атрибута *content*. И у случају потпоља, назив потпоља добија вредност литерала у набројивом типу *SubfieldNameEnum*.

За писање израза у језику Xtend користи се описани објектни модел. За кретање по структури модела користе се следећи механизми:

- атрибутима и референцама класа приступа се статички
  на пример: ако је у променљиву *rec* смештен цео запис који је унет у едитору, вредност атрибута *recordLength* у заглављу записа може се добити следећим изразом *rec.leader.content.recordLength*;
- елементима листе у моделу приступа се помоћу уграђених израза за рад са листама који су описани у одељку 3.2.7
  на пример: друго потпоље поља које је представљено променљивом *df* која је инстанца класе *DataField* може се добити следећим изразом *df.subfields.get(2)*;
- за сваки елемент модела може се добити елемент који га садржи помоћу атрибута *eContainer* који је подразумевани атрибут за све елементе *EMF* модела.
  на пример: поље коме припада потпоље представљено променљивом *sf* добија се следећим изразом
  sf.eContainer.

6.2 ГЕНИРАСАЊЕ ДОДАТНИХ КАРАКТЕРИСТИКА ОСНОВНОГ ЕДИТОРА

У другој итерацији развоја система за каталогизацију реализован је проширени основни едитор који подржава напредне функције едитирања MARC 21 записа. У овој итерацији имплементирани су случајеви коришћења *Record validation* (одељак 6.2.1), *Cintent assist* (одељак 6.2.2) и *Generate catalogue card* (6.2.3.1).

Ова итерација такође је реализована коришћењем развоја заснованог на моделима. Генерисање помоћи за унос реализовано је као проширење EMF модела MARC 21 записа написано у језику Xtend, а контрола унетих података врши се на основу спецификације ограничења над EMF моделом записа написано у језику Check. Генерисање каталошких листића реализовано је као трансформација EMF модела библиогарфског записа у HTML документ каталошког листића. Ове трансформације специфициране су у виду темплејта написаних у језику Xpand.

6.2.1 Спецификација ограничења над библиографским записом

Контрола исправности уноса података у генерисани едитор обухвата комплексну контролу библиографских података на основу ограничења која су дефинисана библиографским стандардом и описана у одељку 4.1.1. У изабраном софтверском окружењу постоји механизам за аутоматску проверу исправности унетих података у едитору са слике 6.5. Та провера се врши на основу спецификације ограничења која је наведена у фајлу *Check.chk* у оквиру пројекта *org.bisis.marc21cataloguing*. У оквиру спецификације ограничења могу се користити проширења специфицирана у фајлу *Extensions.ext*.

Спецификација ограничења односи се на дефинисање израза у језику *Check* који је тачан ако није нарушена исправност структуре или садржаја записа у односу на MARC 21 формат и специфицирање поруке која ће се проследити у случају да тај израз није задовољен. Спецификација ограничења дефинише се над EMF моделом библиографског записа који је описан у одељку 6.1.3.

Спецификација ограничења над библиографским записом приказана је у наставку у неколико примера.

Пример 1. Контрола поновљивости поља

Једна од контрола структуре записа односи се на проверу колико се пута неки елемент записа (поље или потпоље) појављује у запису и ако се појављује

више пута да ли тај елемент према библиографском формату има особину поновљивости. Ова врста контроле се може специфицирати на следећи начин:

```
Context DataField ERROR "Polje"+this.name.toString().
subString(2,5)+" nije ponovljivo!"  :
isFieldOccurenceOK(this);
```

У контексту *DataField* јавиће се одговарајућа порука о насталој грешци да поље са одређеним именом није поновљиво ако је вредност израза *isFieldOccurenceOK* нетачно односно *false*.

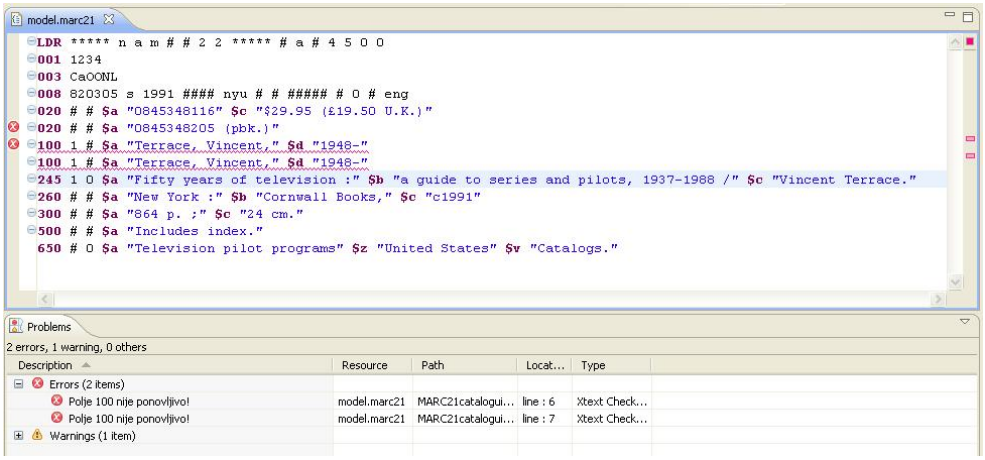
Израз *isFieldOccurenceOK* специфициран је као проширење у фајлу *Extensions.ext* на следећи начин:

```
Boolean isFieldOccurenceOK(DataField df):
  let rec = (Record)df.eContainer:
  if(rec.dataFields.select(e|((DataField)e).name==df.nam
  e).size>1) then
  repeatableFields().contains(df.name.toString().subStri
  ng(2,5));
```

За прослеђено поље *df* преузима се елемент који га садржи а то је цео запис и смешта у променљиву *rec*. Затим се над листом поља записа селектују сва поља чије је име једнако имену поља *df* и уколико је број таквих поља већи од један враћа се логичка вредност тачно ако је поље поновљиво, односно нетачно уколико поље није поновљиво према дефиницији MARC 21 формата.

Поновљивост поља се одређује тако што се проверава да ли назив поља припада листи поновљивих поља која је специфицирана проширењем *repeatableFields()*.

На слици 6.7 приказан је изглед едитора у случају да је два пута унето поље 100 које по MARC 21 формату није поновљиво. Оба поља 100 су подвучена црвеном бојом, а у доњем делу екранске форме су наведени текстуални описи грешака који су генерисани на основу спецификације. Двокликом миша на тај текстуални опис у запису ће се селектовати ред на који се та грешка односи. Исти текстуални опис грешака би се појавио на екранској форми ако би се показивач мишао задржао на иконици испред поља 100 (црвеним круг са знаком X).



Слика 6.7 Извештај о појави грешке у структури записа – непоновљива поља

Пример 2 Контрола исправности шифре

Један пример контроле садржаја записа односи се на проверу унете шифре у шифриране елементе формата, који могу бити карактерска позиција у заглављу записа или контролном пољу, индикатори или потпоља.

Контролно поље *008* на другој позиција садржи шифрирани податак о типу датума који је унет у неке друге елементе записа.

У фајлу *Check* ово ограничење се представља изразом:

```
context CharacterPosition ERROR "Karakterska pozicija
nema odgovarajucu vrednost!":
isCharPositionContentOK(this);
```

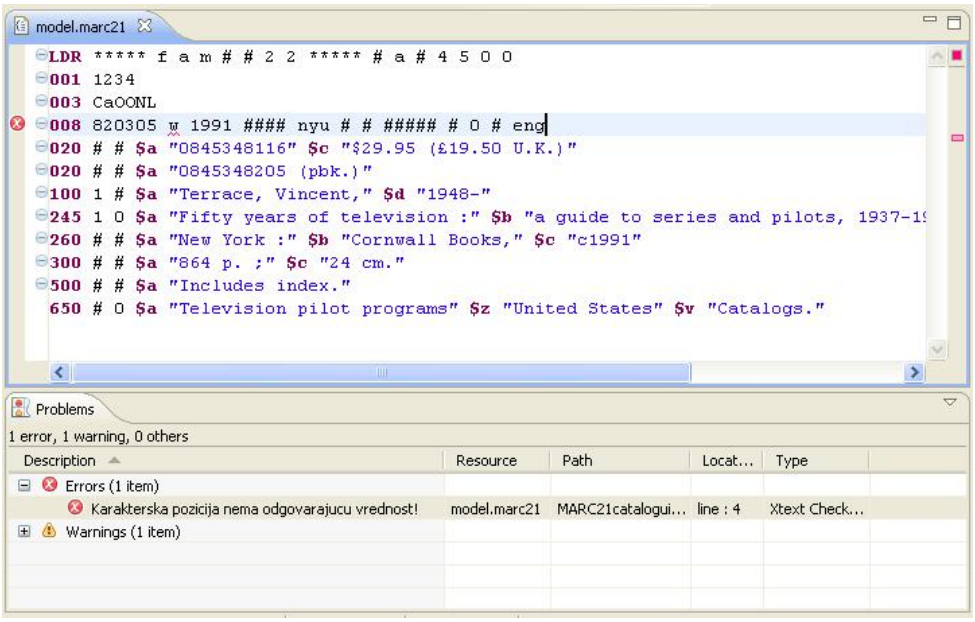
При чему проширење *isCharPositionContentOK(this)* враћа логичку вредност и специфициран је изразом:

```
Boolean isCharPositionContentOK(CharacterPosition cp):
      let cf = (ControlField)cp.eContainer:
      if(cf.name.toString().subString(2,5)=="008" &&
isPosition(cp,"008",1)) then
      field008cp2().contains(cp.content.toString())
      else true;
```

На почетку се преузима елемент записа који садржи позицију карактера, односно контролно поље и смешта у променљиву *cf*. Затим се испитује да ли је име тог контролног поља *008* и да ли је у питању друга позиција (ознаке позиција почињу од 0) и ако јесте враћа логичку вредност тачно ако листа дозвољених шифара за другу позицију поља *008* – *field008cp2()* садржи вредност унету као садржај карактерске позиције која се посматра.

Израз *isPosition(CharacterPosition cp, String fieldName, int charPosIndex)* испитује да ли је дата позиција *cp* у пољу са именом *fieldName* на месту са индексом *charPosIndex*.

На слици 6.8 приказан је изглед едитор у ком је пријављена грешка на другој позицији поља 008. У запису је подвучена шифра *w* која није у шифарнику за другу карактерску позицију поља *008*.



Слика 6.8 Извештај о појави грешке у садржају записа – погрешна шифра за карактерску позицију

Наведени пример испитује само другу позицију контролног поља *008* и једноставно се проширује на све остале шифриране карактерске позиције.

На сличан начин се специфицирају и ограничења на шифриране вредности за индикаторе, потпоља и садржаја у заглављу записа.

Пример 3 Контрола формата садржаја

Још један пример контроле садржаја записа односи се на контролу формата унетог податка. Неке карактерске позиције или потпоља имају тачно одређени формат податка који у њих може да се унесе. Ту спадају подаци као што су датуми, године, ISBN и ISSN бројеви.

Ограничење на садржај потпоља се специфицира изразом:

```
context Subfield ERROR "Sadrzaj potpolja "+this.name+"
nije dobrog formata!": subfieldContentFormatOK(this);
```

Проширење *subfieldContentFormatOK(this)* враћа логичку вредност тачно ако је садржај потпоља доброг формата, и он је специфициран тако да на основу

назива поља и потпоља одреди коју врсту провере да позове. Спецификација овог израза је:

```
Boolean subfieldContentFormatOK(Subfield sf):
switch{
case(((DataField)sf.eContainer).name.toString().subStrin
g(2,5)=="020" &&
sf.name.toString().subString(2,3)=="a"):
          isISBNok(sf.content)
default:true
};
```

Овом спецификацијом се контролише само садржај потпоља *a* поља *020*, али се слично може проширити и за све остале провере формата садржаја.

У MARC 21 поље *020*, потпоље *a* уноси се ISBN број за монографске публикације. ISBN број по MARC 21 формату може имати дужину 10 или 13 и садржи цифре (0..9) или знак X с тим што се после ISBN броја у потпоље *a* поља *020* може унети размак и неки податак у загради.

Помоћно проширење *isISBNok(sf.content)* узима садржај потпоља и над њим примењује конкретан алгоритам провере исправности ISBN броја. Ово проширење је у језику *Check* специфицирано на следећи начин:

```
Boolean isISBNok(String isbn):
 let isbnS = isbn.split(" ").get(0):
 if (isbnS.length==10 || isbnS.length==13) then
     isbnS.toCharList().
      select(e|!isValidISBNCharacter(e)).size==0
 else
  false;
```

Прво се прослеђени стринг операцијом *split* дели на подстрингове који су раздвојени знаком за празан карактер и ти подстрингови се смештају у један низ. Први елемент тог низа је ISBN број чији формат треба проверити. Прво се провера дужина ISBN броја која може бити 10 или 13 и ако је дужина у реду, враћа се логичка вредност тачно ако не постоји знак у ISBN броју који није валидан, односно није цифра или знак X. У наставку је приказана спецификација израза *isValidISBNCharacter(char c)* којим се испитује да ли је знак *c* валидан ISBN знак.

```
Boolean isValidISBNCharacter(char c):
 isDigit (c) || c=='X';
```

У наведеној спецификацији се користи помоћни израз *isDigit(char c)* којим се проверава да ли је знак *c* цифра, и који је облика:

```
Boolean isDigit(char c):
 c=='0' || c=='1' || c=='2' || c=='3' || c=='4' ||
c=='5' || c=='6' ||    c=='7'|| c=='8' || c=='9';
```

6.2.2 Генерисање помоћи за допуну текста

Приликом креирања библиографског записа морају се поштовати правила прописана стандардом. Да би се једноставније креирао запис који у потпуности задовољава MARC 21 формат обезбеђена је помоћ у виду генерисања могућих вредности за унос. Та помоћ се у окружењу Xtext реализују креирањем спецификације за допуну текста за коју се користи језик Xtend и која је описана у одељку 3.2.9. Ова спецификација се дефинише у фајлу *ContentAssist.ext* које се налази у пројекту за спецификацију едитора за каталогизацију, *org.bisis.marc21cataloguing.editor*. Спецификација допуне текста се састоји од специјалних проширења чији назив указује на који елемент језика се односе, а тип који се враћа као резултат проширења је листа чији елементи су типа *Proposal* који је део језика *Xtend*.

Генерисање понуде за допуну текста се разликује од позиције на којој се налази курсор у едитору, односно у зависности од елемента записа који се тренутно едитира. Помоћ се може генерисати приликом уноса оних елемената записа који имају тачно дефинисан скуп могућих вредности. Ту спадају неке позиције у заглављу записа која су шифриране, називи контролних поља, поља или потпоља, као и вредности индикатора, шифриране карактерске позиције или шифрираног потпоља.

Спецификација допуне текста се у потпуности ослања на EMF модел записа који је описан у одељку 6.1.3. За сваки атрибут EMF класе дефинише се листа могућих вредности који се могу унети. За препознавање о ком елементу записа се ради у позиву допуне кода користиће се изрази језика *Xpand*, као што су *if-else*, *switch, let,* и други.

Унаставку текста дата је спецификација израза за допуну текста за неколико примера.

Пример 1. Помоћ за унос шифре у заглављу записа

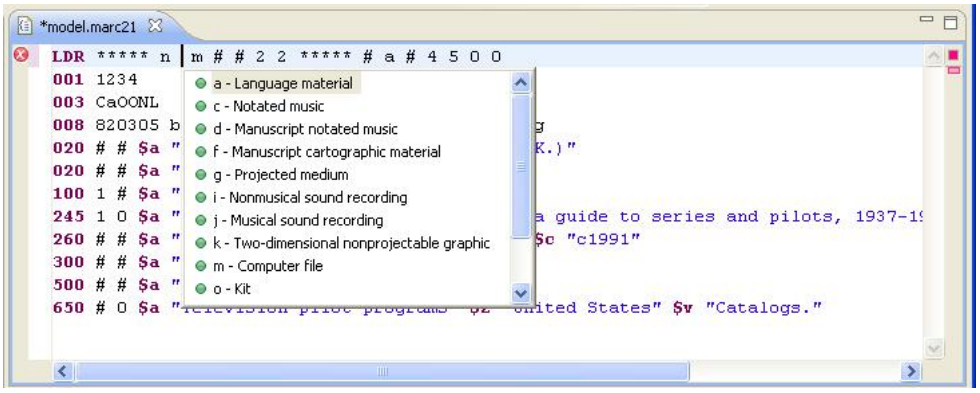
Заглавље библиографског записа се састоји од 16 података, при чему је библиографско значење тих података одређено њиховом позицијом у заглављу записа. На пример, на трећој позицији се налази податак о типу библиографског записа која је шифрирана. Овај елемент записа је у EMF моделу представљен као атрибут *typeOfRecord* правила *LeaderContentType*. За овај елемент записа, листа понуда за допуну текста се специфицира следећим проширењем:

```
List[Proposal]
completeLeaderContentType_typeOfRecord(emf::EObject ctx,
String prefix) :
 {newProposal("a - Language material","a"),
  newProposal("c - Notated music","c"),
  newProposal("d - Manuscript notated music","d"),
  newProposal("f - Manuscript cartographic
material","f"),
```

```
  newProposal("g - Projected medium","g"),
  newProposal("i - Nonmusical sound recording","i"),
  newProposal("j - Musical sound recording","j"),
  newProposal("k - Two-dimensional nonprojectable
graphic","k"),
  newProposal("m - Computer file","m"),
  newProposal("o - Kit","o"),
  newProposal("p - Mixed materials","p"),
  newProposal("r - Three-dimensional artifact or
naturally occurring object","r"),
  newProposal("t - Manuscript language material","t")};
```

Назив проширења говори о томе да се ово проширење односи на EMF класу *LeaderContentType* и то на његов атрибут *typeOfRecord*. Листа понуда се састоји од свих шифара које могу да се унесу за тип записа и то као лабела се наводи шифра заједно са својим описом, а текст који се преноси у едитору је само шифра.

На слици 6.9 приказан је изглед падајуће листе за понуду текста на позицији на којој се уноси шифра за тип записа, односно, трећи податак у заглављу записа.



Слика 6.9 Допуна текста за трећи податак у заглављу записа

Пример 2. Помоћ за унос назива поља

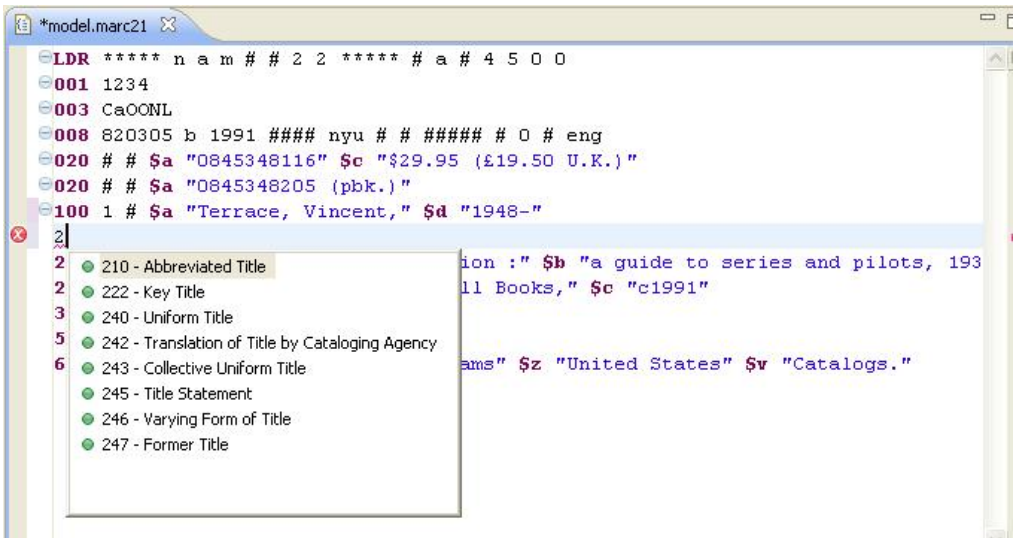
Приликом уноса поља записа кориснику је обезбеђено да поље изабере из понуђене листе поља. Ова помоћ је обезбеђена проширењем за допуну текста које се специфицира за атрибут *name* EMF класе *DataField*. У овом проширењу се као допуна наводе сва поља која су дефинисана за библиографски формат MARC 21. У наставку је приказано ово проширење:

```
List[Proposal] completeDataField_name(emf::EObject
ctx,String prefix) :
 {newProposal("010 - Library of Congress Control
Number", "010 # #"),
```

```
  newProposal("013 - Patent Control Information", "013 #
#"),
  newProposal("015 - National Bibliography Number", "015
# #"),
  newProposal("016 - National Bibliographic Agency
Control Number", "016"),
  newProposal("017 - Copyright or Legal Deposit Number",
"017"),
  newProposal("018 - Copyright Article-Fee Code", "018 #
#"),
  newProposal("020 - International Standard Book
Number", "020 # #"),
  newProposal("022 - International Standard Serial
Number", "022"),
  newProposal("024 - Other Standard Identifier", "024"),
  newProposal("025 - Overseas Acquisition Number",
"025"),
  newProposal("026 - Fingerprint Identifier", "026"),
  newProposal("027 - Standard Technical Report Number",
"027"),
  ...
  newProposal("210 - Abbreviated Title", "210"),
  newProposal("222 - Key Title", "222"),
  newProposal("240 - Uniform Title", "240"),
  newProposal("242 - Translation of Title by Cataloging
Agency", "242"),
  newProposal("243 - Collective Uniform Title", "243"),
  newProposal("245 - Title Statement", "245"),
  newProposal("246 - Varying Form of Title", "246"),
  newProposal("247 - Former Title", "247")
 ...
};
```

У наведеном листингу за спецификацију понуде за допуну назива поља за сваку понуду специфицирана је лабела која садржи назив и опис поља, а текст који се преноси у едитору садржи назив поља, а за поља која немају дефинисане индикаторе преносе се и два знака „#“ којима се одмах попуњавају и вредности за први и други индикатор.

Приликом уноса назива поља корисник може само да затражи помоћ притиском на тастер Ctrl+Space и тада му се у падајућој листи нуде сва поља за унос. Поред тога, може се извршити филтерисање скупа поља на основу унетог првог знака у називу поља. На слици 6.10 приказана је помоћ која је генерисана након уноса цифре „2“ где се сада у листи налазе само поља која почињу цифром „2“.



Слика 6.10 Допуна текста за назив поља

Пример 3. Помоћ за унос назива потпоља

Приликом спецификације помоћи за унос назива потпоља за свако поље се посебно генерише листа могућих потпоља која за поље могу да се унесу, односно у овом случај проширења мора се реализовати гранање у зависности од тога за које поље се генерише листа. Гранање ће бити реализовано преко израза *switch* који је у језику Xtend реализован на исти начин као у програмском језику Јава.

У наставку је приказан пример проширења за генерисање помоћи приликом уноса назива потпоља.

```
List[Proposal] completeSubfield_name(emf::EObject ctx,
String prefix) :
switch{
case(((DataField)ctx.eContainer).name.toString().subStri
ng(2,5)=="020"):{newProposal("$a - International
Standard Book Number","$a \"\""),
  newProposal("$c - Terms of availability","$c \"\""),
  newProposal("$z - Canceled/invalid ISBN","$z \"\""),
  newProposal("$6 - Linkage","$6 \"\""),
  newProposal("$8 - Field link and sequence number ","$8
\"\"")
}
case(((DataField)ctx.eContainer).name.toString().subStri
ng(2,5)=="022"):{newProposal("$a - International
Standard Serial Number","$a \"\""),
  newProposal("$l - ISSN-L","$l \"\""),
```

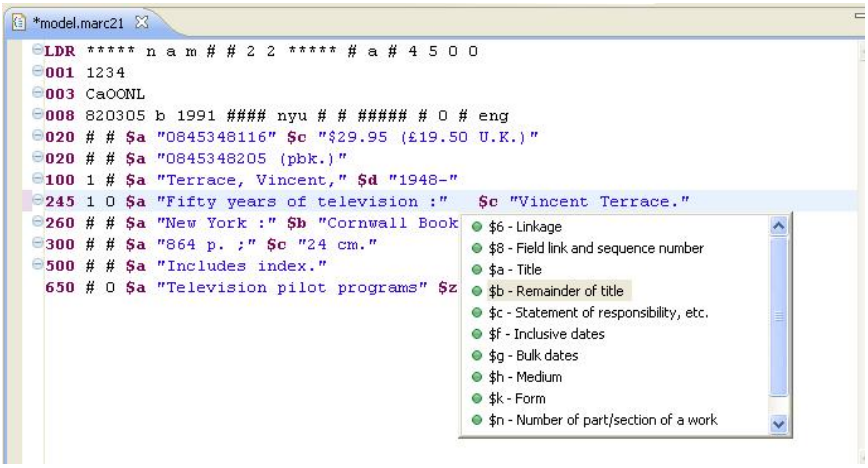
```
  newProposal("$m - Canceled ISSN-L","$m \"\""),
  newProposal("$y - Incorrect ISSN","$y \"\""),
  //...
 newProposal("$6 - Linkage","$6 \"\""),
 newProposal("$8 - Field link and sequence number ","$8
\"\"")
}
case(((DataField)ctx.eContainer).name.toString().subStri
ng(2,5)=="210"):{newProposal("$a - Abbreviated
title","$a \"\""),
  newProposal("$b - Qualifying information ","$b \"\""),
  newProposal("$2 - Source ","$2 \"\""),
  newProposal("$6 - Linkage","$6 \"\""),
  newProposal("$8 - Field link and sequence number","$8
\"\"")
}
case(((DataField)ctx.eContainer).name.toString().subStri
ng(2,5)=="245"):{newProposal("$a - Title","$a \"\""),
  newProposal("$b - Remainder of title","$b \"\""),
  newProposal("$c - Statement of responsibility,
etc.","$c \"\""),
  newProposal("$f - Inclusive dates ","$f \"\""),
  newProposal("$g - Bulk dates","$g \"\""),
  newProposal("$h - Medium","$h \"\""),
  newProposal("$k - Form","$k \"\""),
  newProposal("$n - Number of part/section of a work
","$n \"\""),
  newProposal("$s - Version ","$s \"\""),
  newProposal("$6 - Linkage","$6 \"\""),
  newProposal("$8 - Field link and sequence number ","$8
\"\"")}
default: {}
};
```

На сличан начин ово проширење може се допунити за сва поља MARC 21 формата додавањем нове *case* гране у изразу *switch*.

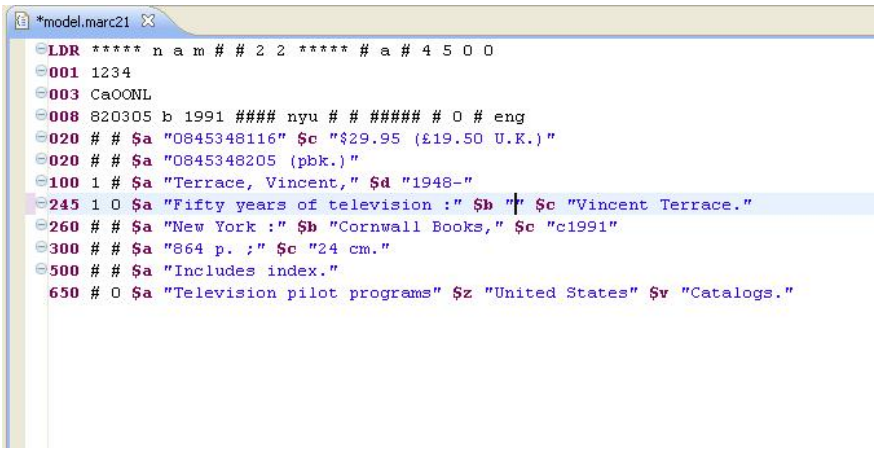
Објекат *ctx* који је први параметар овог проширења ће приликом позива овог проширења добити вредност инстанце потпоља која је у едитору на реду да се унесе на позицији курсора. Да би се приступило елементу који садржи посматрано потпоље позива се израз *ctx.eContainer* који враћа инстанцу поља и може се кастовати типом *DataField*. На овај начин је добијена референца на поље за које се генерише понуда за називе потпоља и која одређује која *case* грана ће вратити резултујућу листу понуде за допуну потпоља.

Садржај потпоља је у граматици специфициран као стринговски тип и уноси се под знацима навода. Наведено проширење ће за изабрано потпоље у едитор пренети назив потпоља и знаке навода између којих треба да се наведе садржај потпоља.

На слици 6.11 приказана је листа понуда за потпоља поља 245 и којој је селектована шифра *$b- Remainder of title*. Притиском на тастер <*Enter*> на позицију курсора ће се пренети селектована шифра са знацима навода. Ова ситуација је приказана на слици 6.12.



Слика 6.11 Допуна текста за назив потпоља поља 245



Слика 6.12 Пренос текста за изабрани назив потпоља

6.2.3 Трансформације библиографског записа

Библиографски запис креиран у едитору може се трансформисати у разне облике писањем темплејта за трансформацију у језику Xpand. Ови темплејти

извршавају се помоћу токова послова (*workflow*) који представљају конфигурационе фајлове написане у XML-у.

Трансформација библиографског записа се састоји у томе да се запис који је тренутно отворен у едитору учита у темплејт којим се креира нови фајл чији садржај је креиран на основу учитаног записа и правила која су специфицирана темплејтом. Фајл који се креира темплејтом може бити у било ком облику, на пример текстуални фајл, XML, html или јава класа.

Као илустративни пример трансформације записа у наставку је показано како се запис креиран у едитору може трансформисати у XML документ. На слици 6.13 приказан је фајл *RecordToXML.xpt* који садржи темплејт написан у језику Xtend за трансформацију записа креираног у генерисаном едитору у XML документ по *MARC21Slim* шеми описаној у одељку 4.5.1. Овај темплејт започиње изразом IMPORT за увоз EMF модела записа чиме је омогућено да се у темплејту користе елементи модела: *Record, Leader, ControlField, DataField, Subfield* и *CharacterPosition*.

RecordToXML.xpt

```
«IMPORT marc21cataloguing»
«DEFINE main FOR Record»
«FILE "record.xml"»<?xml version="1.0" encoding="UTF-8"?>
<record>
<leader>«leader.content.recordLength»«leader.content.typeOfRecord» «leader.content.bibliographicLevel
 »«leader.content.typeOfControl»«leader.content.characterCodingSchene»«leader.content.indicatorCount
 »«leader.content.subfieldCodeCount»«leader.content.baseAddressOfData»«leader.content.encodingLevel
 »«leader.content.descriptiveCatalogingForm»«leader.content.multipartResourceRecordLevel
 »«leader.content.lengthOfLengthOfFieldPortion»«leader.content.lengthOfStartingCharacterPositionPortion
 »«leader.content.lengthOfStartingCharacterPositionPortion»«leader.content.undefined»</leader>«
 FOREACH controlFields AS cf»
  <controlfield tag="«cf.name.toString().subString(2,5)»">«
  FOREACH cf.characterPositions AS cPos
  »«cPos.content
  »«ENDFOREACH
  »</controlfield>«
  ENDFOREACH
  »«FOREACH dataFields AS df»
  <datafield tag="«df.name.toString().subString(2,5)»" ind1="«df.ind1»" ind2="«df.ind2»" >
  «FOREACH df.subfields AS sf
  »    <subfield code="«sf.name.toString().subString(1,2)»">«sf.content»</subfield>
  «ENDFOREACH»</datafield>«
  ENDFOREACH»
</record>
«ENDFILE»
«ENDDEFINE»
```

Слика 6.13 Темплејт за креирање XML документа

Изразом *DEFINE* дефинише се нови темплејт који има своје име, у случају фајла са слике 6.13 то је *main*. Један *xpt* фајл може садржати више темплејта, при чему је сваки темплејт креиран сопственим *DEFINE* блоком.

У изразу *DEFINE* наводи се и на који елемент модела се односи и у овом случају је то *Record.* Овим је дефинисано да је улазни параметар у темплејт истанца EMF класе *Record* која је део EMF модела записа. Приликом покретања темплејта овом улазном параметру доделиће се инстанца целог библиографског записа који је у тренутку позива темплејта отворен у генерисаном едитору.

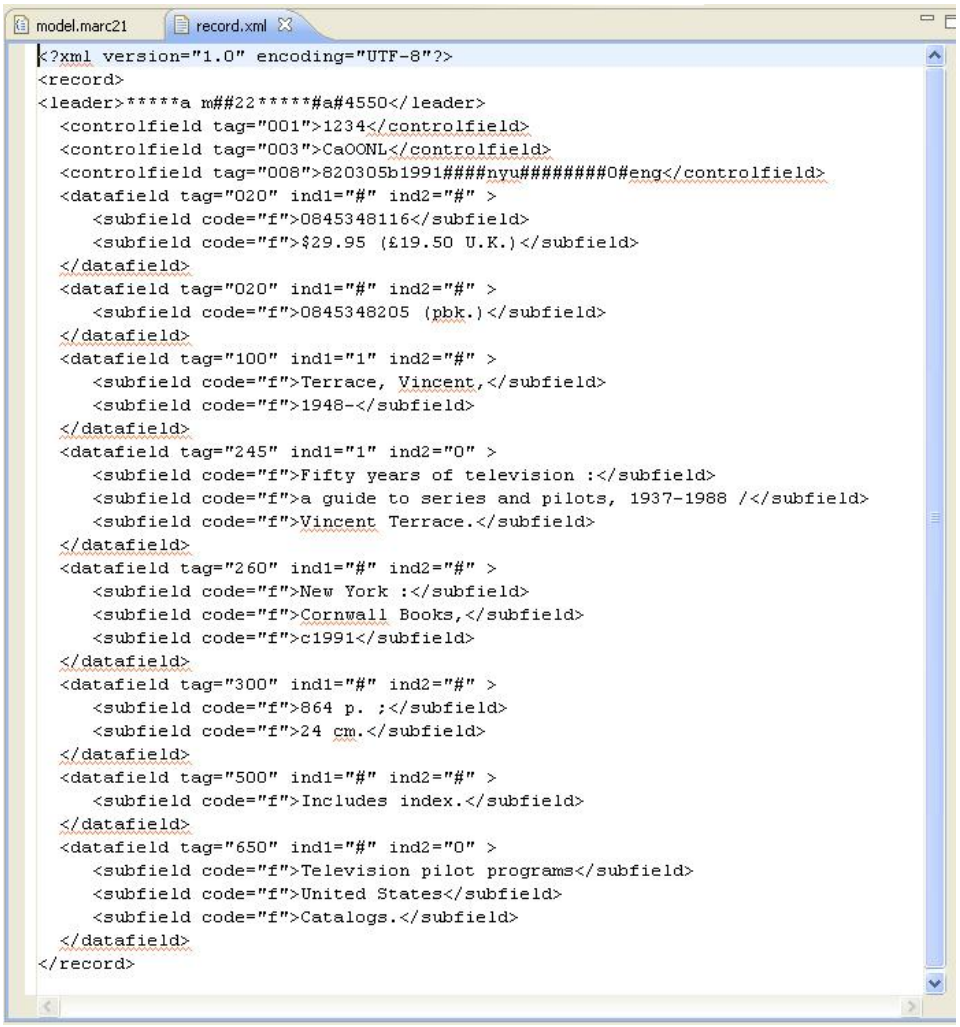
Следећи израз у темплејту на слици 6.13 је *FILE* којим је специфициран назив фајла у који ће се сместити излаз који је добијен као резултат овог темплејта.

После елемента *FILE* наводи се темплејт у виду XML стринга који представља излаз из темплејта. На оним местима где XML документ треба да се попуни елементима из записа наводе се одговарајући изрази између знакова « и ».

Елемент *leader* према *MARC21Slim* шеми садржи све елементе заглавља записа у једном стрингу. Овај случај је у темплејту решен тако што су између почетног и завршног тага елемента *leader* наведени садржаји свих појединачних елемената заглавља записа и то у редоследу који је предвиђен стандардом.

Xpand изразом *FOREACH* могуће је кретање кроз листу елемената. Овај израз је искоришћен за итерацију кроз скуп контролних поља, поља, као и карактерских позиција и потпоља.

На слици 6.14 приказан је изглед XML документа који је добијен као резултат темплејта на слици 6.13 коме је прослеђен запис са слике 6.6.


model.marc21
record.xml
<?xml version="1.0" encoding="UTF-8"?>
<record>
<leader>*****a m##22*****#a#4550</leader>
<controlfield tag="001">1234</controlfield>
<controlfield tag="003">CaOONL</controlfield>
<controlfield tag="008">820305b1991####nyu########0#eng</controlfield>
<datafield tag="020" ind1="#" ind2="#" >
<subfield code="f">0845348116</subfield>
<subfield code="f">$29.95 (£19.50 U.K.)</subfield>
</datafield>
<datafield tag="020" ind1="#" ind2="#" >
<subfield code="f">0845348205 (pbk.)</subfield>
</datafield>
<datafield tag="100" ind1="1" ind2="#" >
<subfield code="f">Terrace, Vincent,</subfield>
<subfield code="f">1948-</subfield>
</datafield>
<datafield tag="245" ind1="1" ind2="0" >
<subfield code="f">Fifty years of television :</subfield>
<subfield code="f">a guide to series and pilots, 1937-1988 /</subfield>
<subfield code="f">Vincent Terrace.</subfield>
</datafield>
<datafield tag="260" ind1="#" ind2="#" >
<subfield code="f">New York :</subfield>
<subfield code="f">Cornwall Books,</subfield>
<subfield code="f">c1991</subfield>
</datafield>
<datafield tag="300" ind1="#" ind2="#" >
<subfield code="f">864 p. ;</subfield>
<subfield code="f">24 cm.</subfield>
</datafield>
<datafield tag="500" ind1="#" ind2="#" >
<subfield code="f">Includes index.</subfield>
</datafield>
<datafield tag="650" ind1="#" ind2="0" >
<subfield code="f">Television pilot programs</subfield>
<subfield code="f">United States</subfield>
<subfield code="f">Catalogs.</subfield>
</datafield>
</record>

Слика 6.14 XML документ - Резултат темплејта

6.2.3.1 Креирање каталошких листића

Једана од веома значајних функционалности система за каталогизацију је могућност да се креирани запис прикаже кориснику у форми каталошког листића и то је део у ком Xpand темплејти налазе своју главну примену у развоју система за каталогизацију. Каталошки листић представља један начин приказа библиографског записа креираног у генерисаном едитору, а може постојати више врста каталошких листића. Формирање каталошког листића захтева могућност напреднијег форматирања излазног текста и зато је као резултујући формат изабран html. Основна идеја састоји се у томе да се за сваки жељени формат листића креира Xpand темплејт који трансформише

библиографски запис у html документ у ком се налазе библиографски подаци из записа форматирани према стандардима за креирање каталошких листића.

На слици 6.15 приказан је почетни део темплејта *RecordToCard.xpt* који трансформише библиографски запис у формат каталошког листића.

RecordToCard.xpt

```
«IMPORT marc21cataloguing»
«EXTENSION org::example::marc21::Extensions»
«DEFINE main FOR Record»
«FILE "card.html"
»<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /></head>
<body>
«IF getDataField("100").getSubfield("a")!=null»
<B>
«LET getDataField("100").getSubfield("a").content.split(" ") AS autor»
«autor.get(0).toUpperCase()» «autor.get(1)»
«ENDLET»
«IF getDataField("100").getSubfield("d")!=null»
«getDataField("100").getSubfield("d").content»
«ENDIF»
</B>
<BR><BR>   
«ENDIF»
«getDataField("245").getSubfield("a").content»
«IF getDataField("245").getSubfield("b")!=null»
«getDataField("245").getSubfield("b").content»
«ENDIF»
«IF getDataField("245").getSubfield("c")!=null»
«getDataField("245").getSubfield("c").content»
«ENDIF»
- «getDataField("260").getSubfield("a").content»
«IF getDataField("260").getSubfield("b")!=null»
«getDataField("260").getSubfield("b").content»
«ENDIF»
«IF getDataField("260").getSubfield("c")!=null»
«getDataField("260").getSubfield("c").content»
«ENDIF»
«IF getDataField("300").getSubfield("a")!=null»
.- «getDataField("300").getSubfield("a").content»
«IF getDataField("300").getSubfield("b")!=null»
«getDataField("300").getSubfield("b").content»
«ENDIF»
«IF getDataField("300").getSubfield("c")!=null»
«getDataField("300").getSubfield("c").content»
«ENDIF»
«ENDIF»
<BR><BR>
«FOREACH getDataFields("500") AS df500»
«((DataField)df500).getSubfield("a").content»
«ENDFOREACH»
```

Слика 6.15 Темплејт за креирање каталошког листића

Поред израза за увоз EMF модела записа темплејт за креирање каталошких листића садржи и израз *EXTENSION* којим се увозе проширења дефинисана над моделом написана у језику Xtend. Назив темплејта је *main*, а темплејт је специфициран за контекст *Record*. Сва проширења која су дефинисана за класу *Record* могу се позивати у темплејту самостално и односе се на објекат који је улазни параметар темплејта. Фајл у који ће бити смештен резултат темплејта је *card.html*.

Тело темплејта на слици 6.15 садржи текст у html формату у ком су на одговарајућим местима наведени изрази за преузимање података из библиографског записа. За преузимање података из библиографског записа користиће се проширења модела специфицирана у фајлу *Extensions.ext*. Једно од тих проширења је *getDataField(String dfName)* дефинисан за елемент *Record* које враћа прво поље у запису чије име одговара параметру проширења. У наставку је приказана спецификација овог проширења у језику Xtend:

```
DataField getDataField(Record rec,String dfName):
 if(rec.dataFields.select(e|e.getName()==dfName).size>0)
then
  rec.dataFields.select(e|e.getName()==dfName).get(0)
 else null;
```

Први параметар овог проширења је типа *Record,* што значи да се оно односи на елементе који су овог типа и може се позивати у облику *rec.getDataField(name)* а у темплејту на слици 6.15 ово проширење се позива самостално и односи се на улазни параметар темплејта.

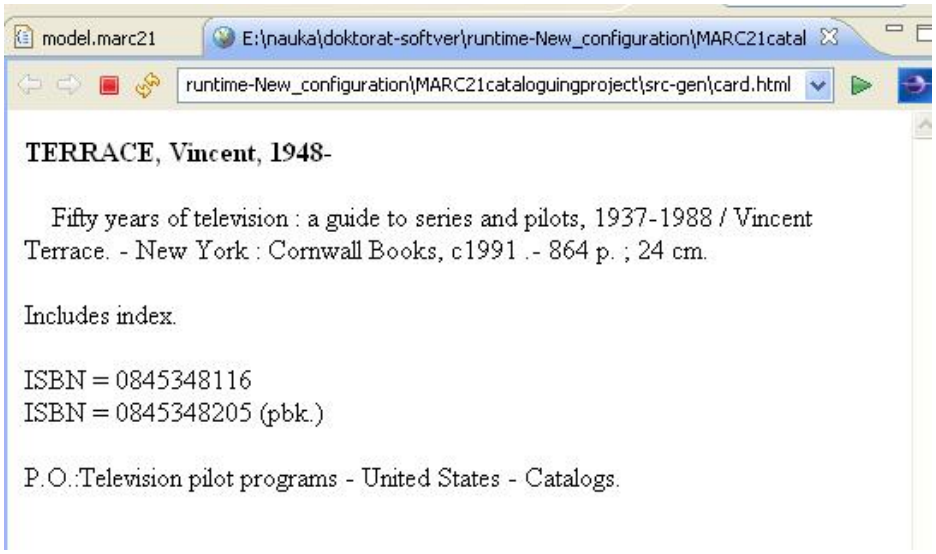
Још једно проширење које се користи у наведеном темплејту је *getSubfield(String sfName)* дефинисано је за поље и враћа његово прво потпоље са одговарајућим именом. Ту је још и проширење *getDataFields(String dfName)* која враћа листу свих поља чија имена одговарају прослеђеном параметру.

Почетни елемент на каталошком листићу је одредница која се преузима из MARC 21 поља 100 у које се уноси аутор књиге. Одредница се на листићу наводи у болдованом стилу и зато је део темплејта којим се специфицира тај део листића наведен између html елемената <B> i </B>. Одредница се формира под претпоставком да су подаци о аутору унети у облику где је прво наведено презиме, затим зарез, затим празан карактер и затим име. Xpand изразом *LET* у параметар *autor* смешта се низ стригова који је добијен када се садржај потпоља *a* поља *100* подели на местима празног карактера. У телу *LET* израза у ком је видљив параметар *autor* формира се испис аутора тако што се први елемент испише великим словима, затим се дода један html знак за празан карактер ( ) и затим се дода други елемент у изворном облику. Одредници још треба додати садржај из потпоља *d* поља *100* уколико постоји.

На сличан начин се формирају и остали елементи каталошког листића поштујући ISBD стандард за креирање листића.

На слици 6.16 приказан је резултат темплејта за креирање каталошког листића са слике 6.15 коме је прослеђен библиографски запис приказан на слици 6.6. Резултат темплејта је html документ и на слици 6.16 овај документ је отворен

у браузеру који интерпретира html тагове и формира одговарајући изглед каталошког листића.



Слика 6.16 Каталошки листић – резултат темплејта

На сличан начин се могу креирати и остали облици каталошких листића, као и неке друга врста приказа библиографског записа.

6.3 ИМПЛЕМЕНТАЦИЈА УНАПРЕЂЕНОГ СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

На основу моделирања система за каталогизацију које је описано у одељку 5.4 у оквиру треће итерације развоја система извршена је имплементација у plug-in технологији Eclipse-a. У овом одељку описана је имплементација свих додатних функционалности које представљају имплементацију једног дела случајева коришћена који се представљени на слици 5.1. У ове додатне функционалности система за каталогизацију спадају:

- приказ података о MARC 21 формату (случајеви коришћења *Show MARC 21 format view, Show bibliographic format view* и *Show holdings format view*),
- унос локацијских података (случај коришћења *Create holdings record*),
- експорт и импорт MARC 21 записа (случајеви коришћења *Store record*, *Select record*),
- библиотечко окружење (случај коришћења *Create record by processing type*) и
- приказ каталошких листића (случај коришћења *Show catalogue card*).

Почетни корак у имплементацији сваке од додатних функционалности је креирање проширења за plug-in који је у претходним итерацијама генерисан у Xtext окружењу. Та проширења су дефинисана над стандардним компонентама Eclipse корисничког интерфејса, односно над plug-in-овима у оквиру Eclipse-а којима су реализоване те компоненте. За реализацију функционалности које захтевају проширења система за каталогизацију специфицирају се тачке проширења.

Свака од функционалности којима је допуњен основни едитор захтева неке промене у корисничком интерфејсу, а ове промене имплементиране су коришћењем стандардних компоненти Eclipse корисничког интерфејса:

- приказ података о MARC 21 формату – креиран као посебан поглед у ком су подаци о MARC 21 формату приказани у облику стабла;
- унос локацијских података – имплементиран као акција којом се креира нови запис за локацијске податке који се повезује са матичним библиографским записом;
- експорт и импорт библиографских записа – имплементирани као акције којима се библиографски запис преузима из едитора и складишти у базу података односно преузима из базе и учитава у едитор; имплементирана је и тачка проширења за plug-in којом се ова функционалност може проширити од стране неког другог система додавањем комуникације са конкретном базом података;
- библиотечко окружење – реализовано као визард из ког се бира тип обраде на основу ког се креира иницијални библиографски запис који се и отвара у едитору; у оквиру ове функционалности имплементирана је тачка проширења којом се plug-in може проширити од стране неког другог система у ком ће се реализовати креирање и складиштење типова обраде;
- приказ каталошких листића – имплементиран као акција која покреће извршавање темплејта за креирање каталошких листића и фајл који се добије као резултат темплејта приказује у екранској форми Eclipse-а;

Све имплементиране акције доступне су преко *toolbar*-а апликације и у менију *Cataloguing* који је такође креиран као проширење plug-in-а Eclipse-а.

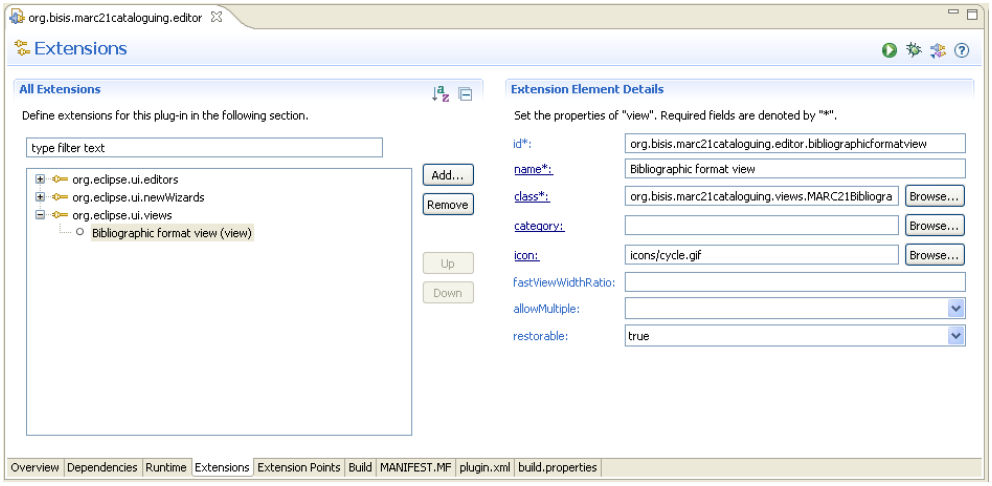
Због тога што се систем за каталогизацију користи као део Eclipse корисничког интерфејса све лабеле које су део система за каталогизацију наведене су такође на енглеском. Eclipse има подршку за интернационализацију лабела преко издвајања стрингова изван апликације и на тај начин се ова апликација може превести на било који језик.

Имплементација свих наведених компоненти започиње спецификацијом у Eclipse Plug-in Manifest едитору. Након тога следи имплементација одговарајућих класа на основу модела који је описан у одељку 5.4. У наставку ће бити описан поступак имплементације сваке од наведених функционалности.

6.3.1 Имплементација приказа библиографског формата

На слици 6.17 приказана је спецификација проширења којом ће се додати нови поглед на екранску форму Eclipse-a. Овај поглед приказује податке о MARC 21 библиографском формату и представља имплементацију случаја коришћења *Show bibliographic format*.

Да би се додао нови поглед plug-in-у дефинише се прошерење за тачку проширења *view* plug-in-а *org.eclipse.ui.views* који је део стандардне Eclipse платформе за креирање корисничког интерфејса. На десној страни екранске форме која је приказана на слици 6.17 уносе се подаци за спецификацију овог проширења. Важан податак је *id* проширења на основу ког се ова компонента може референцирати у даљој имплементацији.



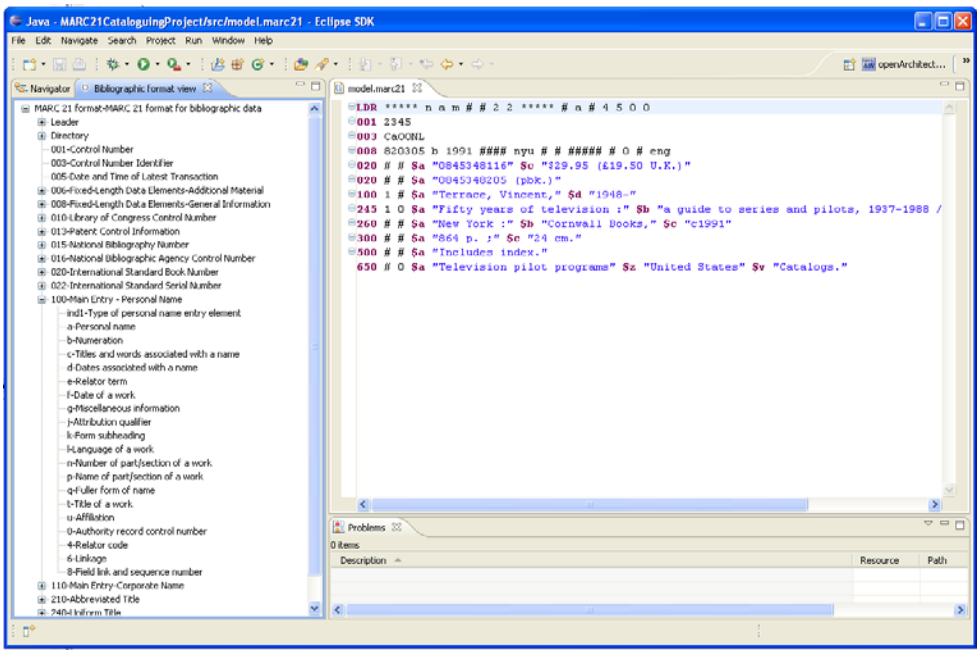
Слика 6.17 Креирање новог погледа за приказ библиографског формата

Назив проширења је *Bibliographic format view*, а класа којом је имплементиран овај поглед је *MARC21BibliographicFormatView* која је део пакета *Viewes* (слика 5.4) и приказана је на дијаграму класа на слици 5.5.

Имплементација класе *MARC21BibliographicFormatView* састоји се у имплементацији динимике која је описана дијаграмом секвенци *ShowBibliographicFormatView* који је приказан на слици 5.6 и описан у одељку 5.4.2, а као резултат имплементације добија се поглед који је приказан на

слици 6.18 као део Eclipse прозора (лева страна екранске форме). У овом погледу приказано је стабло MARC 21 библиографског формата које је креирано на основу XML документа описаног у одељку 4.3.3. Увођењем погледа за приказ библиографског формата кориснику који врши каталогизацију је омогућен увид у спецификацију формата односно правила за каталогизацију. Сада корисник у току едитирања записа има увек доступну помоћ која знатно олакшава процес креирања записа.

На потпуно исти начин креиран је и поглед за приказ података о формату за обраду локацијских података – *Holdings format view* који представља имплементацију случаја коришћења *Show holdings format view*. Овај поглед приказиваће се приликом уноса записа за локацијске податке. Имплементација функционалности креирања овог погледа реализована је на основу дијаграма секвенци са слике 5.7.



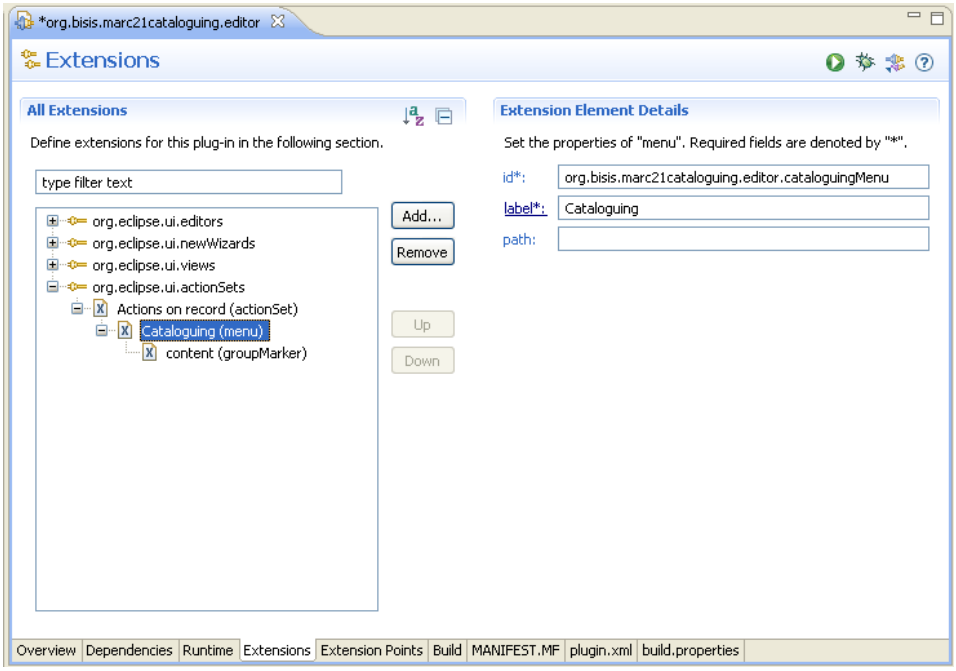
Слика 6.18 Приказ библиографског формата

6.3.2 Имплементација уноса локацијских података

Унос локацијских података за библиографски запис (случај коришћења *Create holdings record*) имплементиран је као акција која ће бити доступна као дугме на *toolbar*–у и као ставка у менију *Cataloguing* који ће се такође креирати у овом кораку имплементације и касније користити и за остале акције.

Приликом спецификације прве акције за plug-in прво се дефинише скуп акција (*action set*) под именом *Actions on record*. Скуп акција представља проширење plug-in-a *org.eclispe.ui.actionSet*.

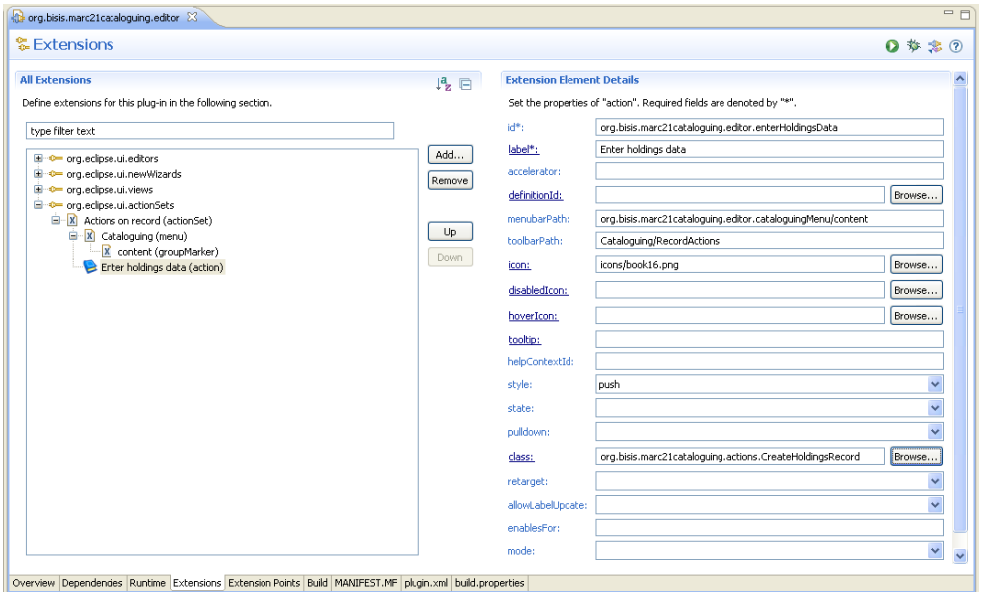
У оквиру скупа акција прво се специфицира мени *Cataloguing* у који ће се смештати све акције које припадају систему за каталогизацију. Да би се нови мени додао у мени линију Eclipse-a потребно је само извршити одређену спецификацију, никаква додатна имплементација није потребна. Та спецификација састоји се у креирању новог менија за скуп акција и додавања елемента *content* том менију. На слици 6.19 приказана је спецификација менија *Cataloguing* у Eclipse Manifest едитору. За мени се специфицира *id* и лабела која ће се појавити у мени линији апликације. У оквиру менија специфициран је елемент *content* који је типа *groupMaker* у који ће се додавати акције које припадају креираном менију.



Слика 6.19 Креирање менија *Cataloguing*

Следећи корак је спецификација конкретне акције *Create holdings record* којом ће се креирати запис за локацијске податке. На слици 6.20 приказана је спецификација ове акције. На десној страни екранске форма приказане на слици 6.20 унети су подаци који чине спицификацију ове акције. Унет је *id* акције, лабела која ће се појавити у менију, затим путања до менија и путања на *toolbar*-у где ће се налазити ова акција, иконица којом ће бити обележена као и класа у којој ће бити имплементирана обрада ове акције. Класа у којој се

обрађује акција за унос локацијских података је класа *CreateHoldingsRecord* која је имплементација истоимене класе из модела која се налази у пакету *Actions* и приказана је на дијаграму класа којим је описан статички модел уноса ликацијских података (слика 5.8). Имплементација креирања записа за локацијске податке и његово отварање у едитору извршена је на основу динамичког модела који је приказан дијаграмом секвенци *EnterHoldingsData* на слици 5.9 и описан у одељку 5.4.3.

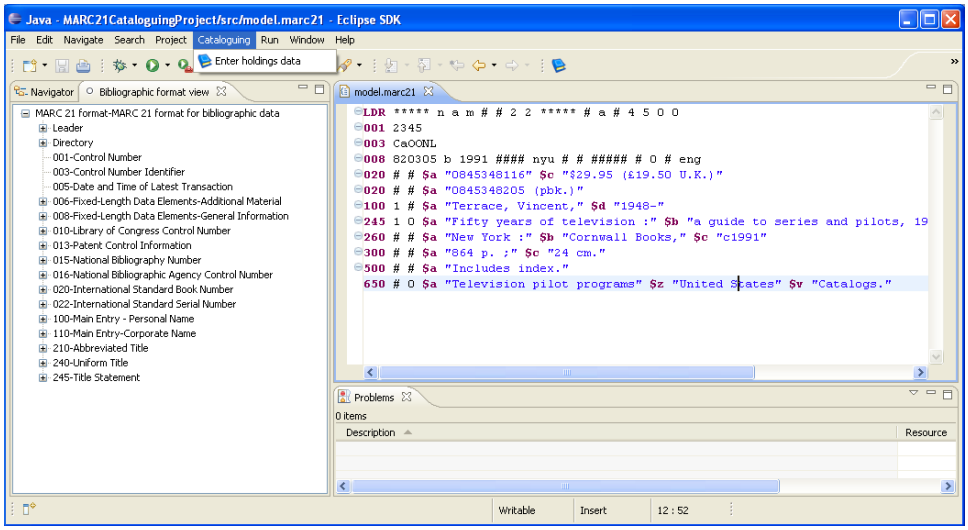


Слика 6.20 Спецификација акције за унос локацијских података

На слици 6.21 приказан је изглед Eclipse прозора у ком је доступна акција за унос локацијских података. Акција је доступна из менија *Cataloguing* и са *toolbar*-а преко дугмета са иконицом .

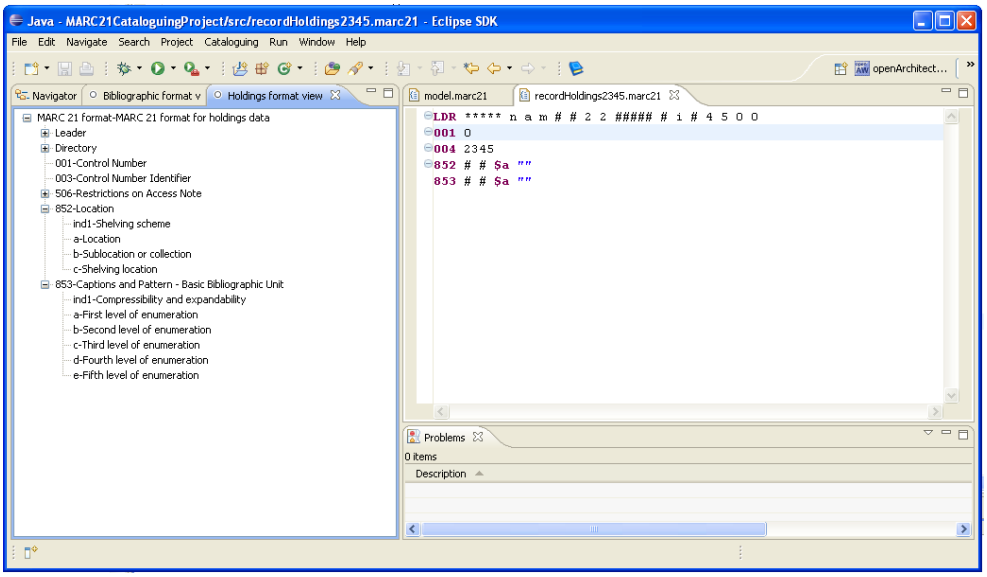
Покретањем акције за унос локацијских података креира се нови запис за локацијске податке који ће бити повезан са библиографским записом који је отворен у едитору. Та веза се остварује преко контролног броја библиографског записа који се налази у контролном пољу *001*. За библиографски запис који је приказан на слици 6.18 контролни број је *2345*. Покретањем акције за унос локацијских података добија се екранска форма приказана на слици 6.22. У овој екранској форми приказан је едитор у коме је активан фајл *recordsHoldings2345.marc21* који представља креирани запис за локацијске податке. У контролном пољу *004* овог записа налази се број *2345* који даје везу са библиографским записом. Поред тога у запис су додата поља која се односе на локацијске податке (852, 853), а приказан је и поглед

Holdings format view у ком се налази стабло MARC 21 формата за локацијске податке (лева страна екранске форме на слици 6.22).



Слика 6.21 Акција за унос локацијских података

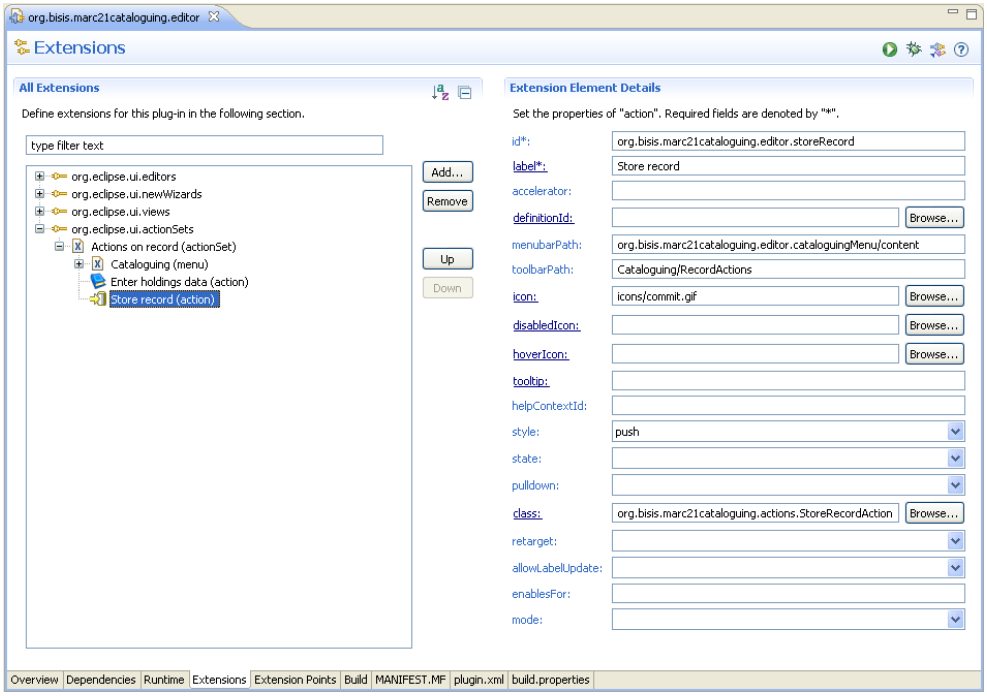
Креирани запис за локацијске податке се обрађује на исти начин као и библиографски запис и над њим се могу извршавати све акције које су доступне за рад са библиографским записом.



Слика 6.22 Креирање записа за локацијске податке

6.3.3 Имплементација експорта и импорта библиографских записа

На слици 6.23 приказана је спецификација проширења којом се реализује нова акција за експорт, односно складиштење записа који су креирани у едитору (имплементација случаја коришћења *Store record*). Поред идентификатора акција и лабеле, овде се специфицира и путања у менију и *toolbar*-у на којој ће се налазити ова акција, иконица којом ће бити обележена акција и класа у којој је имплементирана обрада ове акције.

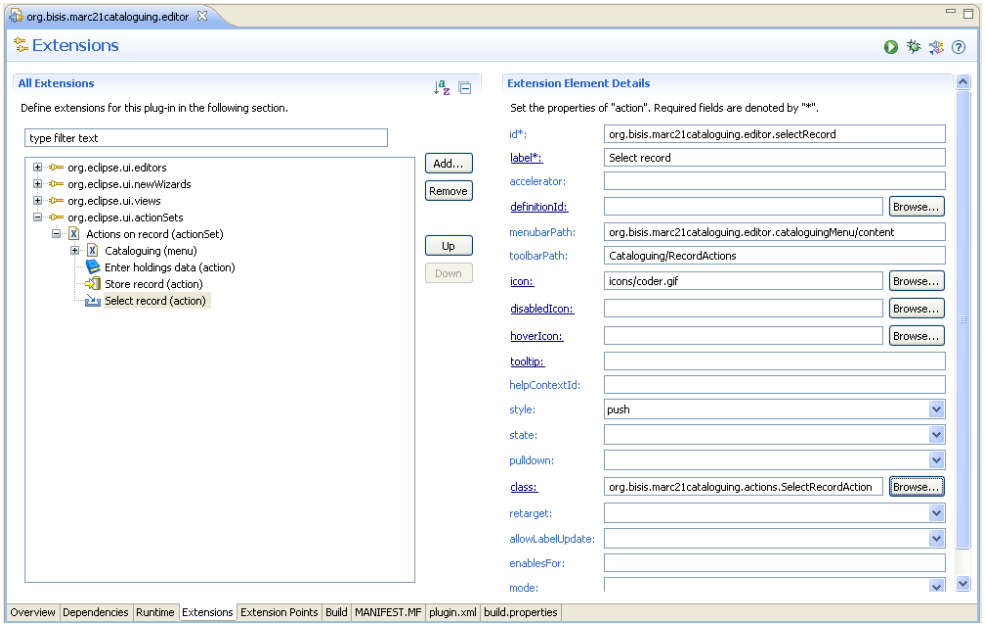


Слика 6.23 Спецификација акције за складиштење записа

Обрада акције за складиштење записа имплементирана је у класи *StoreRecordAction* која је креирана на основу истоимене класе у моделу. Класа *StoreRecordAction* у моделу система за каталогизацију припада пакету *Actions* и приказана је на дијаграму класа који описује статичке особине експорта и импорта записа (слика 5.10, одељак 5.4.4). Имплементација процеса складиштења записа извршена је на основу динамичког модела који је описан дијаграмом секвенци *StoreRecord* (слика 5.11).

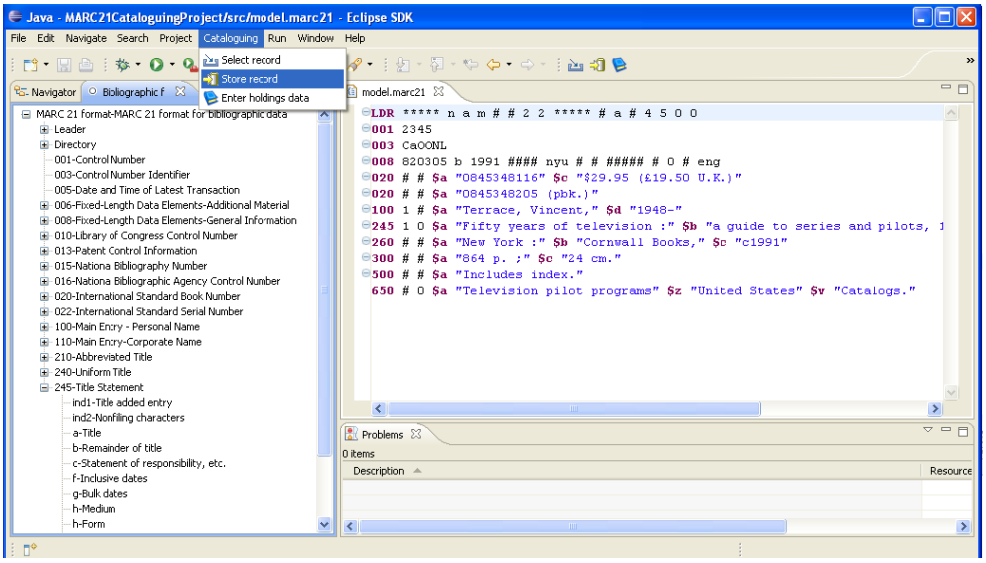
На слици 6.24 приказана је спецификација акције за селектовање записа и његово учитавање у едитор (имплементациај случаја коришћења *Select record*). У овој спецификацију наводе се параметри исти као и за складиштење записа, а класа којом је имплементирана посматрана акција је

SelectRecordAction. И ова класа припада пакету *Action* у моделу система за каталогизацију и приказана је на дијаграму класа који описује експорт и импорт записа. Имплементација процеса учитавања записа на основу његовог ID броја извршена је на основу динамичког модела који је описан дијаграмом секвенци *SelectRecord* (слика 5.12).



Слика 6.24 Спецификација акције за учитавање записа

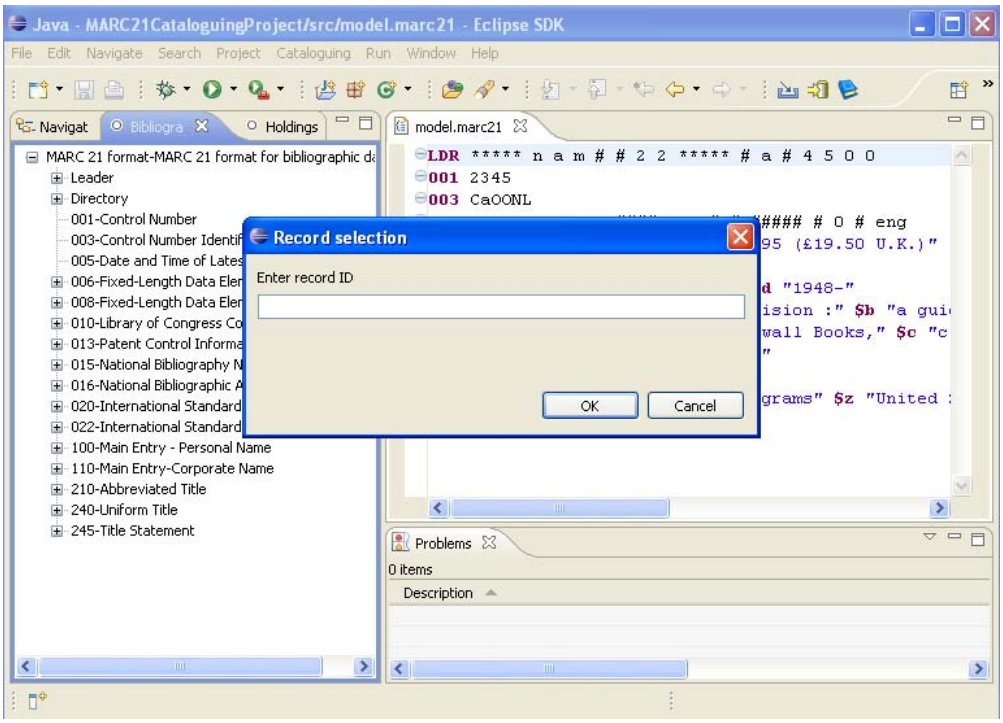
На слици 6.25 приказана је екранска форма Eclipse-a у којој су доступне акције за складиштење и учитавање записа. Акције су доступне са *toolbar*-а апликације и у менију *Cataloguing*.



Слика 6.25 Акције за складиштење и учитавање записа

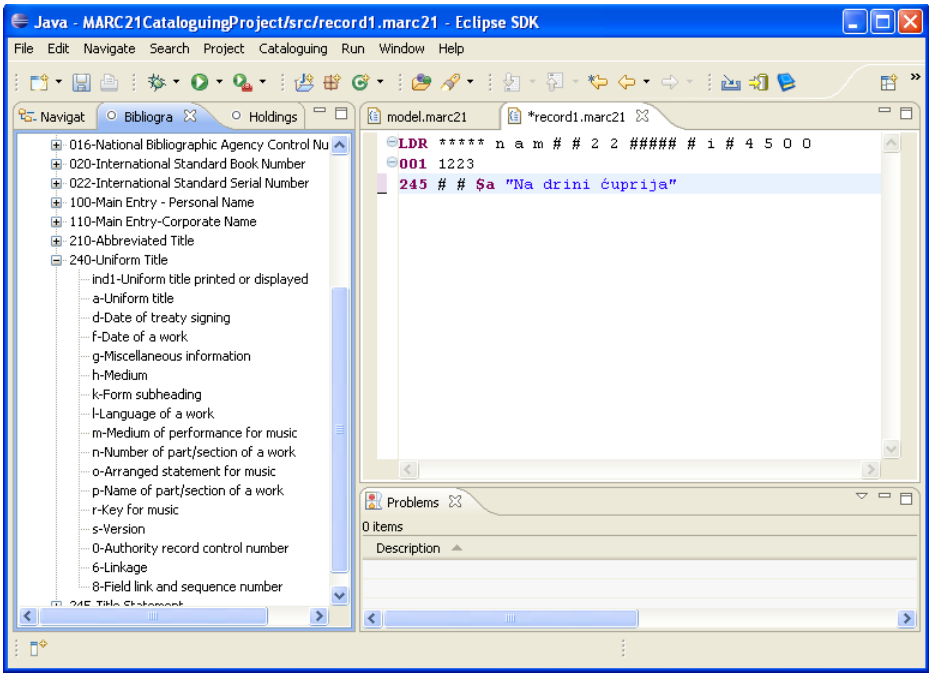
Позивом акције за складиштење записа која је означена иконицом  запис који је отворен у едитору учитава се у објектни модел, а затим из објектног модела серијализује у XML документ и смешта у једну врсту табеле у бази при чему му се додељује ID број. Након извршеног складишта кориснику се пријављује порука о успешности извршавања.

Позивом акције за учитавање записа која је означена иконицом  отвара се помоћни прозор у који корисник уноси ID број записа (слика 6.26). Након уноса ID броја запис са тим бројем се учитава из базе у објектни модел записа, затим се објектни модел серијализује у фајл са екстензијом *marc21* и то према правилима које су прописана Xtext граматиком и тај новокреирани фајл се отвара у едитору за каталогизацију.



Слика 6.26 Селекција записа на основу ID броја

На слици 6.24 приказан је запис који је учитан из базе после селекције записа чији је ID једнак 1. Овај запис садржи заглавље, контролно поље *001* и поље *245* чије потполе *a* садржи наслов публикације *Na drini ćuprija*. Запис који је учитан из базе снимљен је у фајл *record1.marc21* који је отворен у едитору на слици 6.27.

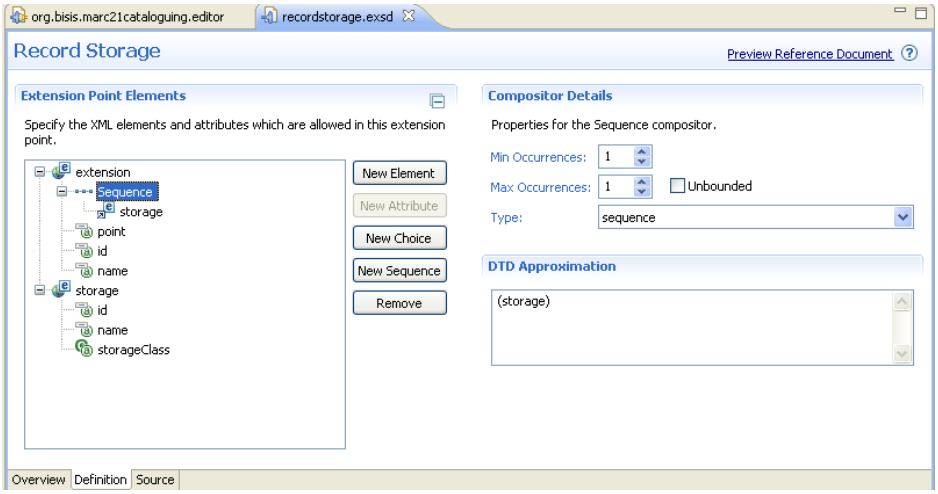


Слика 6.27 Учитавање записа у едитор

6.3.3.1 Спецификација тачке проширења за складиште записа

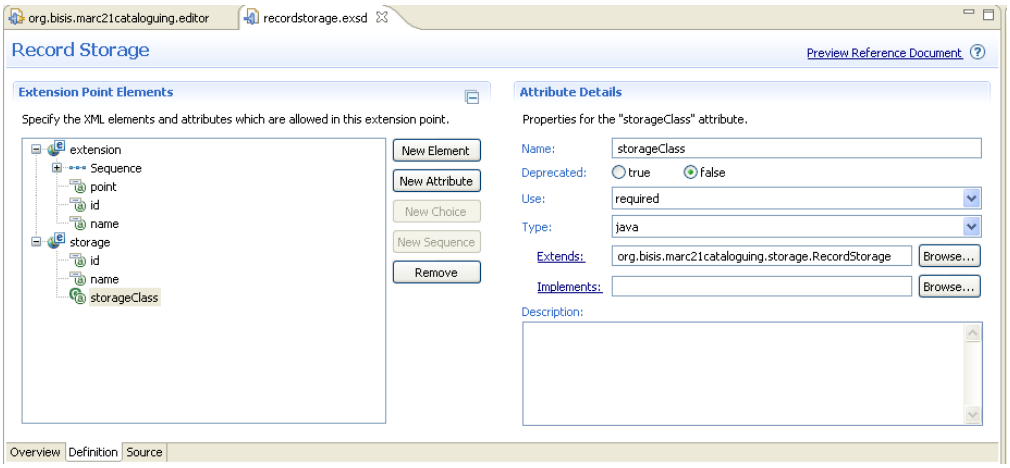
Посматрани систем за каталогизацију реализован је као софтверска компонента која може да се интегрише у различите софтверске системе. Он је реализован као plug-in за Eclipse који може да се користи у оквиру неке друге апликације која је такође реализована у plug-in технологији. Приликом интеграције система за каталогизацију остављена је могућност систему у који се врши интеграција да имплементира складиштење и учитавање записа у сопствену базу података или индекс за претраживање. Ова проширивост система омогућена је архитектуром Eclipse-a која омогућава креирање тачака проширења plug-in-a.

За plug-in *org.bisis.marc21cataloguing.editor* дефинисана је тачка проширења *recordstorage* којом се систем за каталогизацију може проширити имплементацијом складишта записа. На слици 6.28 приказана је спецификација шеме ове тачке проширења у Eclipse едитору за шему тачке проширења. Елемент *extension* ове тачке проширења се састоји од секвенце у којој се појављује најмање и највише један елемент *storage*. Поред тога елемент *extension* има и два атрибута која су дефинисана за све тачке проширења, то су обавезни атрибут *point* и опциони атрибути *id* и *name*.



Слика 6.28 Спецификација тачке проширења *recordstorage*

Елементом *storage* спецификацирана је тачка проширења за складиштење записа. За овај елемент дефинисана су два опциона атрибута *name* и *id* који представљају редом назив и ID складишта и један обавезни атрибут *storageClass*.



Слика 6.29 Спецификација атрибута *storageClass*

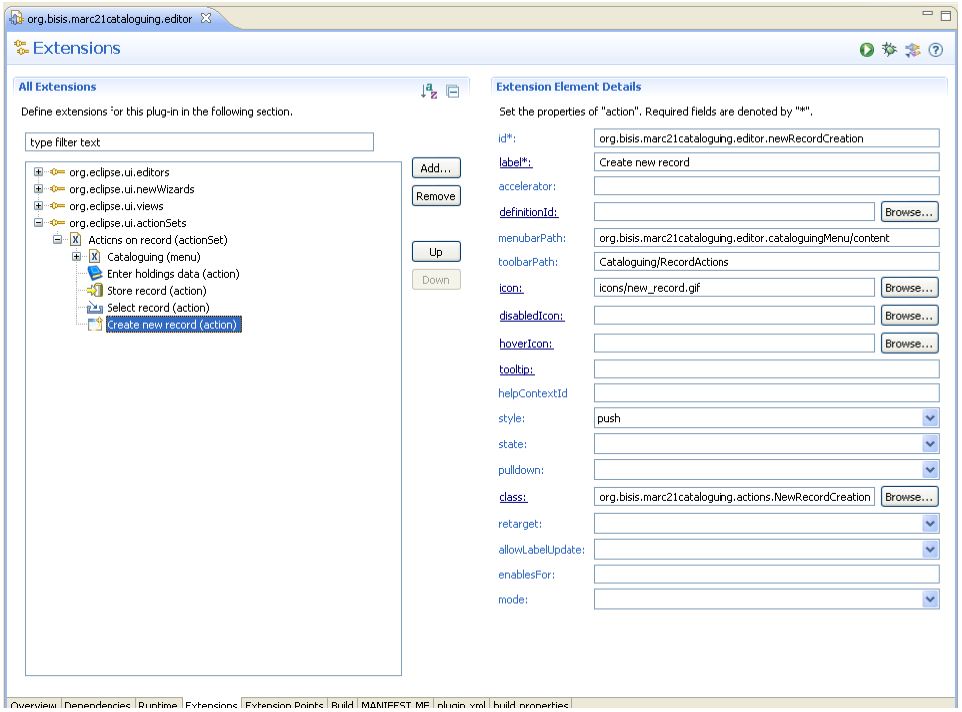
На слици 6.29 приказана је спецификација атрибута *storageClass*. Тип овог атрибута је *java* што значи да се ради о Јава класи. У параметру *Extends* наведен је пун назив класе коју треба да наследи класа која ће се јавити као вредност овог атрибута, а то је класа *RecordStorage* која је описана у моделирању експорта и импорта записа (одељак 5.4.4). Ради се о апстрактној класи која дефинише операције за складиштење и учитавање објектног модела записа. На овај начин специфицирана је тачка проширења преко које систем за

каталогизацију може да буде проширен са класом која наслеђује апстрактну класу *RecordStorage* и у којој је имплементирана комуникација са базом података.

Програмски код који је подршка за тачку проширења складишта записа реализован је у оквиру операције *init()* класе *StorageFactory* која је такође описана у моделирању експорта и импорта записа (одељак 5.4.4). Да би систем за каталогизацију користио проширење потребно је проверити да ли постоји проширење за посматрану тачку проширења и ако постоји инстанцирати одговарајућу класу која је садржај атрибута *storageClass* елемента *storage*. Овако инстанцирана класа користиће се за комуникацију са базом, односно приликом складиштења и учитавања записа.

6.3.4 Имплементација библиотечког окружења

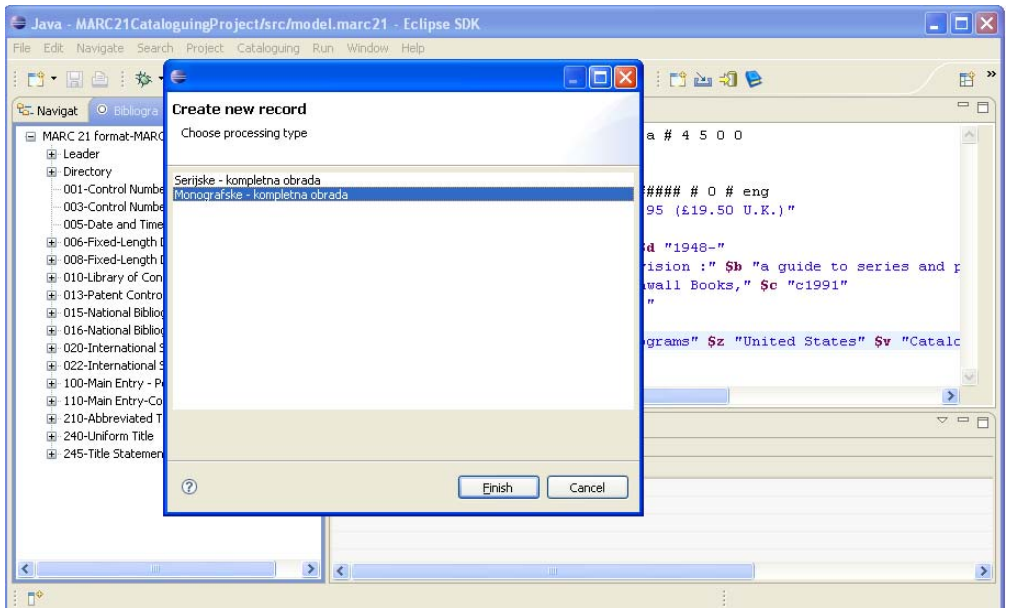
У оквиру система за каталогизацију имплементирана је акција за креирање новог записа на основу изабраног типа обраде (случај коришћења *Create record by process type*). Спецификација ове акције приказана је на слици 6.30. Лабела којом је означена акција је *Create new record*, а класа у којој је она имплементирана је класа *NewRecordCreation*. Ова класа представља имплементацију истоимене класе из пакета *Actions* у моделирању библиотечког окружења (одељак 5.4.5).



Слика 6.30 Спецификација акције за креирања новог записа

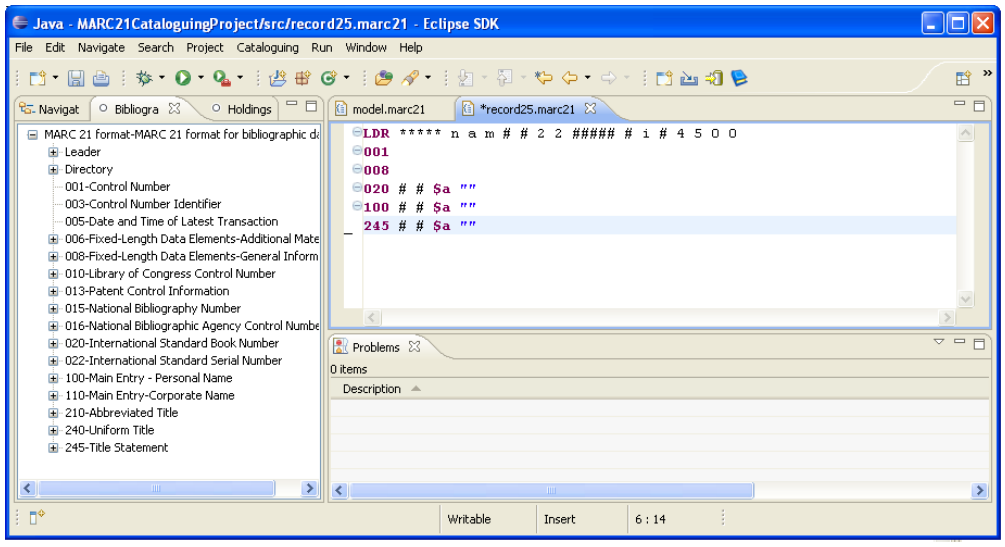
Креирање новог записа на основу изабраног типа обраде имплементирано је као визард на основу динамичког модела који је описан у 5.4.5. Притиском на дугме са иконицом на *toolbar*-у за каталогизацију или из менија *Cataloguing* покреће се визард за избор типова обраде који је приказан на слици 6.31.

Визард који је приказан на слици 6.31 садржи листу назива типова обраде који су учитани из неког каталога типова. Корисник бира један од понуђених типова обраде и притиска дугме *Finish* за завршетак визарда. По завршетку рада визарда, на основу изабраног типа обраде креира се иницијални запис који се отвара у едитору за каталогизацију.



Слика 6.31 Визард за избор типа обраде

На основу изабраног типа обраде креира се фајл са екстензијом *marc21* у ком се налази запис који садржи поља и потпоља дефинисана изабраним типом обраде. На слици 6.32 приказан је тај креирани фајл отворен у едитору за каталогизацију. Према изабраном тупу обраде запис садржи заглавље, контролна поља *001* и *008* и поља *020*, *100* и *245*. Шифре које се налазе у заглављу записа су такође специфициране типом обраде.

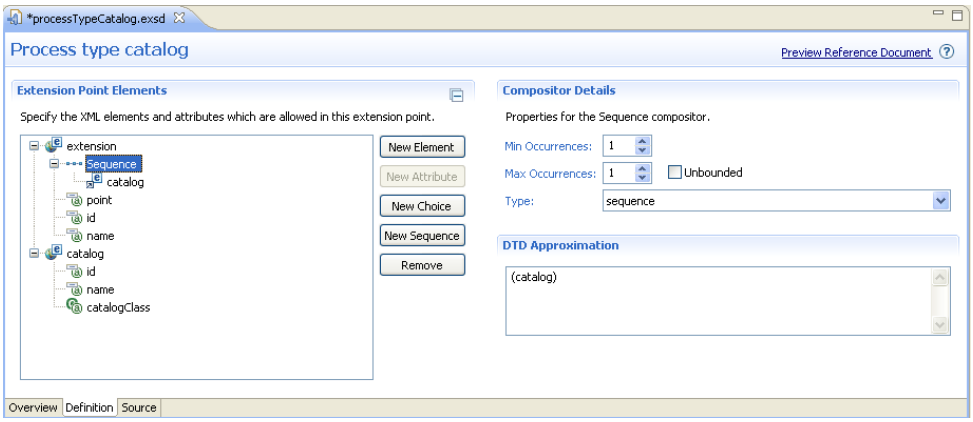


Слика 6.32 Креирање новог записа на основу изабраног типа обраде

6.3.4.1 Спецификација тачке проширења за учитавање типова обраде

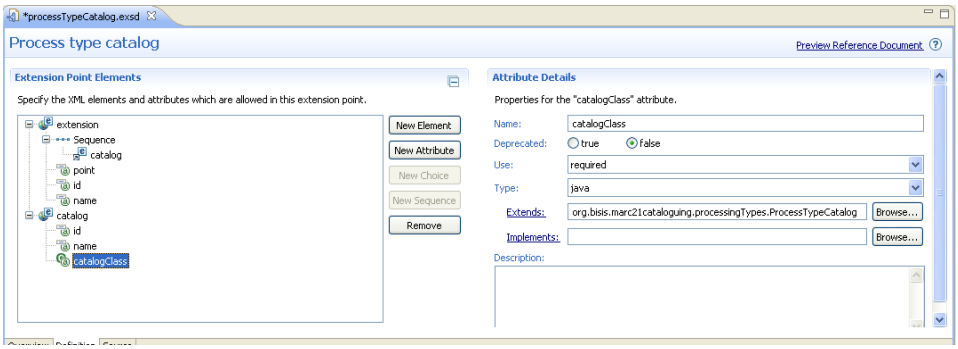
Слично као приликом складиштења и учитавања записа и библиотечко окружење је у оквиру система за каталогизацију имплементирано као прошириво коришћењем тачке проширења plug-in-a. Приликом интеграције система за каталогизацију оставља се могућност да се у оквиру система у који се систем за каталогизацију интегрише имплементира учитавање каталога типова обраде. Претпоставка је да ће се систем за каталогизацију интегрисати у неки библиотечки систем у ком ће бити имплементирана компонента за креирање типова обраде. Коришћење система за каталогизацију нема никаква ограничења на начин складиштења типова обраде. Ова особина је постигнута увођењем тачке проширења у којој се специфицира шта треба други систем да имплементира да би се приликом креирања визарда учитали типови обраде из каталога типова обраде који је генерисан у оквиру тог система.

За plug-in *org.bisis.marc21cataloguing.editor* поред тачке проширења *recordstorage* специфицирана је још једна тачка проширења *processTypeCatalog*. На слици 6.33 приказана је спецификација шеме за ову тачку проширења. Елемент *extension* садрши три стандардна атрибута: *point*, *id*, *name* и секвенцу од манимално једног и максимално једног појављивања елемента *catalog*.



Слика 6.33 Спецификација тачке проширења *processTypeCatalog*

Елемент *catalog* представља спецификацију тачке проширења за креирање каталога типова обраде. Елемент *catalog* има два опциона атрибута, то су *id* у ком се специфицира идентификатор каталога и *name* у ком се специфицира назив каталога. Поред ова два опциона атрибута елемент *catalog* има и обавезни атрибут *catalogClass*. Спецификација овог атрибута приказана је на слици 6.34. Посматрани атрибут означен је као обавезан а тип је *java*. Вредност овог атрибута је Јава класа која наслеђује класу *ProcessTypeCatalog*.



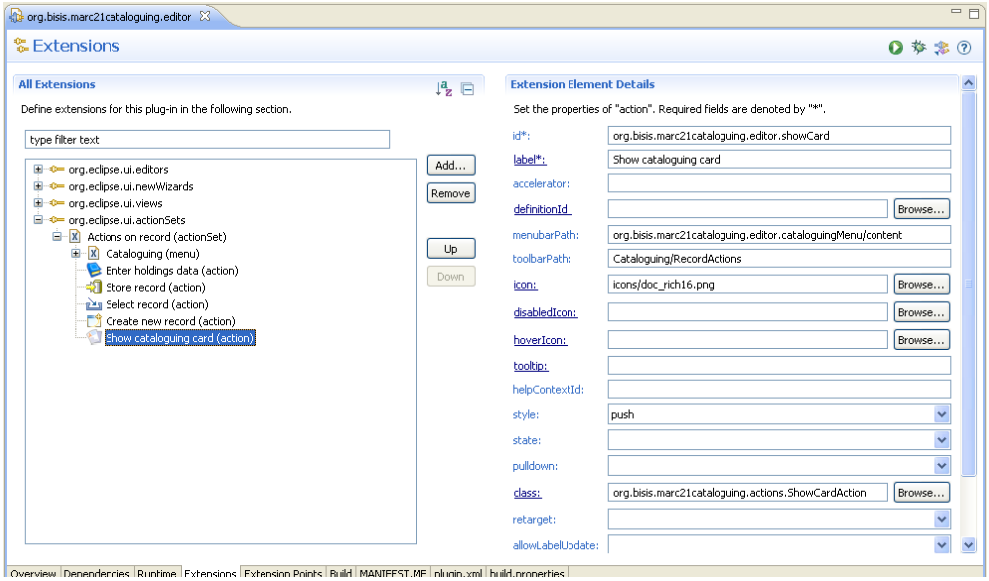
Слика 6.34 Спецификација атрибута *catalogClass*

Класа *ProcessTypeCatalog* описана је у моделирању библиотечког окружења (одељак 5.4.5). Ради се о апстрактној класи која дефинише операције за креирање каталога типова обраде. На овај начин специфицирана је тачка проширења преко које систем за каталогизацију може да буде проширен са класом која наслеђује апстрактну класу *ProcessTypeCatalog* и у којој је имплементирано креирање каталога типова обраде који ће се понудити кориснику у визарду за креирање новог записа на основу типа обраде.

Програмски код који је подршка за тачку проширења *processTypeCatalog* складишта записа реализован је у оквиру операције *init()* класе *CatalogFactory* која је такође описана у моделирању (одељак 5.4.5) библиотечког окружења. Да би систем за каталогизацију користио проширење потребно је проверити да ли постоји проширење за посматрану тачку проширења и ако постоји инстанцирати одговарајућу класу која је садржај атрибута *catalogClass* елемента *catalog*. Овако инстанцирана класа користиће се приликом покретања визарда за нови запис за формирање листи доступних типова обраде.

6.3.5 Имплементација приказа каталошког листића

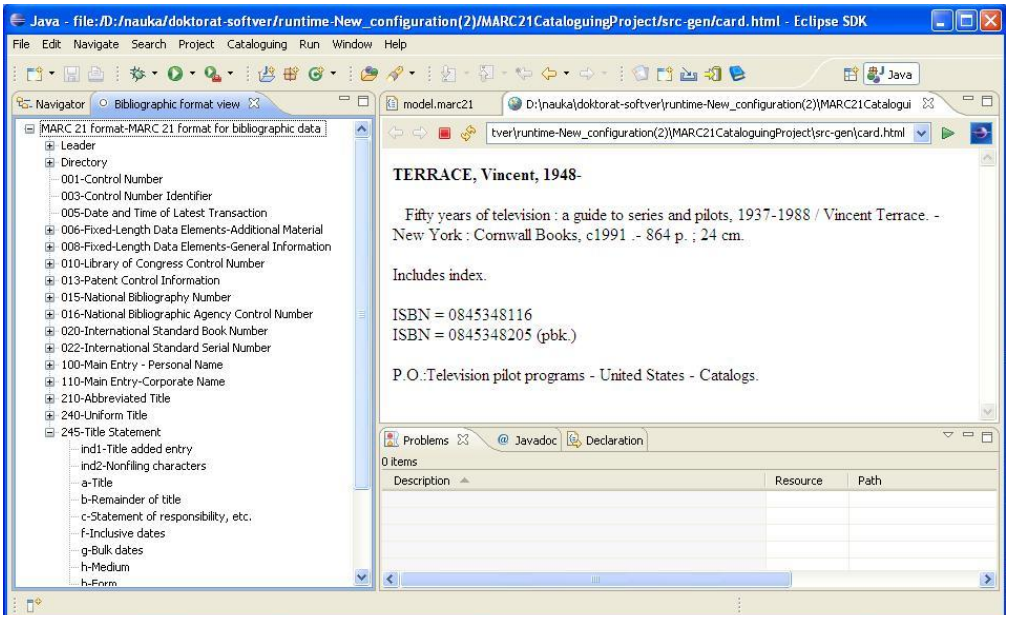
На слици 6.35 приказана је спецификација проширења за акцију креирања и приказа каталошког листића (имплементација случаја коришћења *Show catalogue card*). Лабела за акцију је *Show cataloguing card* а класа којом је имплементирано извршавање ове акције је *ShowCardAction*. Ова класа је имплементација истоимене класе из пакета *Actions* која је описана у моделирању приказа каталошког листића (одељак 5.4.6). Имплементација процеса креирања каталошког листића на основу записа који се обрађује у едитору извршена је на основу динамичког модела који је представљен дијаграмом секвенци у одељку 5.4.6.



Слика 6.35 Спецификација акције за приказ каталошког листића

За имплементацију акције за приказ каталошког листића коришћени су темплејти и генератор који се налазе у plug-in-у *org.bisis.marc21cataloguing.generator*. Овај plug-in је имплементација подсистема *Templates* из моделирања система (слика 5.4). Генератор којим се покреће темплејт је имплементиран као *workflow* фајл односно као XML конфигурациони фајл, а темплејти за креирање листића су креирани у језику Xpand.

Притиском на дугме са иконицом на *toolbar*-у за каталогизацију или из менија *Cataloguing* креира се каталошки листић за библиографски запис који је отворен у едитору. Каталошки листић се креира у виду *html* документа и отвара се у браузеру у оквиру Eclipse апликације. На слици 6.36 приказан је листић отворен у браузеру у оквиру Eclipse окружења.



Слика 6.36 Каталошки листић

Поглавље 7

RCP апликација система за каталогизацију

Систем за каталогизацију који је генерисан у Xtext технологији и унапређен у plug-in технологији доступан је у облику plug-in-а за Eclipse, што значи да се у том облику може користити само у оквиру Eclipse окружења.

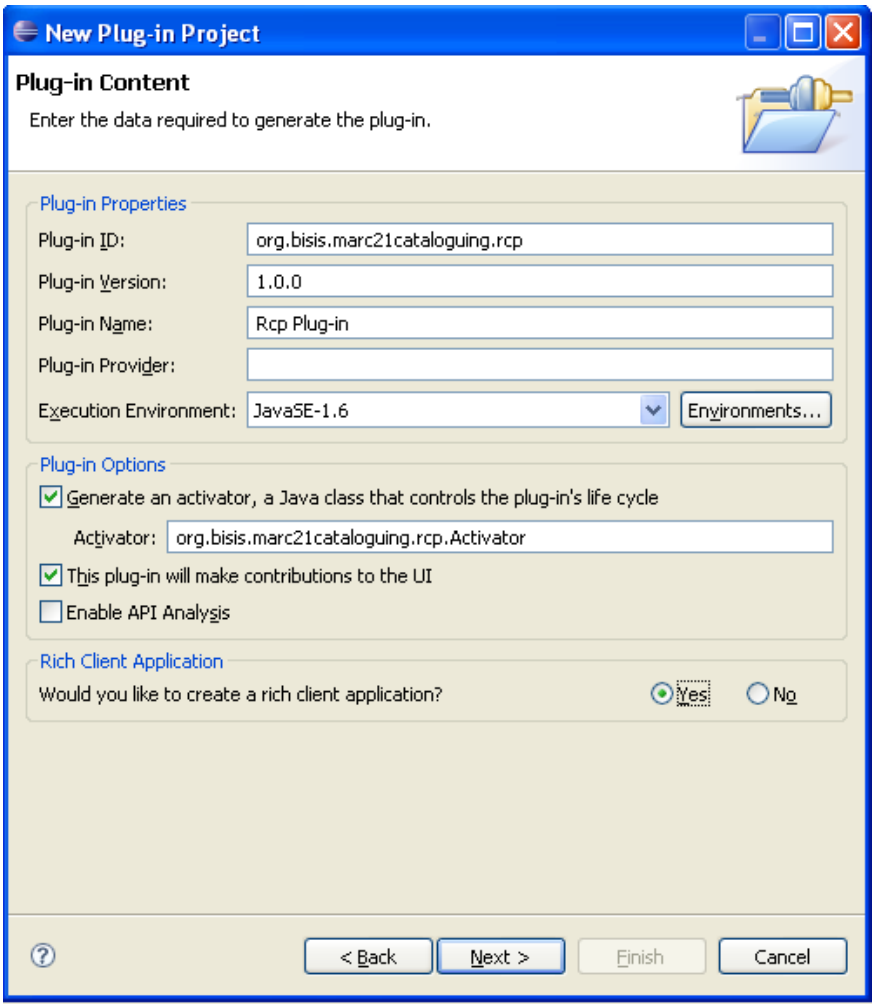
Једно решење којим би се систем за каталогизацију могао издвојити из Eclipse-а је коришћењем технологије Rich Client Platform - RCP [31]. RCP представља технологију за генерисање независних софтверских апликација на основу скупа Eclipse plug-in-ова. RCP апликација представља подскуп plug-in-ова Eclipse-а, а сам израз Rich Client Platform односи се на минималан скуп plug-in-ова неопходних да се направи једна RCP апликација.

На основу Eclipse plug-in-ова који су генерисани софтверским алатом Xtext и унапређени додатним функционалностима у plug-in технологији може се генерисати самостална апликација коришћењем RCP технологије.

Основна идеја приликом креирања RCP апликације система за каталогизацију састоји се у томе да се креира нови plug-in (RCP plug-in) који ће бити главни програм те апликације и који ће користити генерисане и унапређене plug-in-ове система за каталогизацију и још додатних plug-in-ова из Eclipse платформе.

7.1 КРЕИРАЊЕ RCP АПЛИКАЦИЈЕ

Да би се креирала једна RCP апликација почетни корак је креирање plug-in-а који ће бити главни програм апликације. На слици 7.1 приказана је једна страница визард за генерисање RCP plug-in-а *org.bisis.marc21cataloguing.rcp*. На овој страници је наведен *id* plug-in-а, верзија, назив, верзија Јава извршног окружења која се користи и слично. Да би се на основу овог plug-in-а могла направити RCP апликација потребно је чекирати опцију *Would you like to create a rich client application?*. На овај начин остатак визарда за креирање новог plug-in ће понудити додатна подешавња која се односе на RCP апликацију.

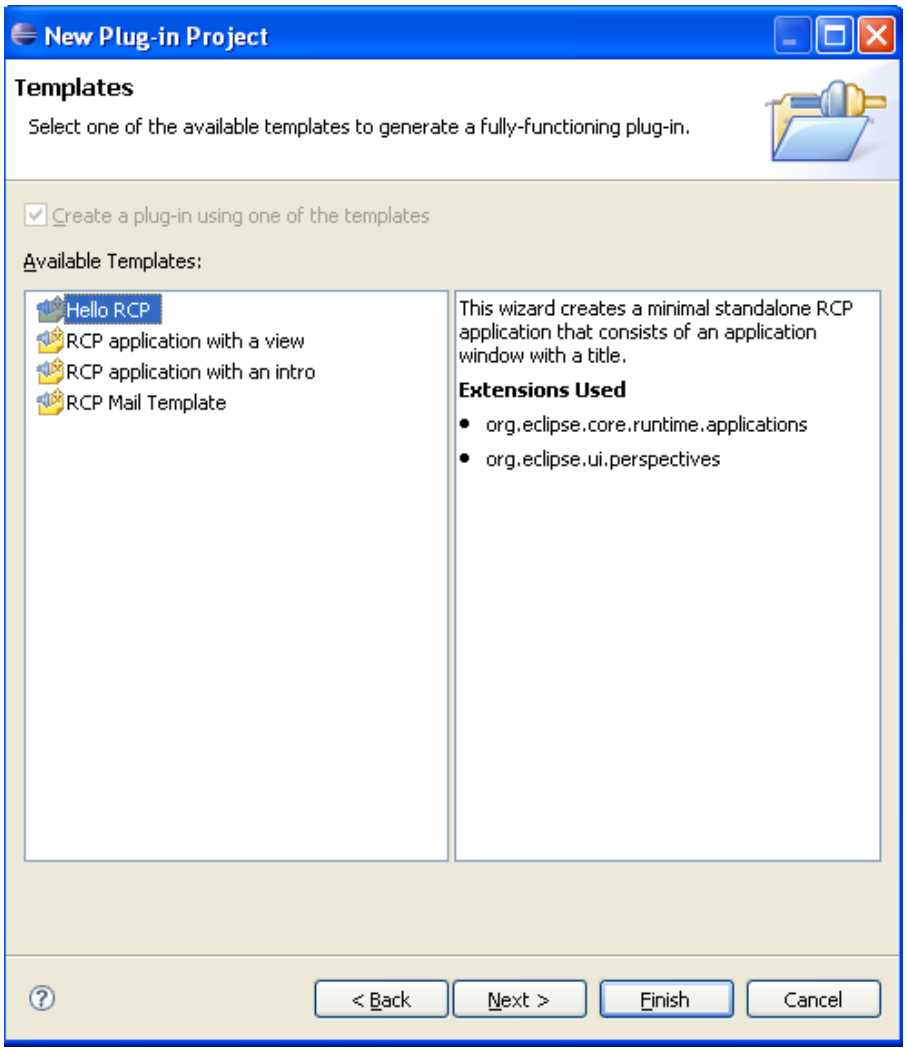


Слика 7.1 Визард за креирање новог plug-in-а

На слици 7.2 приказана је следећа страница у визарду за креирање plug-in-а у којој се може изабрати темплејт за генерисање RCP апликације. Постоје четири понуђена темплејта од којих сваки нуди одређени почетни скуп plug-in-ова Eclipse-а од којих ће се направити RCP апликација. Најједноставнији скуп почетних plug-in-ова добија се изботом темплејта *Hello RCP* који је селектован на слици 7.2. На десној страни екранске форме на слици 7.2 налази се опис темплејта који је селектован на левој страни екранске форме, као и списак plug-in-ова који ће улазити у састав RCP апликације ако зе изабере селектовани темплејт.

Темплејт *Hello RCP* који је селектован на страници визарда која је приказана на слици 7.2 креира минималану RCP апликацију која се састоји од прозора апликације који има свој наслов. Избором овог темплејта креираће се основна RCP апликација која ће садржати plug-in-ове:

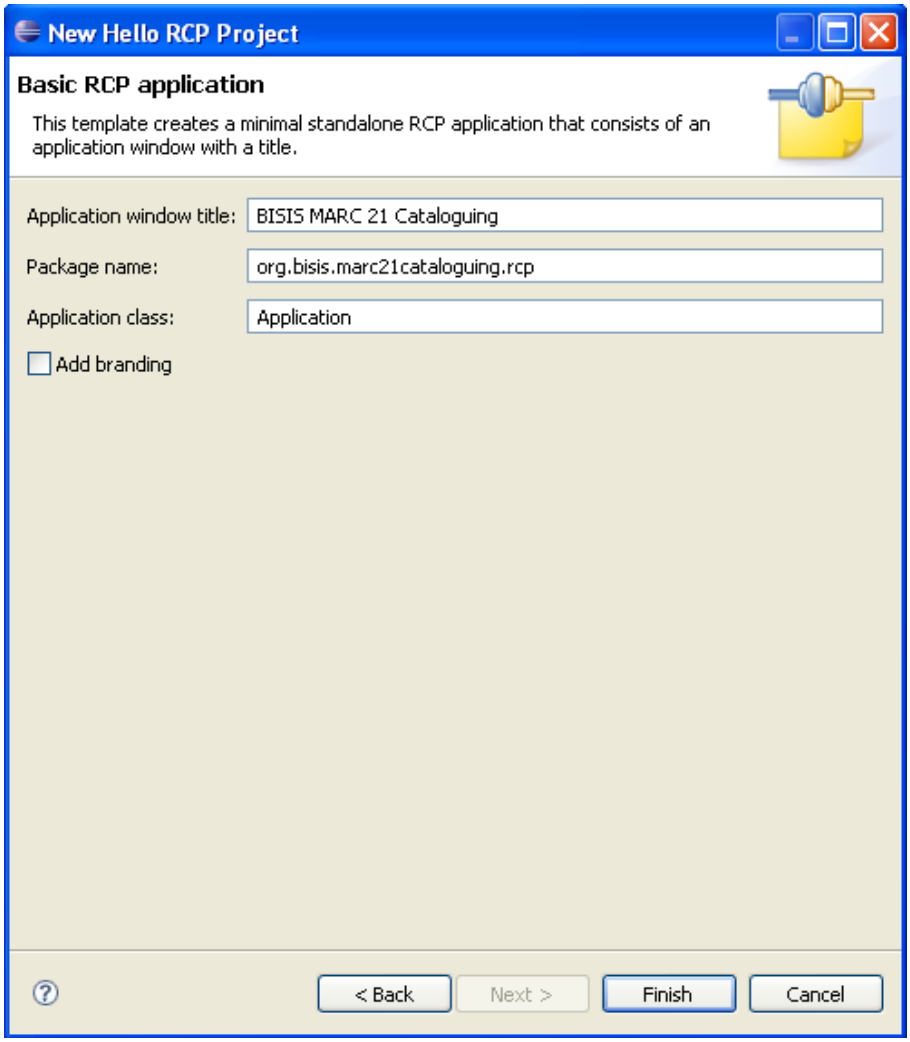
- *org.eclispe.core.runtime.application* и
- *org.eclipse.ui.perspectives*.



Слика 7.2 Избор темплејта за креирање RCP апликације

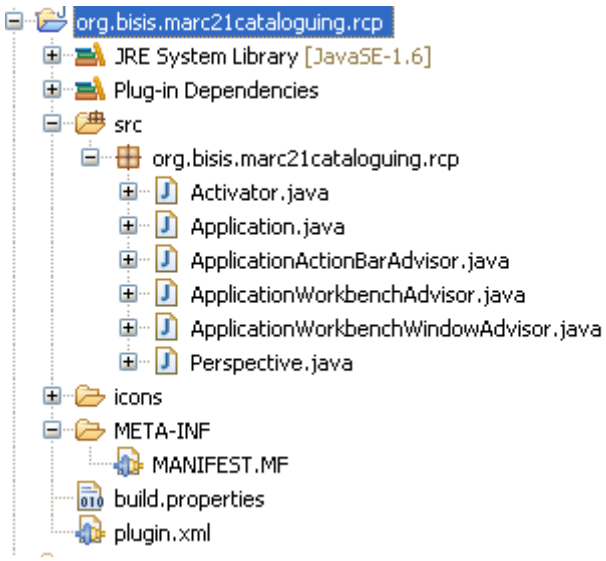
На слици 7.3 приказан је последњи корак у визарду за креирање RCP апликације. У прозору који је приказан на овој слици уноси се назив апликације који ће се појавити у заглављу главног прозора апликације. Поред

тога, на овом прозору наведен је и назив пакета и назив класе којима ће се реализовати RCP апликација.



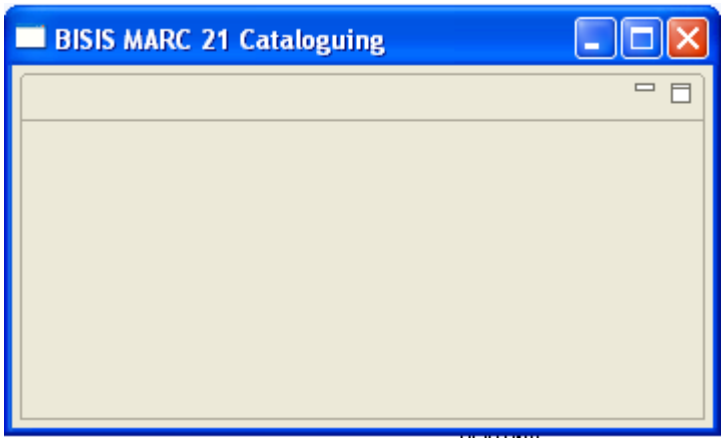
Слика 7.3 Креирање RCP апликације

Резултат спецификације у визарду је генериање пројекта *org.bisis.marc21cataloguing.rcp* у оквиру кога се развија plug-in који ће представаљати главни програм RCP апликације система за каталогизацију. Овај програм је иницијално генерисан са одређеним скупом класа и фајлова. На слици 7.4 приказана је структура пројекта *org.bisis.marc21cataloguing.rcp*. Његов фолдер *src* садржи Јава код који је генерисана на основу визарда. Поред Јава класа визардом су генерисани и фајлови *MANIFEST.MF* и *plugin.xml* за спецификацију карактеристика plug-in-а који је резултат посматраног пројекта.



Слика 7.4 Eclipse пројекат за креирање RCP апликације

Из Plug-in Manifest едитора може се покренути RCP апликација која је до сада креирана. На слици 7.5 приказан је изглед ове основне RCP апликације. RCP апликација која је до сада креирана састоји се од једног прозора који има наслов који је наведен у визарду.



Слика 7.5 Основна RCP апликација

7.2 КОСТУР RCP АПЛИКАЦИЈЕ

Класе које су генерисане визардом за креирање RCP plug-in-а и приказане на слици 7.4 представљају костур за креирање сваке RCP апликације. То су селеђеће класе:

- *Application*,
- *ApplicationActionBarAdvisor*,
- *ApplicationWorkbenchAdvisor*,
- *ApplicationWorkbenchWindowAdvisor*,
- *Perspective*.

Поред наведених класа генерисана је и класа *Activator* која се односи на сам plug-in и у њој се дефинишу акције које ће се извршавати при покретању и при завршетку рада plug-in-а. Остале генерисани класе чини костур RCP апликације и у наставку ће бити описана њихова улога.

Класа *Application* контролише све аспекте извршавања RCP апликације. Она имплементира метод *start()* којим се покреће RCP апликација и метод *stop()* који се позива за завршетак рада апликације.

Класа *ApplicationWorkbenchAdvisor* дефинише начин понашања и изгледа RCP апликације. У њој се специфицирају две ствари: основна перспектива која ће се користити за организацију компоненти корисничког интерфејса у главном прозору апликације и инстанца класе *ApplicationWorkbenchWindowAdvisor* којом је имплементиран главни прозор апликације.

Класом *Perspective* имплементирана је иницијална Eclipse перспектива која ће се користити у имплементацији класе *ApplicationWorkbenchAdvisor*. У класи *Perspective* специфицира се организација компоненти корисничког интерфејса (погледа, едитора, *toolbar*-а и слично) на екранској форми RCP апликације.

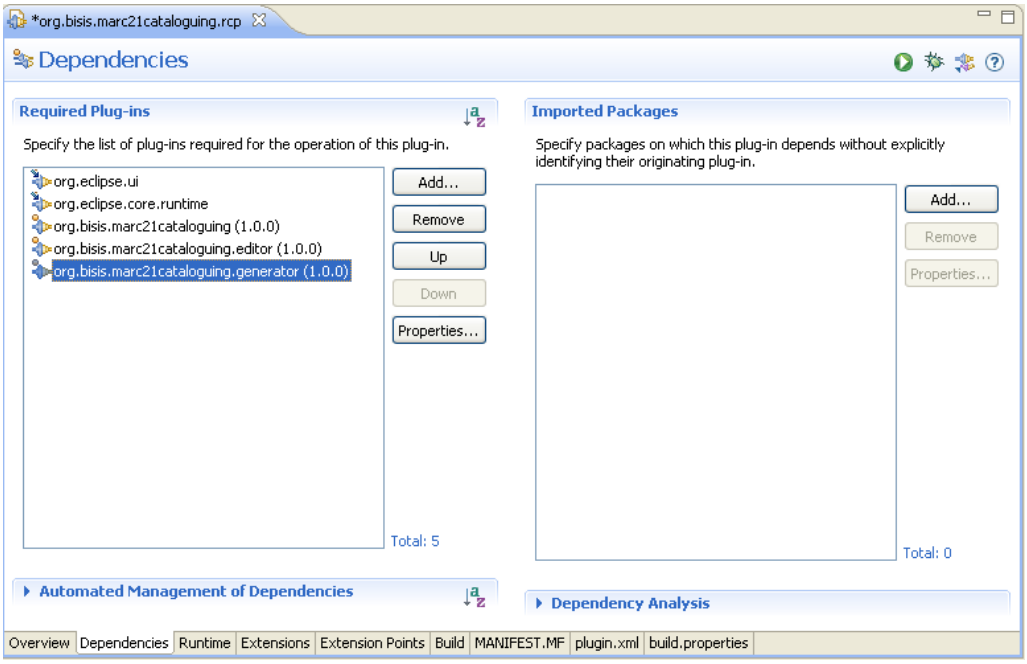
Класа *ApplicationWorkbenchWindowAdvisor* је имплементација главног прозора апликације. У њој се наводе димензије прозора, наслов који ће се јавити у заглављу прозора, статусна линија прозора и слично. Поред тога, ова класа садржи инстанцу класе *ApplicationActionBarAdvisor* у којој се имплементирају акције које ће бити доступне у главном прозору апликације.

7.3 ПРОШИРЕЊЕ RCP АПЛИКАЦИЈЕ PLUG-IN-ОВИМА ЗА КАТАЛОГИЗАЦИЈУ

Креирана RCP апликација још увек нема никакву функционалност. Да би се овој апликацији додале функционалности система за каталогизацију потребно је проширити основни скуп plug-in-ова који чине ову апликацију plug-in-овима којима је реализован систем за каталогизацију.

На слици 7.6 приказан је Plug-in Manifest едитор за RCP plug-in са отвореном картицом *Dependencies* у којој се додају plug-in-ови који ће се користити у раду RCP апликације. Поред основна два plug-in-а овде су укључена и три plug-in-а којима је имплементиран систем за каталогизацију. На овај начин обезбеђено је да у RCP plug-in-у буду доступне све компоненте и акције које су дефинисане у систему за каталогизацију.

У прозору који је приказан на слици 7.6 може се додати било који plug-in који је доступан у оквиру Eclipse платформе чиме се RCP апликацијама проширује функционалностима које су доступне у оквиру Eclipse-а.



Слика 7.6 Додавање plug-in-ова у картици *Dependences*

7.4 КРЕИРАЊЕ ОСНОВНОГ ПРОЗОРА RCP АПЛИКАЦИЈЕ СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

Да би се главни прозор RCP апликације која је приказана на слици 7.5 допунио елементима система за каталогизацију потребно је креирати перспективу која ће укључивати све елементе система за каталогизацију. Та перспектива се креира у класи *Perspective* која је део костура RCP апликације.

Перспектива којом се специфицира распоред компоненти у RCP апликацији система за каталогизацију имплементирана је тако да садржи три компоненте из система за каталогизацију, две компоненте из Eclipse окружења и скуп

акција које се односе на рад за записима креираним у систему за каталогизацију.

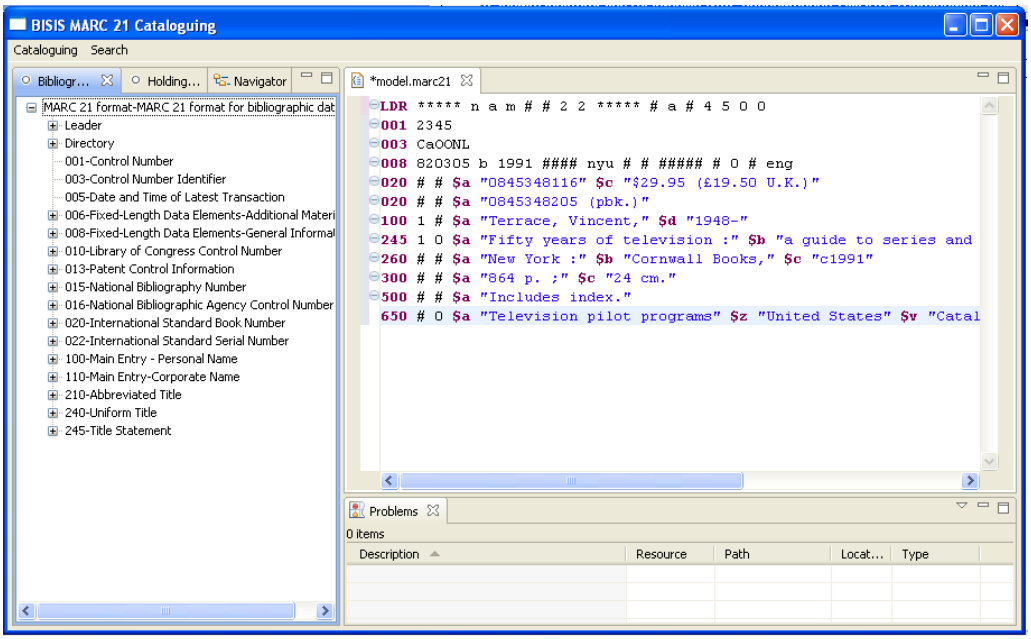
Компоненте система за каталогизацију које су обухваћене перспективом су поглед за приказ стабла MARC 21 библиографског формата, поглед за приказ стабла MARC 21 формата за локацијске податке и Xtext едитор за унос библиографског записа.

Од Eclipse-ових компоненти које ће бити укључене у RCP апликацију система за каталогизацију ту су поглед *Navigator* за управљање ресурсима који ће се отварати у едитору и поглед *Problems* у ком ће се пријављивати грешке настале приликом едитирања у едитору за каталогизацију.

Поред ових компоненти у RCP апликацији ће бити доступне и акције над записима које су дефинисане у оквиру скупа акција *Actions on record* и описане у одељку 6.2.2.

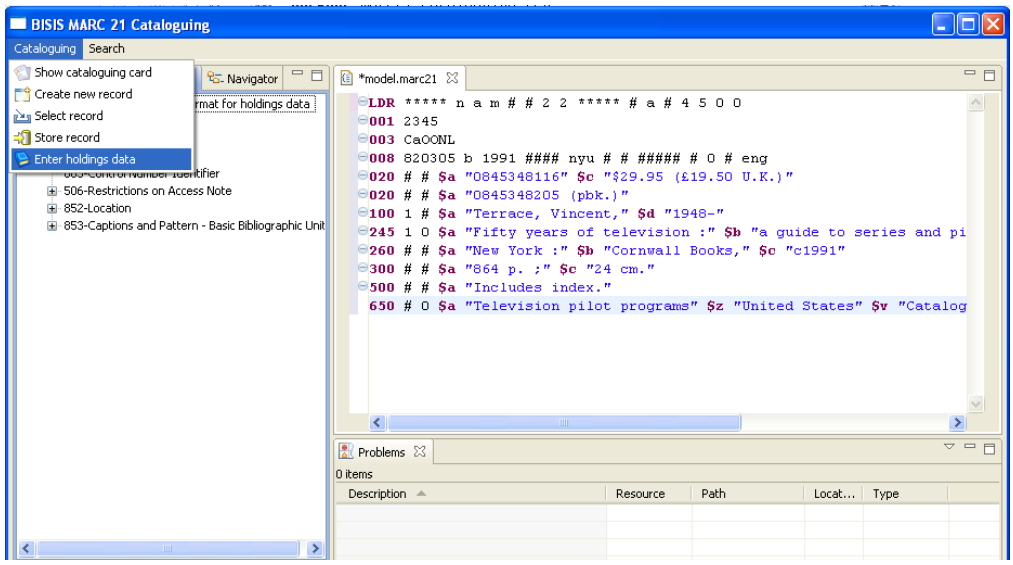
Организација наведених компоненти у оквиру перспективе дефинише се тако да се Xtext едитор налазе у централном делу прозора. Поглед за приказ MARC 21 библиографског формата, поглед за приказ MARC 21 формата за локацијске податке и поглед *Navigator* биће приказани у заједничкој компоненти у различитим картицама. Поглед *Problems* налазиће се у доњем делу екранске форме.

На слици 7.7 приказан је изглед RPC апликације која је допуњена компонентама и акцијама из plug-in-ова система за каталогизацију. Апликација има један основни прозор на ком се налазе едитор за каталогизацију који заузима централно место на прозору. На левој страни прозора налази се компонента са три погледа који се налазе у различитим картицама. На прозору са слике 7.7 отворена је картица *Bibliographic format view*. У доњем делу екранске форме налази се поглед *Problems* у оквиру ког се приказују грешке настале приликом уноса података у запис. Прозор има и мени линију која садржи мени *Cataloguing* у ком су доступне акције за рад са записима.



Слика 7.7 RCP апликација система за каталогизацију

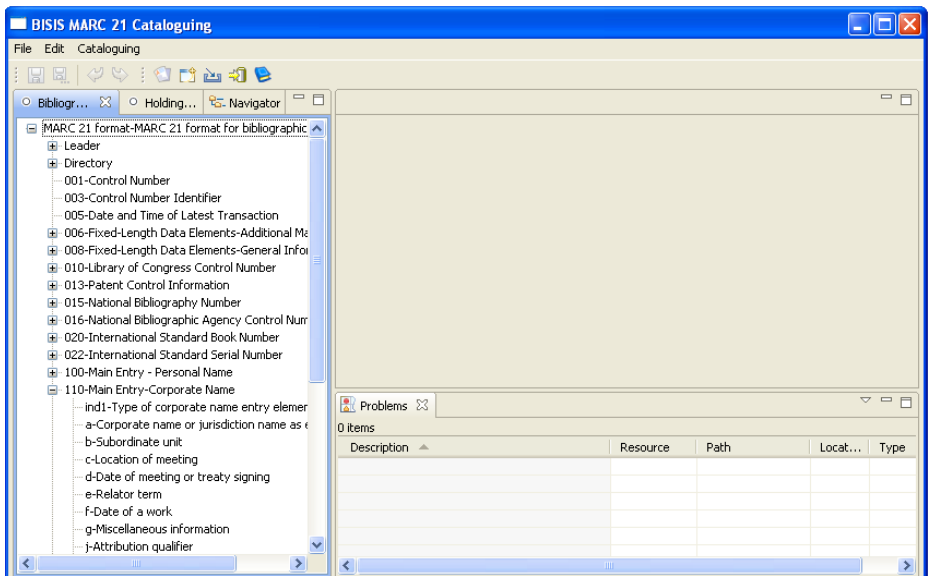
На слици 7.8 приказан је главни прозор RCP апликације за каталогизацију са отвореним менијем *Cataloguing*. У овом менију доступне су све акције које су специфициране у скупу акција *Actions on record* у plug-in-у *org.bisis.marc21cataloguing.editor*. Ту спадају акција за приказ каталошког листића (*Show cataloguing card*), акција за креирање новог записа на основу типа обраде (*Create new record*) , акција за импорт записа његовом селекцијом на основу ID броја (*Select record*), акција за складиштење записа (*Store record*) и акција за унос локацијских података (*Enter holdings data*).



Слика 7.8 Мени за каталогизацију

Да би се генерисана RCP апликација могла користити за каталогизацију независно од Eclipse-а у основни прозор апликације која је приказана на слици 7.7 додате су још неке акције које су део стандардне Eclipse платформе. То су акција за враћење последњег потеза (*undo*), акција за поништавање враћања потеза (*redo*), акција за снимање фајла (*save*) и акција за снимање фајла под новим именом (*save as*). Поред тога у RCP апликацију укључен је још један plug-in из стандардне Eclipse платформа, то је plug-in *org.eclipse.ui.browser*. Додавањем овог plug-in-а у RCP апликацију је додата компонента браузера у ком ће се отварати html документи каталошких листића.

Све наведене акције као и акције које се односе на каталогизацију доступне су и преко *toolbar*-а на главном прозору апликације. На слици 7.9 приказан је главни прозор RCP апликације која је допуњена додатним акцијама. Поред менија *Cataloguing* на мени линији прозора који је приказан на слици 7.9 налазе се и мени *File* у ком су доступне акције *save* и *save as*, као и мени *Edit* у ком су доступне акције *undo* и *redo*. Испод линије менија налази се *toolbar* у ком су доступне све наведене акције.



Слика 7.9 RCP апликација са комплетним скупом акција

7.5 ПРОШИРЕЊА PLUG-IN-ОВА СИСТЕМА ЗА КАТАЛОГИЗАЦИЈУ

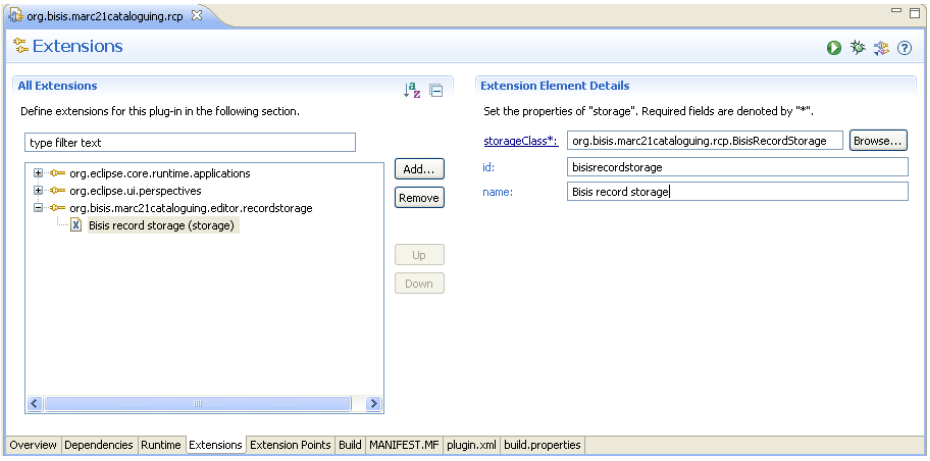
У оквиру RCP plug-in-а могу се специфицирати проширења за тачке проширења дефинисане у оквиру plug-in-ова који су укључени у реализацију посматраног plug-in-а и наведени у одељку *Dependences* на слици 7.6.

Као што је наведено приликом описа plug-in-а *org.bisis.marc21cataloguing.editor* у њему су дефинисане две тачке проширења, то су тачке проширења *recordstorage* и *processTypeCatalog*. Тачка проширења *recordstorage* омогућава проширење система имплементацијом складиштења и учитавања записа у базу података и описана је у одељку 6.2.3.1. Тачка проширења *processTypeCatalog* омогућава проширење система креирањем каталога типова обраде који ће се користити у каталогизацији и описана је у одељку 6.2.4.1.

Коришћењем наведених тачака проширења могуће је у оквиру RCP апликације имплементирати сопствену комуникацију са базом података и креирање каталога типова обраде. У наставку је показано дефинисање проширења за тачку проширења *recordstorage* у оквиру RCP plug-in-а. На сличан начин може се специфицирати ограничење и за тачку проширења *processTypeCatalog*.

На слици 7.10 приказана је спецификација проширења *Bisis record storage* за тачку проширења *recordstorage*. Овим проширењем дефинисано је складиштење и учитавање записа који ће се користити у RCP апликацији

система за каталогизацију. За тачку проширења *recordstorage* дефинисана су три параметра, *storageClass* који садржи назив класе у којој је имплементирана комуникација са базом података и која према спецификацији тачке проширења треба да наследи апстрактну класу *RecordStorage* из пакета *Storage* (одељак 6.1.4) и да имплементира њене две операције, *storeRecord(rec:Record)* којом се складишти објектни модел записа и *getRecord(recId:int)* којом се учитава запис са ID бројем.



Слика 7.10 Пришерење Bisis record storage

Проширење чија је спецификација приказана на слици 7.9 имплементирано је у класи *org.bisis.marc21cataloguing.rcp.BisisRecordStorage*. Назив ове класе наведен је као вредност параметара *storageClass* на левој страни екранске форме. Класа *org.bisis.marc21cataloguing.rcp.BisisRecordStorage* наслеђује апстрактну класу *RecordStorage* и имплементира њене две операције тако да се запис из објектног модела складишти у базу података која се користи у оквиру система БИСИС, односно запис се учитава из базе система БИСИС.

На овај начин имплементирана је функционалност која омогућава да се позивом акције *Store record* у менију за каталогизацију запис који је креиран у едитору складишти у базу система БИСИС и да се приликом селекције записа на основу ID броја запис учитава из базе система БИСИС.

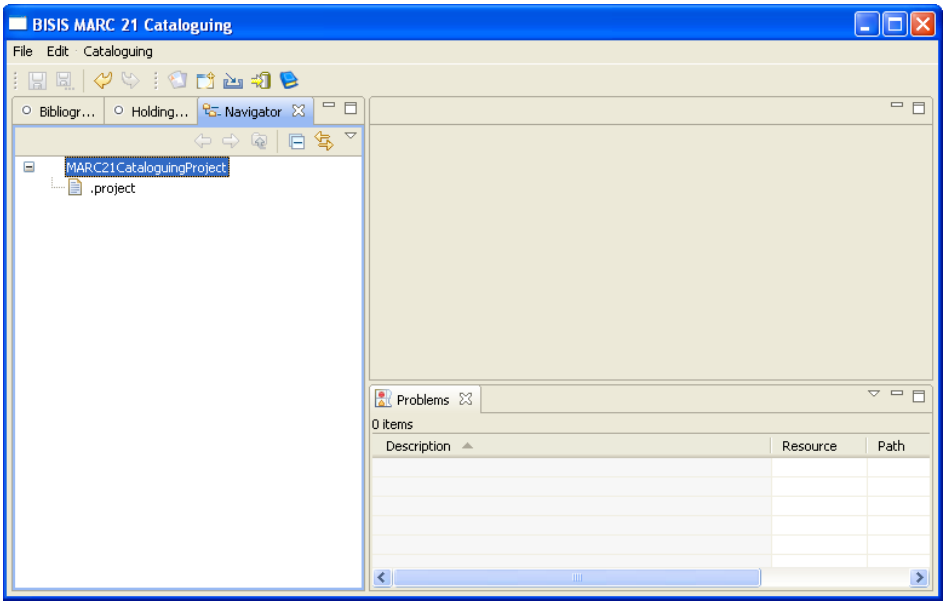
На сличан начин може се дефинисати и проширење за креирање каталога типова обраде при чему би се типови обраде учитавали из базе система БИСИС.

На описан начин имплементирана је RCP апликација која се може користити у оквиру софтверског система БИСИС. Слично томе може се направити RCP апликација која би се користила и у неком другом библиотечком софтеврском

систему. Све што треба урадити је имплементирати сопствену комуникацију за базом и креирање каталога типова обраде.

7.6 ОПИС КОРИШЋЕЊА RCP АПЛИКАЦИЈЕ ЗА КАТАЛОГИЗАЦИЈУ

Приликом првог покретања RCP апликације за каталогизацију потребно је креирати нови пројекат за каталогизацију у ком ће се смештати сви фајлови са MARC 21 записима који се креирају у едитору, као и html документи са каталошким листићима који се креирају за MARC 21 записе. Нови пројекат се креира помоћу визарда који се добија десним кликом у погледу *Navigator*. На слици 7.11 приказан је главни прозор апликације са отвореним погледом *Navigator* у ком је креиран пројекат за каталогизацију *MARC21CataloguingProject*. Сви фајлови у којима ће се креирати MARC 21 записи снимаће се у овај пројекат.



Слика 7.11 Пројекат за каталогизацију

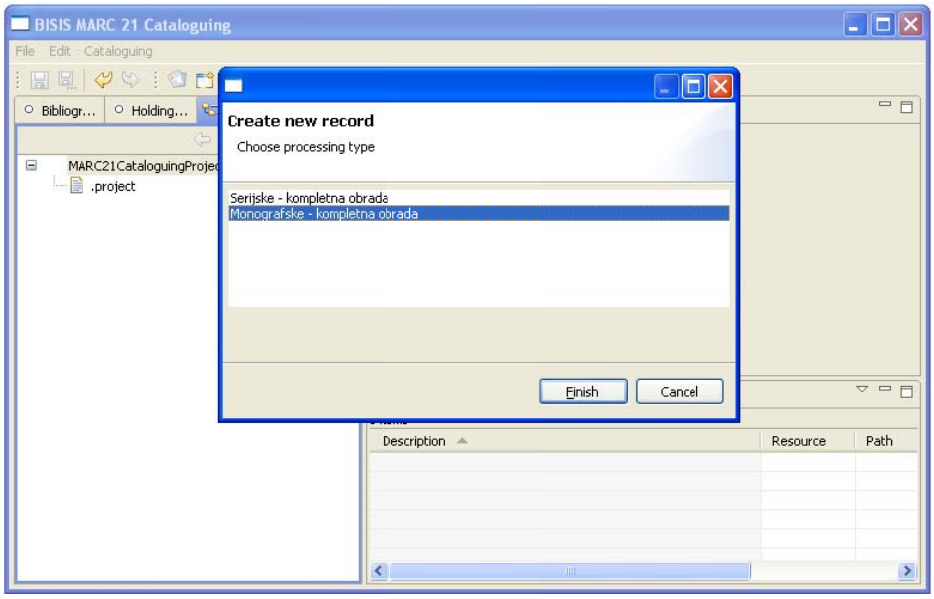
Један пројекат за каталогизацију представља групу записа у којој се наводе библиографски записи, записи за локацијске податке и генерисани каталошки листићи. У једној RCP апликацији може се креирати више пројеката. Организација пројеката за каталогизацију и записа у оквиру њих препуштена је кориснику апликације и осликава организацију ресурса у фајл систему клијента.

Централни део прозора који је приказан на слици 7.11 резервисан је за отварање ресурса пројеката. Фајлови са екстензијом *marc21* отвараће се у едитору за каталогизацију. Фајлови са екстензијом *html* отвараће се у браузеру

који је део генерисане RCP апликације. У том централном делу прозора може бити отворено више фајлова при чему је увек један активан и све акције се извршавају над тим активним фајлом.

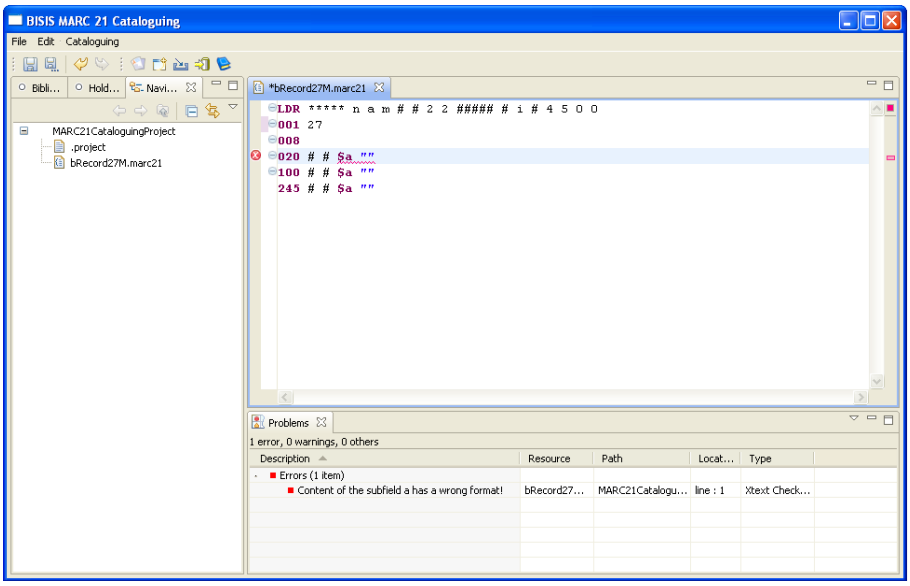
7.6.1 Креирање новог записа

Креирање новог записа у апликацији врши се позивом акције *Create new record* која је обележена иконицом . Позивом ове акције покреће се визард за избор типа обраде на основу кога ће се креирати иницијални запис у едитору. На слици 7.12 приказан је изглед визарда за избор типа обраде, након избора типа обраде *Monografske-kompletna obrada* креира се нови фајл у пројекту *MARC21CataloguingProject* који садржи иницијални MARC 21 запис који садржи поља, потпоља и подразумеване вредности дефинисане у изабраном типу обраде. Тај новокреирани фајл се отвара за едитирање.



Слика 7.12 Креирање новог записа

На слици 7.13 приказан је отворен фајл *bRecord27M.marc21* у ком се налази иницијални монографски запис креиран на основу изабраног типа обраде. Назив овог фајла је креиран од следећих података: *b* – библиографски запис; *Record27* – запис са редним бројем 27 што представља следећи слободан ID записа у локалној бази записа; *M* – монографски запис.

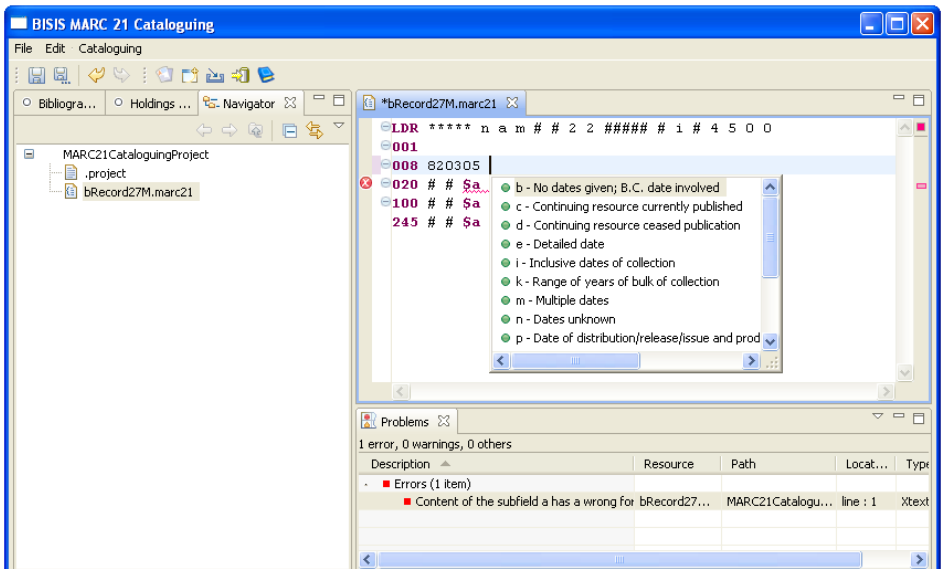


Слика 7.13 Креирање новог записа

7.6.2 Обрада записа у едитору

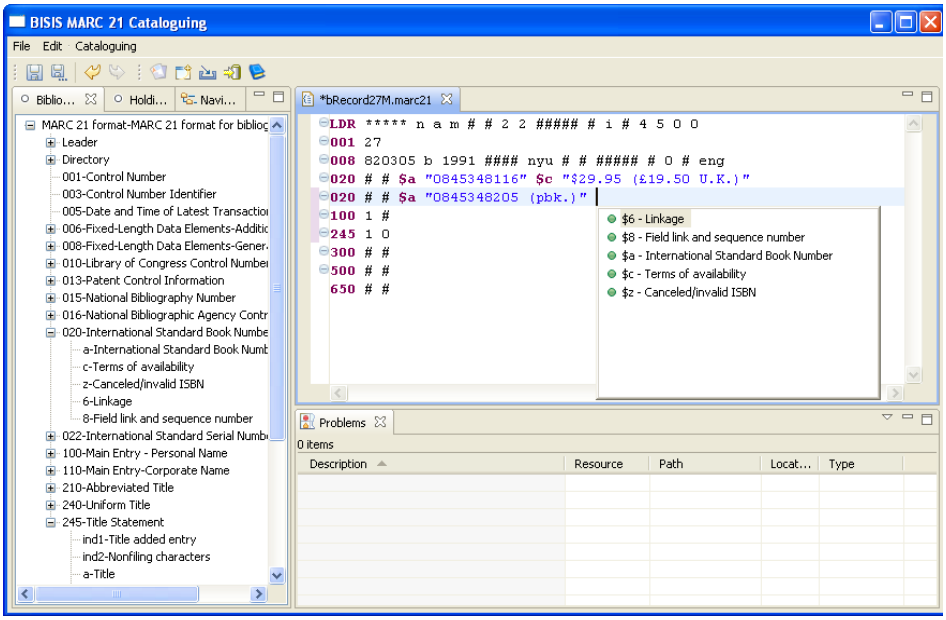
Запис који је приказан на слици 7.13 едитира се слободним уносом текста у едитор. У току уноса података пријављује се евентуални настанак грешке. На слици 7.13 приказан је случај пријављене грешке за потпоље *a* поља 020. Потпоље на које се односи грешка је подвучено црвеним а у доњем делу екрана у одељку *Problems* приказан је текст који описује узрок настале грешке.

Приликом уноса података у запис може се користити помоћ за допуну текста притиском на тастер Ctrl+Space после чега се на основу позицији на којој се налази курсор генерише падајућа листа са понудом вредности за унос. Ова помоћ може се користити за унос назива контролног поља, поља, потпоља, као и за унос шифара за шифриране карактерске позиције, потпоља и за индикаторе. На слици 7.14 приказан је унос шифриране вредности за другу карактерску позицију контролног поља *008*. На сличан начин уносе се и остале наведене шифриране вредности.



Слика 7.14 Унос шифриране вредности

Кориснику је на располагању и поглед *Bibliogarphic format view* у ком је приказано стабло библиографског формата које представља упутство за каталогизацију. Овај поглед се отвара притиском на картицу *Bibl...* на левој страни екранске форме главног прозора апликације (слика 7.14). На слици 7.15 приказан је изглед главног прозора RCP апликације у току едитирања. На прозору са ове слике отворен је поглед *Bibliographic format view*, а у едитору је у току унос новог потпоља поља 020 и отворена је падајућа листа са понудом потпоља поља 020 која могу да се унесу.



Слика 7.15 Обрада записа у RCP апликацији

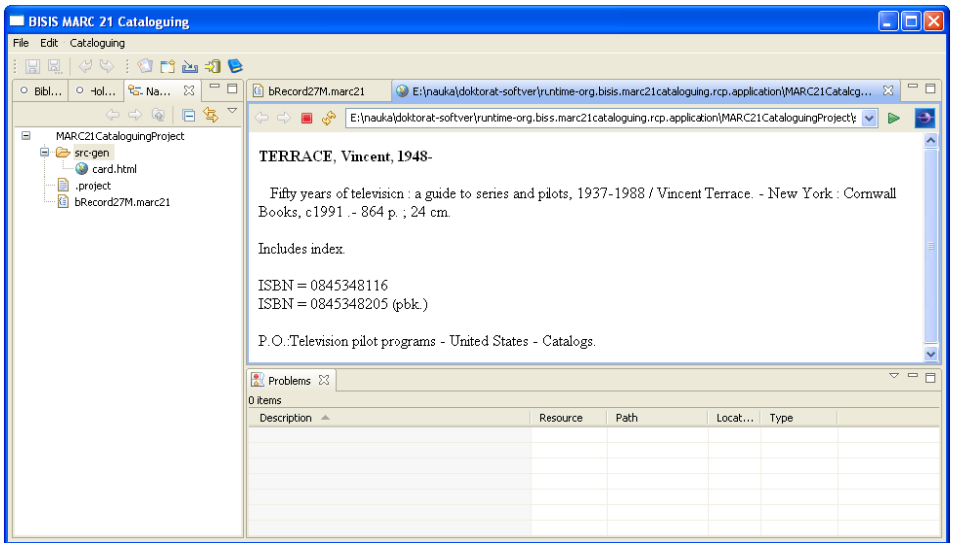
У едитору за унос записа кориснику су на располагању стандардне акције за копирање текста: *cut, copy*, и *paste* преко пречица са тастатуре. Коришћењем ових акција у запис се може пренети текст из било ког електронског извора (интернет странице или текстуалног фајла). Због начина уноса записа у едитор коришћењем наведених акција у едитор се могу копирати делови записа заједно за пољима и потпољима. Поред тога, наведеним акцијама може се пребацивати текст и у оквиру једног записа или између више записа који су отворени у едитору за каталогизацију.

Поред акција за копирање доступне су и стандардне акције за поништавање потеза. У току уноса података корисник може да врати потез (унос текста, брисање, и слично) позивом акције *undo* која је означена иконицом  или да поништи резултат акције *undo* позивом акције *redo* која је означена иконицом  .

Брисање елемената записа врши се на исти начин као и у стандардним текстуалним едиторима, селекцијом текста и притиском на тастер *Delete*.

У основном прозору апликације за каталогизацију кориснику су доступне и акције над фајлом у ком се едитира запис. Текуће стање записа се може снимити на диск рачунара у фајлу *bRecord27M.marc21* позивом акције *save* која је означена иконицом  или се исти запис може снимити у други фајл позивом акције *save as* која је означена иконицом .

У току обраде записа корисник може да прегледа креирани запис у облику каталошког листића. Позивом акције *Show cataloguing card* која је означена иконицом генерише се каталошки листић. Листић се генерише у облику html фајла *card.html* који се по генерисању аутоматски отвара у апликацији. На слици 7.16 приказан је отворен каталошки листић који је генерисан на основу целог унетог записа у едитору. На основу садржаја фајла *bRecord27M.marc21* генерисан је фајл *card.html* који је приказан у браузеру на слици 7.15.



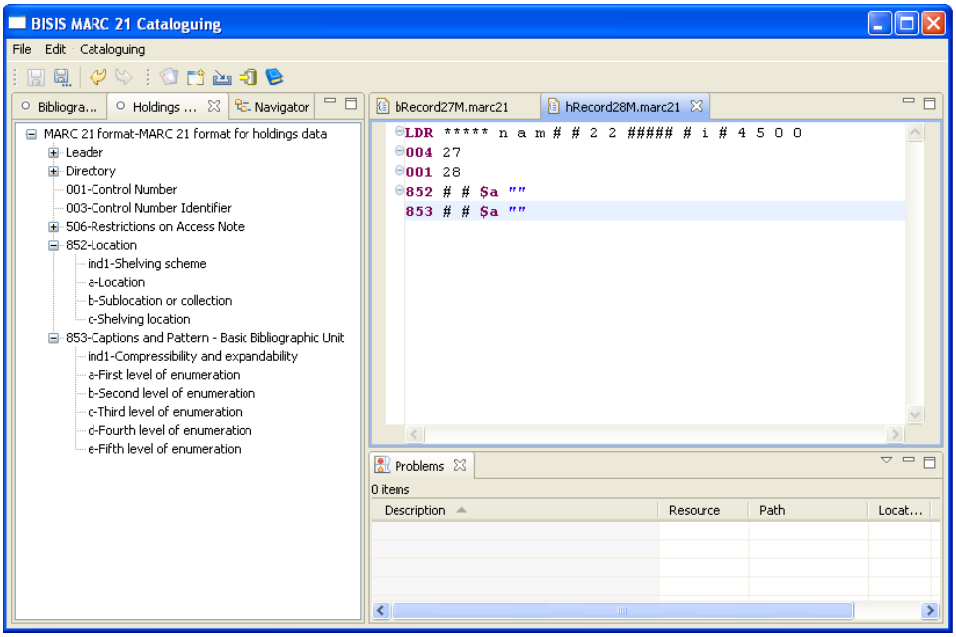
Слика 7.16 Приказ листића

Када је завршена обрада записа у едитору креирани запис се може складиштити у базу података позивом акције *Store record* означене иконицом . Тек после позива ове акције запис је складиштен у бази података система за каталогизацију.

7.6.3 Унос локацијских података

За креирани библиографски запис могу се везати локацијски подаци креирањем записа за локацијске податке. Позивом акције *Enter holdings data* која је означена иконицом за библиографски запис који је активан у едитору креира се нови запис за локацијске податке. На слици 7.17 приказан је резултат позива наведене акције за библиографски запис који је креиран у фајлу *bRecord27M.marc21*. Резултат акције је нови запис за локацијске податке који је креиран у фајлу *hRecord28M.marc21*.Овај запис у контролном пољу *004* садржи контролни број библиографског записа на који се односи. У пољу *001* овог записа налази се његов контролни број који је добијен као следећи

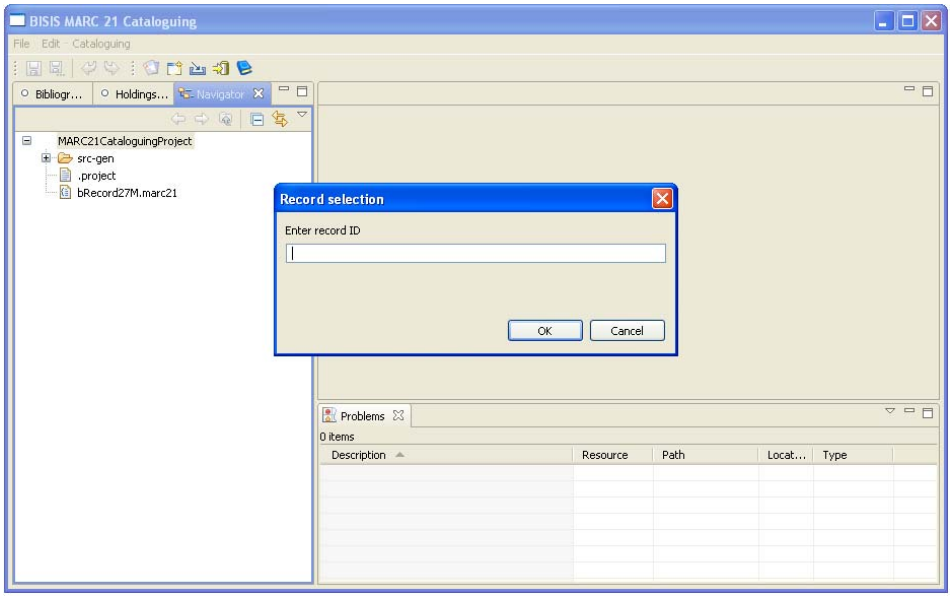
доступан ID број записа у локалној бази података. Новокреирани запис за локацијске податке отворен је за едитирање, а заједно са њим отворен је и поглед *Holdings format view* у ком је представљено стабло MARC 21 формата за локацијске податке које представља упутство за унос података у запис за локацијске податке. Запис за локацијске податке се у едитору обрађује на исти начин као и библиографски запис и све описане акције важе и над овим записом.



Слика 7.17 Креирање записа за локацијске податке

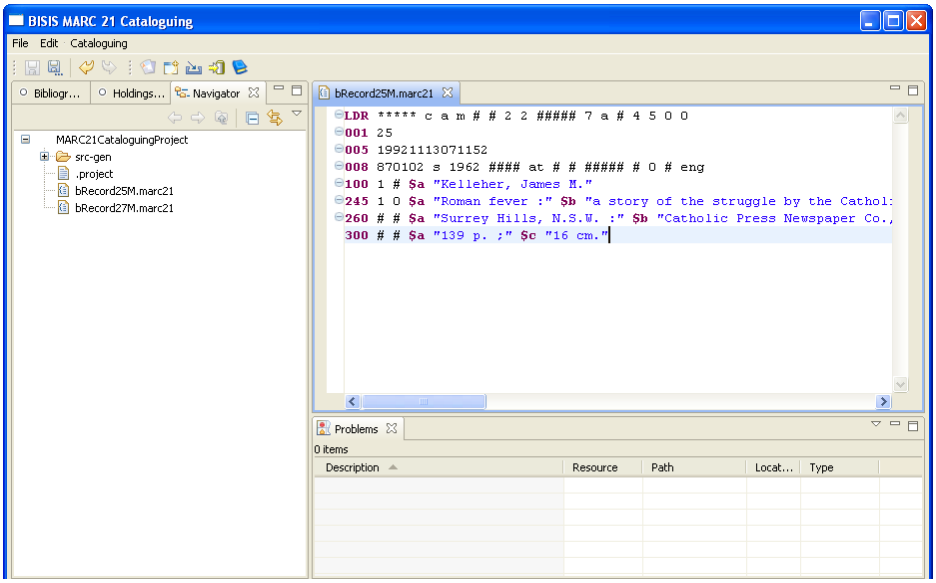
7.6.4 Учитавање постојећег записа

У генерисаној RCP апликацији може се вршити учитавање постојећег записа из базе података ради измене или уноса новог записа на основу постојећег. Учитавање записа из базе врши се позивом акције *Select record* која је означена иконицом на *toolbar*-у апликације. Позивом ове акције отвара се помоћни прозор приказан на слици 7.18 у ком се уноси ID записа који ће се учитати у едитору.



Слика 7.18 Учитавање записа

На слици 7.19 приказан је учитан библиографски запис чији је ID једнак 25. Овај запис учитан је у фајл *bRecord25M.marc21* који је отворен за едитирање. Овај запис се може изменити и поново складиштити у базу позивом акције *Store record* при чему ће се запис снимити под истим ID бројем и на овај начин извршена је модификација постојећег записа. Са друге стране, посматрани фајл се може снимити под другим именом позивом акције *save as* чиме се на основу постојећег записа креира нови запис који се смешта у фајл под другим именом. Овако креирани запис добија нови контролни број (садржај поља 001) и приликом снимања у базу ће се снимити под новим ID бројем. На овај начин може се креирати нови запис на основу постојећег.



Слика 7.19 Библиографски запис учитан у едитор

7.7 Интеграција RCP апликације система за каталогизацију

Описана RCP апликација за каталогизацију може се интегрисати у библиотечки информациони систем који се заснива на податак-централизованој (*data-centered*) софтверској архитектури. Основна одлика ове софтверске архитектуре је да постоји централизовано складиште података које користи више независних софтверских компоненти (Qian, et al, 2008). Те софтверске компоненте не комуницирају међусобно, него само преко складишта података. Постоје све врсте податак-централизоване софтверске архитектуре, то су *repository* и *blackboard* које се разликују у томе ко врши контролу тока извршавања. Код *repository* архитектуре складиште података је пасивно, а независне софтверске компоненте које га користе су активне и врше контролу извршавања. За *blackboard* архитектуру је карактеристично да је складиште података активно и оно врши контролу тока извршавања.

Интеграција система за каталогизацију у библиотечки информациони систем који се заснива на *repository* архитектури састоји се у томе да се у plug-in технологији имплементира проширење за комуникацију за складиштем података (складиштење и учитавање записа, учитавање типова обраде) и генерише RCP апликација која подржава ту комуникацију. На овај начин обезбеђен је механизам за креирање записа и њихово смештање у складиште. Ово складиште записа затим могу да користе остале компоненте библиотечког система, као што су циркулација, компонента за генерисање извештаја, компонента за преузимање записа преко интернета и слично.

Поглавље 8

Закључак

Резултати истраживања приказани у овој дисертацији су наставак истраживања добијених у магистарској тези (Димић, 2007). Резултат добијен у тој тези је едитор за обраду библиографске грађе по варијанти UNIMARC формата која се користи у библиотечком софтверском систему БИСИС. Реализација овог едитора заснована је на XML технологијама. Овај едитор је интегрисан у четврту верзију система БИСИС и описана је у другом поглављу ове дисертације.

Циљ истраживања које је описано у петом, шестом и седмом поглављу ове дисертације представља даљи развој и унапређење едитора за обраду библиографске грађе по MARC 21 формату. Основна идеја је да се реализује едитор који подржава унос структуираних података MARC 21 записа у форми која је карактеристична за савремене едиторе за програмске језике. Овакав приступ реализације едитора за MARC 21 записе није пронађен у доступној литератури.

За реализацију циља истраживања коришћен је обједињени процес за развој објектно-оријентисаних система у комбинацији са развојем система заснованим на моделу и развојем система заснованим на софтеврским компонентама. За моделирање система коришћен је CASE алат MagicDraw верзија 16.0 који подржава UML 2.0. За спецификацију модела и генерисање програмског кода коришћено је софтверско окружење Xtext и Eclipse Modeling Framework.

Оригинални резултати истраживања приказани су у четвртом, петом, шестом и седмом поглављу.

Добијени научни резултати приказани у четвртом поглављу су:

- Дат је предлог XML шеме библиографских формата (MARC 21 и UNIMARC) која представља модел за XML документе библиографских формата који се могу користити као улазна информација у системима за каталогизацију.

- На основу предложене шеме креиран је објектни модел библиографских формата који се може користити за имплементацију софтверске компоненте која обезбеђује информације о библиографским форматима на основу којих се врши каталогизација.
- На основу анализе постојећих XML шема MARC записа креиран је објектни модел MARC 21 записа који се у оквиру система за каталогизацију користи у разним аспектима обраде записа.

Наведени модели библиографских формата и записа су у основи спецификације и имплементације система за каталогизацију описаног у петом и шестом поглављу.

У поглављу пет описана је примена обједињеног процеса на развој система за каталогизацију и дата је спецификација тог система. Добијени научни резултати приказани у петом поглављу су:

- спецификација случајева коришћења система за каталогизацију;
- спецификација архитектуре система за каталогизацију дата у облику дијаграма пакета и дијаграма класа;
- моделирање статичких и донамичких карактеристика за следеће функционалности система за каталогизацију:
    - приказ података о библиографском формату за који је коришћен објектни модел MARC 21 формата који је дат у поглављу 4;
    - унос локацијских података;
    - експорт и импорт записа у облику објектног модела MARC 21 записа који је дат у поглављу 4;
    - приказ каталошких листића;
    - библиотечко окружење.

Моделирање наведених функционалности извршено је у CASE алату MagicDraw верзија 16.0.

У шестом поглављу описана је имплементација система за каталогизацију. Добијени научни резултати приказани у овом поглављу су:

- У софтверском алату Xtext специфицирана је граматика за опис модела MARC 21 записа. Дат је комплетан листинг ове спецификације, као и опис појединачних концепата граматике. Ова спецификација извршена је на основу објектног модела MARC 21 записа који је дат у поглављу 4.
- На основу наведене спецификације у Xtext окружењу генерисан је основни едитор за каталогизацију по MARC 21 формату и описане су његове карактеристике. Овај едитор генерисан је у облику скупа plug-in-ова за Eclipse. Овако генерисани едитор подржава унос

структуираних података MARC 21 записа у форми која је карактеристична за савремене едиторе за програмске језике.

- На основу специфициране Xtext граматике генерисан је EMF модел за MARC 21 библиографски запис који је централни део софтверске архитектуре едитора за каталогизацију.
- Над генерисаним EMF моделом библиографског записа у оквиру софтверског алата Xtext извршене су следеће спецификације:
    - на основу ограничења на структуру и садржај MARC 21 записа која су описана у четвртом поглављу извршена је спецификација тих ограничења коришћењем језика Check. Овом спецификацијом обезбеђена је контрола уноса података у реалном времену;
    - спецификација темплејта за трансформацију записа у форму каталошког листића. Ова спецификација извршена је коришћењем језика Xpand;
    - спецификација понуде предефинисаног скупа података за унос. Ова спецификација написана је у језику Xtend.
- У Eclipse plug-in технологији и програмском језику Јава извшена је имплементација додатних функционалности система за каталогизацију:
    - приказ података о библиографском формату;
    - унос локацијских података;
    - експорт и импорт записа;
    - приказ каталошких листића;
    - библиотечко окружење.

Све наведене додатне функционалности система за каталогизацију реализоване су као проширења стандардних компоненти корисничког интерфејса у оквиру Eclipse-a. Поред тога, коришћењем концепта тачака проширења plug-in-a реализована је могућност да се систем за каталогизацију може проширити функцијама које су специфичне за неко складиште података. Дефинисане су две тачке проширења, то су тачка проширења за складиштење и учитавање записа и учитавање библиотечког окружења.

Увођењем наведених тачака проширења реализована је могућност да се систем за каталогизацију може интегрисати у било који библиотечки информациони систем који се заснива на податак-централизованој (*data-centered*) архитектури (тип *repository*). Наиме, да би се систем за каталогизацију интегрисао у неки библиотечки информациони систем потребно је имплементирати проширење за складиштење и учитавање записа, као и проширење за учитавање типова обраде у зависности од складишта које

се користи у библиотечком информационом систему у који се врши интеграција.

На основу наведених научних резултата креирана је RCP апликација за каталогизацију која је описана у седмом поглављу. На основу креираних plug-in-ова којима је имплементиран систем за каталогизацију укључујући и plug-in-ове стандардне Eclipse платформе, коришћењем RCP технологије генерисана је апликација која се може користити независно од Eclispe-a. На овај начин добијена је софтверска компонента за каталогизацију која представља основни сегмент електронског пословања библиотека. Та компонента може се повезати са другим компонентама које реализују остале сегменте електронског пословања библиотека. Комуникација између тих софтверских компоненти остварује се преко заједничког складишта података (што је карактеристика repository софтверске архитектуре).

Практична примена добијених резултата биће искоришћена у даљем развоју система БИСИС. Садашња актуелна четврта верзија система БИСИС базирана је на варијанти UNIMARC формата а планира се прелазак на MARC 21 формат. Полазну основу за формирање верзије система БИСИС базиране на MARC 21 формату представљају резултати приказани у овој дисертацији.

RCP апликација за каталогизацију подржава два MARC 21 формата, то су MARC 21 формат за библиографске податке и MARC 21 формат за податке о фонду односно локацијске податке. Комплетна каталогизација по MARC 21 стандарду обухвата још три MARC 21 формата, то су MARC 21 формат за класификационе податке, MARC 21 формат за информације о заједници и MARC 21 формат за нормативне податке. Едитор који је генерисан у Xtext окружењу подржава свих пет MARC 21 формата јер граматика на основу које је он генерисан подржава све наведене формате. То значи да се приказани едитор може проширити и за све преостале MARC 21 формате тако што ће се за сваки од њих специфицирати ограничења и додати посебан поглед за приказ стабла формата.

Посебан допринос односи се на методолошки аспект пројектовања и имплементације библиотечког информационог система који се може користити и у другим информационим системима. Овај аспект састоји се у комбинацији више методолошких поступака. Основни методолошки поступак је обједињени процес развоја софтвера у оквиру ког је систем за каталогизацију развијен у три итерације. Прве две итерације реализоване су коришћењем развоја софтвера заснованог на моделима, а у трећој итерацији коришћен је развој заснован на компонентама и развој заснован на XML технологијама.

Предложени начин генерисања едитора на основу спецификације граматике у Xtext окружењу може се користити и за генерисање едитора у оквиру других информационих система. Xtext окружење нуди значајне могућности за спецификацију унапређења генерисаног едитора као што су спецификација ограничења, трансформација унетих података, креирање помоћи кориснику приликом уноса и слично. Сва наведена унапређења омогућавају креирање савремених корисничких интерфејса само на основу спецификације и без писања програмског кода.

У оквиру истраживања у овој дисертацији показано је да се коришћењем Eclipse plug-in технологије могу креирати апликације за крајњег корисника. Развој апликација у plug-in технологији омогућава коришћење велике библиотеке креираних компоненти корисничког интерфејса Eclipse платформе чиме је и у развоју додатних функционалности система избегнуто писање програмског кода. Поред тога, plug-in технологија омогућава развој лако проширивих апликација коришћењем концепта тачке проширења. На овај начин креирају се софтверске компоненте које се могу користити од стране великог броја различитих информационих система.

Литература

Референце

Belić, K. and Surla, D. (2008a), "Model of User Friendly System for Library Cataloguing"*, ComSIS*, Vol. 5, No. 1, pp. 61-85.

Belić, K. and Surla, D. (2008b), "User-friendly web application for bibliographic material processing", *The Electronic Library*, Vol.26, No 3, pp. 400-10

Boberić, D. and Surla, D. (2009), "XML Editor for Search and Retrieval of Bibliographic Records in the Z39.50 Standard", *The Electronic Library*, Vol.27, No 3, pp. 474-95

Боберић Д. (2007), "XML едитор за преузимање библиогарфских записа", магистарска теза, *Природно-математички факултет*, Нови Сад

Боберић, Д. и Сурла, Д. (2007), "Преузимање библиогарфских записа по Z39.50 стандарду", монографија, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Будимир, Г. и Сурла, Д. (2004), "Систем за контролу квалитета XML библиографских записа", монографија, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Budinsky, F., Steinberg, D., Merks, E., Ellersick, R and Grose, T. (2003), " Eclipse Modeling Framework: A Developer's Guide", *Addison Wesley*, United States

Carvalho, J. and Cordeiro, M. (2002), "XML and bibliographic data: the TVS (transport, validation and services) model", paper presented to the *68th IFLA Council and General Conference*, доступно на: www.ifla.org/IV/ifla68/papers/075-095e.pdf (прегледано 31. августа 2009)

Carvalho, J., Cordeiro, M. I., Lopes, A and Vieira, M. (2004), "Meta-information about MARC: an XML framework for validation, explanation and help systems", *Library Hi Tech*, Vol 22, No 2, pp. 131-7

Carvalho, J (2005), "An XML representation of the UNIMARC manual: a working prototype", paper presented to the World Library and Information Congress: 71th IFLA General Conference and Council, доступно на: http://www.ifla.org/IV/ifla71/papers/199e-Carvalho.pdf (прегледано 31. августа 2009)

Clayberg, E and Rubel, D. (2008), " Eclipse Plug-ins, Third Edition", *Addison Wesley*, United States

Dimić, B. and Surla, D. (2009), "XML Editor for UNIMARC and MARC21 cataloguing", *The Electronic Library*, Vol. 27., No 3, 509-28

Dimić, B., Milosavljević, B., and Surla, D. (2010) "XML schema for UNIMARC and MARC 21 formats", *The Electronic Library* (in press)

Dimić, B. and Milosavljević, B. (2009), "Customisation of cataloguing in library software system BISIS", *4th International Conference on Engeneering technologies ICET 2009*, Novi Sad

Димић, Б и Сурла, Д.(2007), "Обрада библиографске грађе у софтверском систему БИСИС", монографија, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Димић, Б. (2007), "XML едитор за обраду библиогарфске грађе", магистарска теза, *Природно-математички факултет*, Нови Сад

Efftinge, S., Friese, P., Haase, A., Hübner, D., Kadura, C., Kolb, B, Köhnlein, J., Moroff. D., Thoms, K., Völter, M., Schönbach. P., Eysholdt, M., Hübner. D. and Reinisch, S. (2008), „openArchitectureWare User Guide, Version 4.3.1", *openArchitectureWare*, доступно на: http://www.openarchitectureware.org/pub/documentation/4.3.1/openArchitectureWare-4.3.1-Reference.pdf (прегледано 18.07.2009.)

Gorton, I. (2006), "Essential software architecture", *Springer-Verlag Berlin Heidelberg*, Germany

Harris, R. and Warner, R. (2004), „The Definitive Guide to SWT and JFace", *Springer-Verlag*, New York, United States

Jacobson I., Booch G. and Rumbaugh, J. (1999) „The Unified Software development process", *Addison-Wesley*

Khurshid, Z. (2001) "The cataloger's workstation in the electronic library environment", *The Electronic Library*, Vol. 19, No. 2, pp. 78-83

Khurshid, Z. (2003), "Electronic tools for cataloging", *OCLC Systems & Services*, Vol. 19, No. 1, pp. 23-7

Kolb, B. and Voalter, M (2008), "Textual DSLs with oAW Xtext", openArchitectureWare, доступно на: http://www.openarchitectureware.org/pub/documentation/4.3/xtext-4.3.pdf (прегледано 19.07.2009.)

Lange, H. R. (1993), "Catalogers and workstations: A retrospective and future view", *Cataloging & Classification Quarterly* Vol. 16 No. 1, pp 39-52

Larman, C. (2002), “Applying UML and Patterns: an introduction to object-oriented analysis and design and the unified process, second eddition”, *Prentice Hall PTR*, USA

Leroy, Y. S. and Thomas, L. S. (2004), “Impact of Web Access on Cataloging“, *Cataloging & Classification Quarterly* Vol. 38, No. 2, pp. 7-16

Лазаревић, Б., Бендер, М., Цветановић, С., Дикановић, В., Ђуричић, Д., Конечни, А., Коруновић, Д., Миливојевић, Љ., Нешковић ,С., Пауновић, Ђ. и Сурла, Д. (1996), “Формирање и претраживање База података у систему научних и технолошких информација Србије”, *Министарство за науку и технолошки развој Републике Србије*, Београд

McAffer, J. and Lemieux, J. (2005),“Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications“, *Addison Wesley Professional*, United States

Мијић, В. (2003), “XML едитор за опис UNIMARC формата“*,* магистарска теза, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Milosavljević, B. and Dimić, B. (2007)*,*”XML schema of UNIMARC format variant and bibliographic record in BISIS software system“, *NSJOM*, Vol. 37, No.1, 2007, pp 115-28.

Milosavljevic, B. and Tesendic, D. (2010), “Software Architecture of Distributed Client/Server Library Circulation”, *The Electronic Library* (in press)

Milosavljević, B., Boberić, D. and Surla, D. (2010), “Retrieval of Bibliographic Records Using Apache Lucene”, *The Electronic Library* (in press)

Ojala, O. (2005), “T-76.115 Technical Specification TEXlipse project Group TeXlapse”, ID: TEXLIPSE-TECH-1Version: 1.8, доступно на: http://surfnet.dl.sourceforge.net/project/texlipse/texlipse%20documentation/version%201.0/texlipse-techspec-1.0.0.pdf (прегледано 8. септембра 2009.)

Powell, A. (2004), „Model with the Eclipse Modeling Framework, Part 1: Create UML models and generate code“, *IBM*, доступно на: http://www.ibm.com/developerworks/library/os-ecemf1 (прегледано 24.07.2009.)

Qian, K., Fu, X., Tao, L., Xu, C. and Diaz-Herrera, J. L. (2008) “Software Architecture and Design Illuminated”, *Jones and Bartlett Publishers*, Canada

Rađenović, J., Milosavljević, M. and Surla, D. (2009), “Modelling and implementation of catalogue cards using FreeMarker” *Program-electronic libarary and information systems*, Vol. 43, No.1, pp. 63-76

Rađenović, J., Milosavljević, B. and Surla, D. (2006), "Generating catalog cards by the BISIS library software", INFOTEKA, Vol 7, No 1-2, pp 61-74

Рађеновић J., Сурла, Д. и Милосављевић Б. (2006), "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

Рађеновић, J.(2006), "Моделирање и имплементација библиографских каталошких листића у софтверском пакету FreeMarker", магистарска теза, *Природно-математички факултет*, Нови Сад

Roper, O. J. and Pennell, C. (2001), "The Cataloger's Desktop Suite", *OCLC Systems & Services*, Vol. 17, No. 3, pp. 121-32

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

Spjuth, O., Helmus, T., Willighagen, E.L., Kuhn, S., Eklund, M., Wagener, J., Murray-Rust, P., Steinbeck, C. and Wikberg, J.E. (2007), "Bioclipse: an open source workbench for chemo- and bioinformatics", *BMC Bioinformatics* doi:10.1186/1471-2105-8-59

Škrbić, S. and Surla, D. (2008), "Bibliographic records editor in XML native environment", *Software-Practice and Experience*. Vol 38, Issue 5, pp 471-91

Шкрбић С. и Сурла, Д. (2004), "Управљање XML библиогарфском записима у XML технологији", XXXI Симпозијум о операционим истраживањима *SYM-OP-IS 2004*, Иришки Венац стр. 133-6

Шкрбић, С. и Сурла, Д. (2005), "Обрада библиографске грађе у XML native технологији", монографија, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Taft, Darryl K. (2006), "JPMorgan stays focused on Eclipse" , *eWeek*, Vol. 23 Issue 41, pD1-D4

Tešendić, D., Milosavljević, B. and Surla, D. (2009), "A Library Circulation System for City and Special Libraries", *The Electronic Library*, Vol. 27, No. 1, pp 162-8

Тешендић, Д. и Сурла, Д. (2007), "Коришћење библиотеħке грађе у софтеврском систему БИСИС", монографија, *Природно-математички факултет*, Департман за математику и информатику, Нови Сад

Тешендић, Д. (2007), "Систем за коришћење библиотечке грађе", магистарска теза, *Природно-математички факултет*, Нови Сад

Vidaković, J. and Racković, M. (2006), "Generating Content and Display of Library Catalogue Cards Using XML Technology", *Software: Practice and Experience*, Vol. 36 No. 5 pp. 513-524

Watson, G.R. and DeBardeleben, N.A. (2006), "Developing scientific applications using Eclipse", *IEEE Computing in Science & Engineering*, Vol. 8, No. 4, pp. 50-61

Zeremski, M. and Surla, D. (2003), "Validation of XML bibliographic records in Java environment"*, Novi Sad Journal of Mathematics*, Vol. 33, No. 2, pp.111-8

Зеремски, М. и Сурла, Д. (2001ц), "Моделирање библиографских записа у облику XML докумената"*, ИНФОТЕКА: Часопис за информатику и библиотекарство*, Београд, Вол. 2 Број 1-2, стр. 93-7.

Зеремски, М. и Сурла, Д. (2002),"Валидација библиографских записа помоћу XML Schema језика"*,* Зборник радова са Симпозијума о рачунарским наукама и информационим технологијама *YU INFO 2002* (на CD-ROM-у), Копаоник

Зеремски, М. и Сурла, Д.(2001а), "Моделирање UNIMARC формата у XML – Data jeziku"*,* Зборник радова са Симпозијума о рачунарским наукама и информационим технологијама *YU INFO 2001* (на CD-ROM-у), Копаоник

Зеремски, М. и Сурла, Д.(2001б),"Моделирање UNIMARC формата помоћу XML Schema језика", Зборник радова са XXVIII Југословенског симпозијума о операционим *истраживањима SYM-OP-IS 2001*, Београд, 2001. стр. 261-4.

Web stranice

[1] MARC Records, Systems and Tools, доступно на: http://www.loc.gov/marc/marcservice.html (прегледано 31. августа 2009.)

[2] MARC Record Services, доступно на: http://www.loc.gov/marc/marcrecsvrs.html (прегледано 31. августа 2009.)

[3] Marc Systems, доступно на: http://www.loc.gov/marc/marcsysvend.html (прегледано 31. августа 2009.)

[4] MARC Specialized Tools, доступно на: http://www.loc.gov/marc/marctools.html (прегледано 31. августа 2009.)

[5]UDC Consortium, доступно на: http://www.udcc.org/about.htm (прегледано 31.августа 2009.)

[6] ISSN, доступно на: http://www.issn.org/ (прегледано 31. августа 2009.)

[7] ISBN, доступно на: http://www.isbn.org/ (прегледано 31.августа 2009.)

[8] The Unicode Consortium, http://unicode.org/ (прегледано 30 августа 2009.)

[9] ISBD, доступно на: http://archive.ifla.org/VII/s13/pubs/isbd.htm (прегледано 31.августа 2009.)

[10] MARC Format Overview. Network Development and MARC Standards Office, Library of Congress, доступно на: http://www.loc.gov/marc/status.html (прегледано 28.августа 2009.)

[11] The Library of Congress, доступно на: http://www.loc.gov (прегледано 28. августа 2009.)

[12] MARC Standards. Network Development and MARC Standards Office, Library of Congress, доступно на: http://lcweb.loc.gov/marc/ (прегледано 28. августа 2009.)

[13] UNIMARC Manual: Bibliographic Format 1994-IFLA Universal Bibliographic Control and International MARC Core Programme (UBCIM), доступно на: http://www.ifla.org/VI/3/p1996-1/sec-uni.htm (прегледано 31. августа 2009)

[14] Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, National Information Standards Organization, Bethesda, Maryland, доступно на: http://www.loc.gov/z3950/agency/Z39-50-2003.pdf (прегледано 30. августа 2009.)

[15] SRU: Search/Retrieval via URL - SRU, CQL and ZeeRex (Standards, Library of Congress), доступно на: http://www.loc.gov/standards/sru/ (прегледано 14. септембра 2009.)

[16] Z3983 (2008), The NCIP Implementation Group Home Page, доступно на: http://ncip.envisionware.com/ (прегледано 11. септембра 2009.)

[17] BookMARC, Bibliographic Information Services, доступно на: http://www.bookmarc.pt/tvs/ (прегледано 31. августа 2009.)

[18] BookMarc, Prototype XML schema for UNIMARC, доступно на: http://www.bookmarc.pt/unimarc/ (прегледано 31. августа 2009.)

[19] Schema FORMAT.xsd, доступно на: http://www.bookmarc.pt/unimarc/schema/FORMAT.html (прегледано 31. августа 2009.)

[20] Schema MARC21DOC.xsd, доступно на: http://www.bookmarc.pt/documentation/marcdoc/xsd.html (прегледано 31. августа 2009.)

[21] MarcEdit, доступно на: http://oregonstate.edu/~reeset/marcedit/html/ (28. августа 2009.)

[22] GIS Information Systems, доступно на: http://www.gisinfosystems.com (прегледано 28. августа 2009.)

[23] CDS/ISIS database software, доступно на: http://www.unesco.org/isis/ (прегледано 28. августа 2009.)

[24] Concourse Software Product, доступно на: http://www.booksys.com/v2/products/concourse/ (прегледано 28. августа 2009.)

[25] Book Systems, Inc, доступно на: http://www.booksys.com (прегледано 28. августа 2009.)

[26]Eclipse.org home, доступно на: http://www.eclipse.org/ (прегледано 28. августа 2009.)

[27]Java Technology, <http://java.sun.com> (прегледано 28. августа 2009.)

[28] Eclipse Plugin Development Tutorial, доступно на: http://www.eclipsepluginsite.com/index.html (прегледано 31.јула 2009)

[29] SWT: The Standard Widget Toolkit, доступно на: http://www.eclipse.org/swt/(прегледано 06. августа 2009.)

[30] JFace – Eclipsepedia, доступно на: http://wiki.eclipse.org/index.php/JFace (прегледано 06. августа 2009.)

[31] Rich Client Platform – Eclipsepedia, доступно на: http://wiki.eclipse.org/index.php/Rich_Client_Platform> (прегледано 31.07.2009.)

[32] Eclipse Modeling – EMF Home, доступно на: http://www.eclipse.org/modeling/emf/ (прегледано 24. јула 2009.)

[33] openArchitectureWare.org - Official openArchitectureWare Homepage, доступно на: http://www.openarchitectureware.org/ (прегледано 25 јула.2009.)

[34] ANTLR Parser Generator, доступно на: http://www.antlr.org/ (прегледано 25. јула.2009.)

[35] TeXlipse homepage - LaTeX for Eclipse, доступно на: http://texlipse.sourceforge.net/ (погледано 8. септембра 2009)

[36] Apache Lucene, доступно на: http://lucene.apache.org/ (прегледано 31. августа 2009.)

[37] FreeMarker: Java Template Engine Library – Overview, доступно на: http://freemarker.org/ (прегледано 31. августа 2009.)

[38] MDA, доступно на: http://www.omg.org/mda/ (прегледано 23. септембра 2009.)

[39]Component-based software engineering - Wikipedia, the free encyclopedia, доступно на:

http://en.wikipedia.org/wiki/Component-based_software_engineering (прегледано 25. септембра 2009.)

[40] Object Constraint Language, доступно на: http://www.omg.org/technology/documents/formal/ocl.htm (прегледано 25. јула 2009.)

[41]PDE, доступно на: http://www.eclipse.org/pde/ (прегледано 30. јула 2009.)

[42] W3C XML Schema, доступно на: http://www.w3.org/XML/Schema (прегледано 31. августа 2009.)

[43] ANSI/NISO Z39.2 - Information Interchange Format, доступно на: http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key=fb7a107043228a342cb704973825aca7bc6ae58d (прегледано 31. августа 2009)

[44] ISO 2709:1996 - Information and documentation - Format for information exchange, доступно на: http://www.iso.org/iso/catalogue_detail.htm?csnumber=7675 (прегледано 31. августа 2009)

[45] AACR2, доступно на: http://www.aacr2.org/ (прегледано 31. августа 2009)

[46] MARC 21 Format for Bibliographic Data: Table of Contents (Network Development and MARC Standards Office, Library of Congress), доступно на: http://www.loc.gov/marc/bibliographic/ecbdhome.html (прегледано 31. августа 2009.)

[47] MARC 21 Format for Authority Data: Table of Contents (Network Development and MARC Standards Office, Library of Congress), доступно на: http://www.loc.gov/marc/authority/ecadhome.html (прегледано 31. августа 2009.)

[48] MARC 21 Format for Holdings Data: Table of Contents (Network Development and MARC Standards Office, Library of Congress, доступно на: http://www.loc.gov/marc/holdings/echdhome.html (прегледано 31. августа 2009.)

[49] MARC 21 Format for Classification Data: Table of Contents (Network Development and MARC Standards Office, Library of Congress) доступно на: http://www.loc.gov/marc/classification/eccdhome.html (прегледано 31. августа 2009.)

[50] MARC 21 Format for Community Information: Table of Contents (Network Development and MARC Standards Office, Library of Congress, доступно на:

http://www.loc.gov/marc/community/eccihome.html (прегледано 31. августа 2009.)

[51] MARC 21 XML Schema (MARCXML), Official Web Site, Library of Congress, 2003, доступно на: http://www.loc.gov/standards/marcxml (прегледано 31. августа 2009.)

[52] MARC21Slim XML Schema, доступно на: http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd (прегледано 31. августа 2009.)

[53] Open Archives Initiative, доступно на: http://www.openarchives.org/ (прегледано 31. августа 2009.)

[54] OAI schema, доступно на: http://www.openarchives.org/OAI/oai_marc.xsd (прегледано 31. августа 2009.)

[55] MagicDraw, доступно на: http://www.magicdraw.com/ (прегледано 14 септембра 2009.)

Биографија

Бојана Димић Сурла рођена је 13.05.1982. године у Сомбору. На Природно-математички факултет Универзитета Новом Саду, одсек за математику, смер дипломирани информатичар уписала се школске 2001/2002. године. У периоду од 2001-2005. године положила је све испите предвиђене планом и програмом са просечном оценом 9.76 (девет и 76/100). Дипломски рад одбранила је у септембру 2005. године са оценом 10 (десет).

Последипломске студије уписала је школске 2005/2006. године на Природно-математичком факултету у Новом Саду, смер информатика. Све испите предвиђене планом и програмом положила је са просечном оценом 10 (десет). Магистарску тезу под насловом „XML едитор за обраду библиографске грађе“ одбранила је у јуну 2007. године и стекла академски назив магистра информатичких наука.

У периоду од фебруара 2006. до фебруара 2007. године била је стипендиста Министарства науке и заштите животне средине Републике Србије. У марту 2007. године засновала је радни однос на Природно-математичком факултету у Новом Саду на радном месту истраживач-приправник и ангажована је на пројекту *Апстрактни модели и примена у рачунарским наукама*, који финансира Министарство науке и заштите животне средине Републике Србије. Изабрана је у звање истраживач-сарадник за ужу научну област Информациони системи на Природно-математичком факултету у Новом Саду на три године почевши од 01. јануара 2008. године.

Држала је вежбе из предмета Информациони системи 1, Информациони системи 2 и Вештачка интелигенција 1 на основним студијама, и из предмета Информациони системи 2 на мастер студијама.

Има десет објављених научних радова од којих су два рада у међународном часопису са SCI листе, један рад у часопису националног значаја, два рада са међународног скупа штампана у целини, једна монографија од националног

значаја и четири рада са скупа националног значаја штампана у целини. Сви радови припадају области дисертације.

Одлично чита, пише и говори енглески језик.

УНИВЕРЗИТЕТ У НОВОМ САДУ

ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број: **РБР**	
Идентификациони број: **ИБР**	
Тип документације: **ТД**	Монографска документација
Тип записа: **ТЗ**	Текстуални штампани материјал
Врста рада: **ВР**	Докторска дисертација
Аутор: **АУ**	Бојана Димић Сурла
Ментор: **МН**	др Милош Рацковић, редовни професор, ПМФ, Нови Сад
Наслов рада: **НР**	Софтверски систем за каталогизацију по MARC 21 формату
Језик публикације: **ЈП**	српски (ћирилица)
Језик извода: **ЈИ**	српски/енглески
Земља публиковања: **ЗП**	Србија

Уже географско подручје:	Војводина
УГП	
Година:	2009
ГО	
Издавач:	Ауторски репринт
ИЗ	
Место и адреса:	Природно-математички факултет, Трг Доситеја Обрадовића 4, Нови Сад
МА	
Физички опис рада:	(8/240/111/0/107/0/0)
ФО	(број поглавља/страна/лит.цитата/ табела/слика/графика/прилога)
Научна област:	Информатика
НО	
Научна дисциплина:	Информациони системи
НД	
Предметна одредница/ кључне речи:	Каталогизација, MARC 21, UML, Xtext, EMF, Eclipse, plug-in технологија, RCP
ПО	
УДК	
Чува се:	Библиотека Департмана за математику и информатику ПМФ-а у Новом Саду
ЧУ	
Важна напомена:	Нема
ВН	
Извод:	Извршено је моделирање и имплементација софтверског система за каталогизацију по MARC 21 формату. За реализацију система коришћен је обједињени процес за развој софтвера, развој заснован на моделу и развој заснован на софтверским компонентама. Моделирање је извршено у CASE алату
ИЗ	

MagicDraw верзија 16.0 који подржава UML 2.0. Имплементација је реализована коришћењем Eclipse plug-in технологије и програмског језика Јава.

У софтверском алату Xtext специфицирана је граматика за опис модела MARC 21 записа. На основу ове граматике генерисан је основни едитор и EMF модел. Основни едитор је проширен додатним спецификацијама над EMF моделом. То су следеће спецификације: ограничења на структуру и садржај библиографских записа коришћењем језика Check; темплејти за трансформацију записа у форму каталошког листића коришћењем језика Xpand; понуда предефинисаног скупа података за унос у језику Xtend.

Извршено је проширење основног едитора додатним функционалностима система за каталогизацију: приказ података о библиографском формату, унос локацијских података, експорт и импорт записа, приказ каталошких листића и библиотечко окружење.

Коришћењем RCP технологије генерисана је софтверска компонента за каталогизацију која се може користити у различитим библиотечким информационим системима.

Датум прихватања теме од НН већа: 11. 06.2009.

ДП

Датум одбране:

ДО

Чланови комисије:

КО

Председник:	др Драган Машуловић, ванредни професор, ПМФ, Нови Сад
члан:	др Милош Рацковић, редовни професор, ПМФ, Нови Сад

члан:	др Синиша Нешковић, доцент, ФОН, Београд
члан:	Др Зора Коњовић, редовни професор, ФТН, Нови Сад
члан:	Др Бранко Милосављевић, вандрендни професор, ФТН, Нови Сад

UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCE

KEY WORDS DOCUMENTATION

Accession number:

ANO

Identification number:

INO

Document type: Monograph publication

DT

Type of record: Textual printed material

TR

Content code: Doctoral dissertation

CC

Author: Bojana Dimić Surla

AU

Mentor/comentor: Miloš Racković, Ph. D., full professor

MN

Title: Software system for MARC 21 cataloguing

TI

Language of text: Serbian (Cyrilic)

LT

Language of abstract: English

LA

Country of publication: Serbia

CP

Locality of publication: Vojvodina

LP

Publication year:	2009
PY	
Publisher:	Author's reprint
PU	
Publication place:	Faculty of Science and Mathematics, Trg Dositeja Obradovića 4, Novi Sad
PP	
Physical description:	(8/240/111/0/107/0/0)
PD	(chapters/pages/literature/tables/ pictures/graphs/appendix)
Scientific field:	Informatics
SF	
Scientific discipline:	Information Systems
SD	
Subject/ Key words:	Cataloguing, MARC 21, UML, Xtext, EMF, Eclipse, plug-in technology, RCP
SKW	
UC	
Holding data:	Library of Department of Mathematics and Informatics, Trg Dositeja Obradovića 4
HD	
Note:	None
N	

Abstract:

AB Modelling and implementation of software system for MARC 21 cataloguing have been done. Unified software development process is used as well as model-driven software development and component-based software development. System modelling is done in CASE tool MagicDraw (version 16.0) which supports UML 2.0. System implementation is realised using Eclipse plug-in

technology and Java programming languages.

Software tool Xtext is used for specification of MARC 21 record grammar. On the basis of this grammar the basic editor and its EMF model have been generated. The basic editor is extended with additional specifications on generated EMF model. Those specifications are: constraints on structure and content of bibliographic record written in Check language; Xpand templates for transforming records into cataloguing cards; content assist extensions written in Xtend.

Addition functionalities of cataloguing system are also added to basic editor, and those are: showing data about MARC 21 format, entering holdings data, export and import of records, showing cataloguing cards and librarian environment.

At the end, RCP technology is used for generating software component for cataloguing that can be used in different library information systems.

Accepted by the Scientific Board:	June 11, 2009.
ASB	
Defended on:	
DE	
Thesis defend board:	
DB	
President:	Dragan Mašulović, Ph. D., associate professor, Faculty of Science and Mathematics, Novi Sad
Member	Miloš Racković, Ph. D., full. prof., Faculty of Science and Mathematics, Novi Sad
Member:	Siniša Nešković, Ph. D., assistant professor, Faculty of Organizational Sciences, Belgrade

Member:	Zora Konjović, Ph. D., full. prof., Faculty of Engineering, Novi Sad.
Member:	Branko Milosavljević, Ph. D., associate professor, Faculty of Engineering, Novi Sad