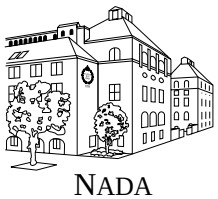
KTH Numerisk analys
och datalogi

Analys av stöd i en metadata-editor för Resource Description Framework

Jacob Widén

TRITA-NA-E04031



NADA

Numerisk analys och datalogi
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Analys av stöd i en metadata-editor för Resource Description Framework

Jacob Widén

TRITA-NA-E04031

Examensarbete i datalogi om 20 poäng
vid Programmet för datateknik,
Kungliga Tekniska Högskolan år 2004
Handledare på Nada var Ambjörn Naeve
Examinator var Yngve Sundblad

Analys av stöd i en metadata-editor för Resource Description Framework

Sammanfattning

Det finns ett växande behov inom e-learning-samhället av effektivare sätt att kunna söka och dela lärobjekt, till exempel föreläsningsanteckningar eller filmer för undervisning, för olika Learning Management Systems. Det har kommit några nya applikationer för att lösa detta behov, och de använder sig av metadata för att beskriva de olika lärobjekten. Denna metadatastandard heter Learning Object Metadata (LOM). Det här examensarbetet syftar till att finna de problem som uppstår när LOM-editorn ImseVimse utökas för att stödja en bindning mot Resource Description Framework (RDF). Bindningen, som kallas LOM/RDF-bindningen, är en beskrivning av hur varje LOM-element i LOM-standarden ska läsas och skrivas av ImseVimse.

De problem jag funnit, när jag implementerat LOM/RDF-bindningen i ImseVimse, kan delas in i tre olika kategorier. Den första kategorin är de problem som beror på hur LOM/RDF-bindningen är modellerad för de olika LOM-elementen. Den andra kategorin är de problem som är beroende av ny funktionalitet i ImseVimse. Den tredje och sista kategorin är de problem som uppstår när användaren öppnar eller sparar en LOM-instans.

Analysis of support in a metadata editor for Resource Description Framework

Abstract

Within the e-learning community there is a growing demand for better ways to search and share Learning Objects, for example lecture notes or educational movies, for the different Learning Management Systems. There are some new applications available to do this, and they rely on the metadata describing the different Learning Objects. The metadata standard is called Learning Object Metadata (LOM). This master's project aims at finding the problems arisen when the LOM-editor ImseVimse is extended with a binding for the Resource Description Framework (RDF). This binding, called the LOM/RDF-binding, is a description of how each and every LOM-element within the LOM-standard should be read and written by ImseVimse.

The problems I found, while implementing the LOM/RDF-binding in ImseVimse, could be divided in three different categories. The first category is the problems that depend on how the LOM/RDF-binding have modelled the different LOM-elements. The second category is the problems that depend on a need for new functionality in ImseVimse. The third and last category is the problems that occur when the user opens or saves a LOM-instance.

Innehållsförteckning

Inledning

Denna rapport sammanfattar ett examensarbete på institutionen för Numerisk Analys och Datalogi (Nada), KTH. Arbetet är en del av civilingenjörsutbildningen i datateknik på KTH. Uppdragsgivare är Centrum för användarorienterad IT-design (CID) på Nada.

Bakgrund

Knowledge Management Research-gruppen[1] (KMR) är en forskargrupp på CID[2] på Nada[3]. KMR bedriver forskning inom flera fält där det underliggande målet är att skapa nya kraftfulla sätt att strukturera och kommunicera information, inom undervisnings-, industri- och administrativa miljöer. Ett av de områden som KMR har forskat kring är *Learning Management Systems* (LMS). KMR har utvecklat editorn *ImseVimse* för att underlätta modifiering och utveckling av metadata för *lärobjekt* till LMS. Genom att vidareutveckla ImseVimse, så att den förutom ren XML-metadata även kan hantera RDF/XML-metadata, blir ImseVimse den första editorn för metadata för lärobjekt som kan hantera båda dessa metadatatyper.

Learning Management Systems

Learning Management Systems är system som använder sig av teknikstödd inlärning. Några exempel på LMS är datorbaserade träningssystem, interaktiva lärmiljöer, intelligenta datorstödda instruktionssystem, distanslärsystem och samarbetande lärmiljöer.

Lärobjekt

Den information som LMS använder sig av, till exempel kursmaterial, heter lärobjekt (eng. Learning Object). Lärobjekt är enheter, digitala eller icke-digitala, som kan användas eller refereras av LMS. Några exempel på lärobjekt är multimedia-innehåll, instruktioner, utbildningsmål och information om personer, organisationer eller händelser som refereras under teknikstödd inlärning.

Utvecklare av material för LMS har ett stort utbud av mjukvaruverktyg till sitt förfogande. Genom att använda dessa verktyg kan utvecklare skapa nytt material för LMS utan att för den skull behöva besitta någon större programmerings-kunskap. Det stora utbudet av mjukvaruverktyg har dock resulterat i att materialet som skapats inte har kunnat sökas och användas på ett uniformt sätt. För att lösa detta problem har IEEE[4] skapat *Learning Object Metadata*.

Metadata

Metadata är data om data, det vill säga beskrivande information om data. Metadata beskriver innehållet, kvalitén, tillståndet och annan karakteristisk information om

[1] http://kmr.nada.kth.se/

[2] http://cid.nada.kth.se/

[3] http://www.nada.kth.se/

[4] http://www.ieee.org/

den data som beskrivs. Det är helt nödvändigt att använda sig av metadata om man vill bevara en datas användbarhet över en längre tid. Metadata kan innehålla information om hur den data som beskrivs samlades in eller hur den har hanterats, vilket kan vara helt avgörande information för hur den ska kunna användas i framtiden.

Genom webben får vi idag, på ett sätt som tidigare inte varit möjligt, tillgång till distribuerad information. Metadata hjälper oss att enklare hitta och hantera informationen. Med mer metadata tillgänglig kommer det att bli lättare att söka information eftersom sökmotorerna då i sin tur har tillgång till mer information. Då går det att göra mer exakta och detaljerade sökningar. Det blir även möjligt att utveckla agenter, det vill säga automatiserade program, som går igenom webben och letar information åt oss eller utför affärstransaktioner åt oss med hjälp av just metadata.

Som ett enkelt exempel kan antas att man letar efter en film som finns tillgänglig på webben. Om filmen inte är beskriven med metadata är enda sättet att finna den att man lyckas med en fritextsökning i till exempel sökmotorn Google. Om den däremot är beskriven med metadata så förenklas sökningen avsevärt. Det går då att söka på de egenskaper man känner till om filmen. Det kan till exempel vara regissörens namn, skådespelares namn, filmens utgivningsår och vilket språk filmen är inspelad på. Om då en eller flera av dessa kriterier överensstämmer med den metadata som beskriver filmen, så har man chansen att hitta filmen.

Learning Object Metadata

Learning Object Metadata (LOM) är en metadatastandard för lärobjekt framtagen av IEEE. LOM släpptes i version 1.0 i juli 2002, kallad LOMv1.0. Syftet med denna standard är att studenter och lärare på ett enkelt sätt ska kunna söka, utvärdera och använda olika lärobjekt [3]. Detta uppnås genom möjligheten att dela och utbyta lärobjekt mellan olika LMS. LOM-standarden består av LOM-element indelade i de nio kategorierna:

- **General:** Innehåller generell information som beskriver lärobjektet i sin helhet.
- **Life Cycle:** Innehåller information som beskriver historien och det aktuella tillståndet för lärobjektet och dom enheter som har påverkat lärobjektet.
- **Meta-Metadata:** Innehåller information om metadatainstansen själv, istället för det lärobjekt den beskriver.
- **Technical:** Innehåller information om de tekniska kraven och den tekniska karaktären på lärobjektet.
- **Educational:** Innehåller information om huvuddragen vad avser undervisning och pedagogik för lärobjektet.
- **Rights:** Innehåller information om rättigheterna vad avser intellektuella egenskaper för lärobjektet samt hur det får användas.
- **Relation:** Innehåller information om de eventuella relationer det här lärobjektet har mot andra lärobjekt.
- **Annotation:** Innehåller kommentarer om hur lärobjektet ska användas i undervisningen, samt när och av vem dessa kommentarer gjordes.
- **Classification:** Innhåller information om vart lärobjektet hamnar inom ett visst klassifikationssystem.

Ett exempel på ett LOM-element är *Context,* som återfinns under LOM-kategorin *Educational*, som beskriver i vilket sammanhang det aktuella lärobjektet ska

användas. Genom användningen av LOM går det på ett enkelt sätt att söka efter lärobjekt som till exempel används inom ett visst sammanhang.

Datatyper

LOM använder sig av flera olika datatyper. Den grundläggande datatypen är CharacterString vilket är en textsträng. Vidare finns de sammansatta datatyperna LangString, DateTime, Duration och Vocabulary. De två intressantaste av dessa är LangString och Vocabulary som här beskrivs lite närmare.

LangString

En LangString är en datatyp vars värde representerar en eller flera teckensträngar. Dessa teckensträngar är semantiskt ekvivalenta. Med det menas att de är olika översättningar av något. En LangString består av dels en CharacterString som innehåller själva textsträngen och dels en CharacterString som innehåller information om vilket språk textsträngen är på. Den sistnämnda CharacterStringens värde utgörs av standardiserade språkkoder.

Vocabulary

Vocabulary, eller det svenska ordet vokabulär som jag använder i rapporten, är definierat för flera LOM-element. Vokabulären använder sig av en lista med lämpliga värden för ett visst LOM-element. Det är tillåtet att använda andra värden också men metadata som använder sig av de rekommenderade värdena kommer att vara lättare för andra användare eller system att förstå rent semantiskt.

Vokabulären består dels av en CharacterString för värdet och dels av en CharacterString som anger källan för värdet. Anledningen till att det finns en CharacterString för källan är att man ska kunna urskilja huruvida ett värde tillhör den rekommenderade listan eller om det kommer från en annan definition.

Bindningar

LOM-standarden är en generell standard. LOM-standarden kan representeras i flera olika format. Det kan vara till exempel HTML, SQL-tabeller eller XML. För närvarande finns och arbetas det endast med två bindningar. Den ena är mot XML [4] och den andra är mot Resource Description Framework (RDF). Båda dessa ligger för närvarande för omröstning om godkännande inom respektive arbetsgrupp. En bindning är det sätt på vilket man binder samman den generella LOM-standarden med ett visst format, det vill säga hur man läser in och skriver en LOM-instans till det formatet. Det som är viktigt att tänka på när man skapar en bindning är hur de olika LOM-elementen på ett enkelt och unikt sätt ska representeras i det aktuella formatet, utan att för den skull förlora semantik gentemot LOM-standarden.

ImseVimse

ImseVimse är en editor för LOM. Den är utvecklad av KMR-gruppen och stödjer fram till det här examensarbetet endast LOM-bindningen mot XML. När jag är klar med mitt examensarbete är det tänkt att ImseVimse även ska stödja en bindning mot RDF. ImseVimse är skriven i Java och finns tillgänglig som öppen källkod via SourceForge[5].

[5] http://sourceforge.net/

Problemställning

Detta kapitel beskriver målet med mitt examensarbete och de problem jag skall lösa. Vidare beskrivs den metod jag har valt att använda för att lösa problemen.

Problem

Examensarbetet behandlar följande två problem.

- Vilka problem uppstår när LOM-editorn ImseVimse utökas med en bindning mot RDF? Den *LOM/RDF-bindning* som skall implementeras är framtagen i samarbete med flera internationella forskningsgrupper under ledning av Mikael Nilsson på CID.
- Uppstår det några problem som kommer att kräva modifikationer i ImseVimse för att möjliggöra införandet av LOM/RDF-bindningen?

Syfte

Ovanstående frågor är intressanta, dels för att utreda vilka problem som finns med den nuvarande LOM/RDF-bindningen, och dels för att se vad LOM/RDF-bindningen kommer att innebära för de editorer där man försöker implementera den.

Examensarbetets mål

Målet med examensarbetet är att finna vilka problem som uppstår när ImseVimse utökas med LOM/RDF-bindningen.

Metod

Min metod för att lokalisera vilka problem som uppstår när ImseVimse utökas med LOM/RDF-bindningen, är att implementera LOM/RDF-bindningen och notera vilka problem som uppkommer. En anledning till att jag väljer denna metod, är att det finns ett önskemål från min uppdragsgivare att jag ska försöka implementera LOM/RDF-bindningen.

En alternativ metod skulle kunna ha varit att analysera ImseVimses befintliga programkod och programstruktur, och utifrån det försöka dra slutsatser om vilka problem som uppstår vid ett införande av LOM/RDF-bindningen. Denna metod skulle troligtvis inte lyckas fånga upp lika många problem, som den metod jag väljer, eftersom det är svårt att förutse vilka problem som uppstår vid en implementering.

Den förutsättning som måste vara uppfylld för att jag ska kunna använda mig av min valda metod är att det går att implementera huvuddelen av LOM/RDF-bindningen inom en rimlig tid under mitt examensarbete. Min uppdragsgivare har förklarat att ImseVimse är strukturerat på ett sätt som underlättar införandet av nya bindningar. Det finns även ett färdigt ramverk för semantisk webb, kallat *Jena*, för att hantera RDF-modeller, som jag kan använda mig av. Med denna vetskap har jag uppskattat tiden för implementeringen, och planeringen inför densamma, till sex

veckor. Då denna tid kan anses vara rimlig för mitt examensarbete, tycker jag att förutsättningen för att jag ska kunna använda den valda metoden, är uppfylld.

För att uppnå målet med examensarbetet, delades arbetet in i följande delar:

1. **Inläsningsdel**
   Här har arbetet bestått av att läsa och sätta mig in i det material jag har blivit rekommenderad av min handledare. Detta material är främst syftat till att jag ska få en grundläggande förståelse om metadata, den semantiska webben, hur RDF fungerar och hur LOM/RDF-bindningen är utformad. Utifrån detta material har jag även varit tvungen att fördjupa mig inom vissa sidospår.
2. **Planering av implementeringen**
   Här har arbetet främst bestått av att analysera ImseVimses befintliga programkod och programstruktur, för att se hur jag själv ska kunna bygga vidare på den. Jag har även varit tvungen att lära mig hur Jena fungerar.
3. **Implementering av uppdateringar i ImseVimse för stöd av LOMv1.0**
   För att kunna implementera LOM/RDF-bindningen, vilken bygger på LOMv1.0, är det nödvändigt att uppdatera ImseVimse så att programmet följer LOMv1.0.
4. **Implementering av LOM/RDF-bindningen i ImseVimse**
   Detta steg utgjorde den största delen av implementeringen. Det har varit viktigt att hela tiden göra noteringar om vilka problem jag har stött på för att ha ett bra underlag när det väl var dags att skriva rapporten.
5. **Rapportskrivning**
   Avslutningsvis så sammanställde jag problemen som kommit fram under arbetet.

Avgränsningar

Undersökningen av vilka problem som uppstår med implementeringen av LOM/RDF-bindningen syftar endast till att lokalisera problemen och deras orsak. Målet är inte att finna en lösning för dem. Om jag har ett förslag på lösning är jag dock välkommen att beskriva det. Arbetet med att uppdatera ImseVimse så att det stödjer LOMv1.0 kommer att resultera i att bindningen mot XML upphör att fungera. Arbetet med att uppdatera XML-bindningen så att den fungerar igen, ligger utanför mitt examensarbete.

Resource Description Framework

Resource Description Framework är ett språk för att beskriva information om resurser på webben. I det här kapitlet beskriver jag RDF genom att förklara bakgrunden till varför det finns samt hur det är uppbyggt och tänkt att fungera. Detta görs för att LOM/RDF-bindningen och de problem jag beskriver i de följande kapitlen ska vara förståliga.

Bakgrund

RDF är resultatet av en vidareutveckling av Platform for Internet Content Selection[6] (PICS). PICS används för att möjliggöra filtrering av webbsidor. På grund av begränsningar i specifikationen av PICS och behovet av ny funktionalitet för att mer generellt kunna beskriva resurser på Internet startade World Wide Web Consortium[7] (W3C) en arbetsgrupp för att ta fram en lösning för de nya behoven.

Den nya arbetsgruppen gick under namnet PICS-Next Generation. Redan i ett tidigt skede av arbetet stod det klart att den nya infrastrukturen som skapades stödde fler tillämpningar än det ursprungligen var tänkt. Som ett resultat av det skapades arbetsgruppen Resource Description Framework, vars arbete även inkluderade de nya tillämpningarna.

RDF är resultatet av ett behov att kunna erbjuda robust och flexibel arkitektur för stöd av metadata på webben [6]. Utvecklingen av RDF har en betydande påverkan på de standarder som utvecklas inom e-learning [8]. Trots att utvecklingen av RDF är starkt inspirerad av specifikationen för PICS så är det ingen enskild individ eller organisation som har skapat RDF, utan RDF är utvecklat under samarbete mellan många parter. Flera av de företag som är medlemmar i W3C har bidragit med intellektuellt kapital i utvecklingen.

Användning

Användningsområdena för RDF är otaliga. Det finns till exempel projekt med inriktning på hur man lagrar och utbyter metadata som beskriver digitala ljud, filmer och bilder, eller projekt där man automatiserar publicering och utbyte av nyheter på olika plattformar.

RDF-modellen

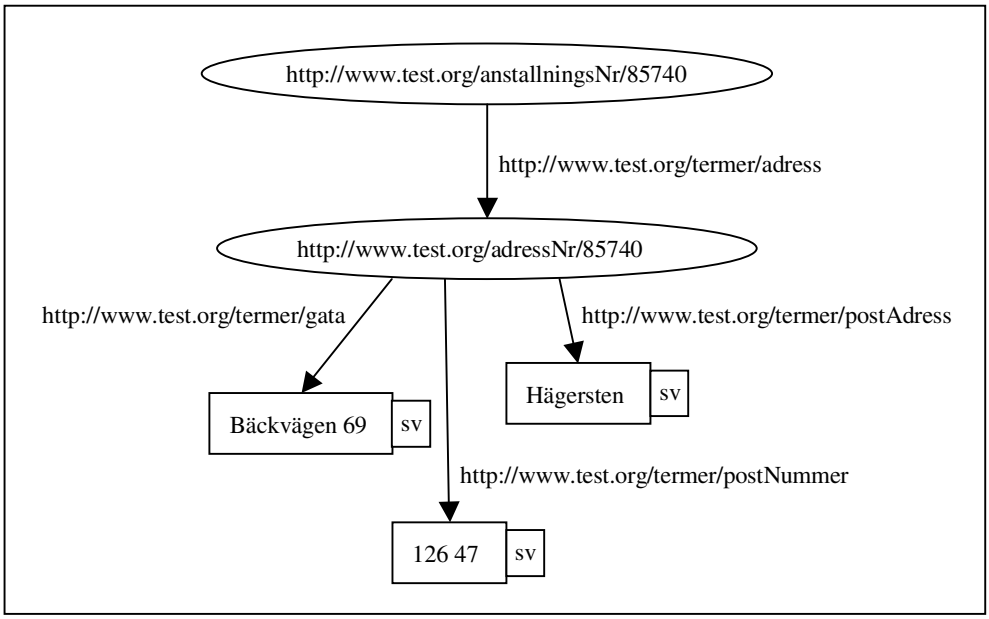
RDF erbjuder en modell för att beskriva *resurser*. Den är framför allt tänkt att användas för att representera metadata om webbresurser. Resurserna har egenskaper i form av attribut eller kännetecken. I RDF är en resurs definierad som ett objekt som är unikt identifierbart genom en Uniform Resource Identifier[8] (URI). Resursernas egenskaper identifieras genom *egenskapstyper*, vilka binds till ett

[6] http://www.w3.org/PICS/

[7] http://www.w3.org/

[8] http://www.w3.org/Addressing/Activity

värde. Egenskapstyperna uttrycker relationen mellan värdet och resursen. Värdet kan vara antingen atomärt, i form av en sträng eller ett tal, och återfinns då i en *Literal*. Eller det kan vara en annan resurs, som i sin tur kan ha olika egenskaper. En samling av egenskaper kallas *beskrivning*. Figur 1 är ett exempel på en RDF-graf av en RDF-beskrivning föreställande ett företags modellering av en anställds adress.

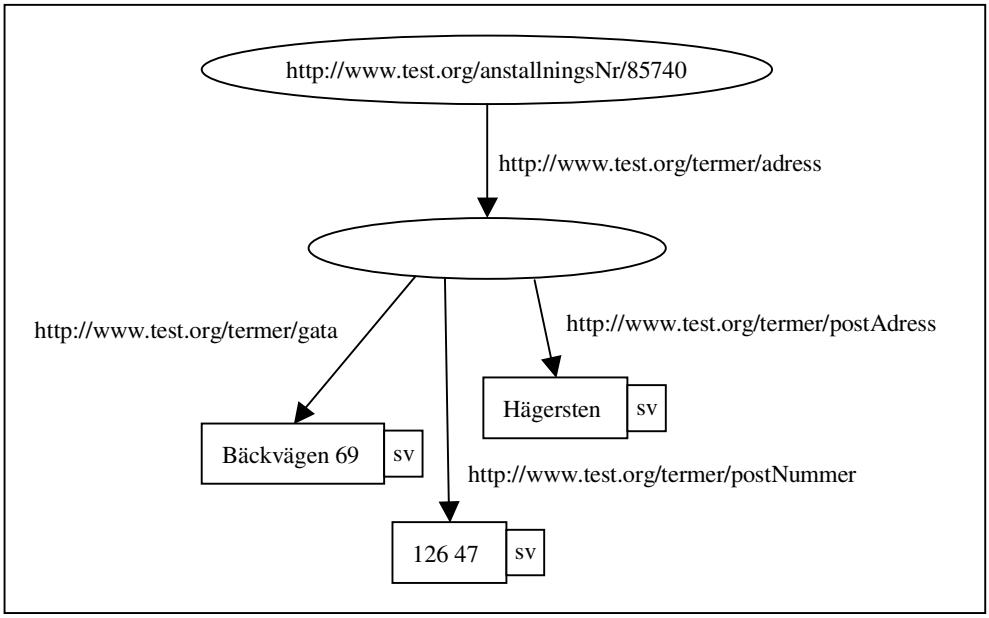


Figur 1. RDF-graf föreställande ett företags modellering av en anställds adress.

Den övre resursen i figur 1, med URI `http://www.test.org/anstallningsNr/85740`, representerar en anställd. Den anställdes adress är modellerad via egenskapstypen `http://www.test.org/termer/adress` till en resurs med URI `http://www.test.org/adressNr/85740`. Denna resurs har i sin tur tre egenskapstyper med tillhörande värden i Literaler knutna till sig. Det är en Literals språkkod som anges i den lilla rutan bredvid Literalen.

Anonyma noder

Om man tittar närmare på figur 1, så kan man se att den nedre av de två resurserna endast finns för att strukturera de olika typerna av adressvärden. Om den inte hade funnits, och de tre adressvärdena via sina respektive egenskapstyper hade varit knutna direkt till den övre resursen, så hade det inte gått att se att de olika adressvärdena har med varandra att göra. Man hade då förlorat semantik i modellen. Men genom att förena de tre adressvärdena till den nedre resursen skapar man en struktur som representerar den anställdes adress. Detta sätt att strukturera en RDF-modell är mycket vanligt. Dessa mellanliggande resurser representerar ofta ingenting. Den enda funktion de har, är att de skapar struktur. I figur 1 är den nedre resursen inte den anställdes adress utan endast en nod som binder samman de delar som utgör den faktiska adressen. Den skulle lika gärna kunna vara utan URI. En sådan resurs brukar kallas för *anonym nod*, och saknar URI. En RDF-modell kan innehålla många olika anonyma noder. När ett dataprogram ska hantera en RDF-modell måste det dock skapa unika identifierare för de olika anonyma noderna för att kunna särskilja dem. Figur 2 nedan innehåller samma exempel som figur 1. Men denna gång är det modellerat med hjälp av en anonym nod.



Figur 2. RDF-graf föreställande ett företags modellering av en anställds adress. Adressen för den anställde är modellerad med hjälp av en anonym nod.

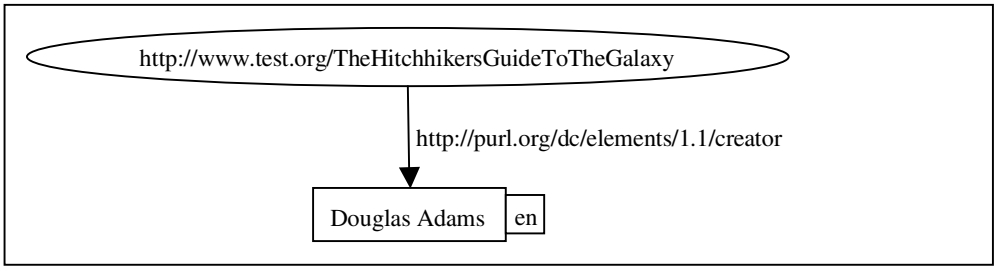
RDF-syntax

RDF erbjuder en enkel men ändå kraftfull modell för att beskriva resurser men behöver en syntax för att möjliggöra lagring och utbyte av filer. RDF använder sig av en XML-syntax definierad för RDF, kallad RDF/XML, för utbyte och hantering av metadata. XML är en delmängd av den internationella texthanteringsstandarden Standard Generalized Markup Language (SGML) och är framförallt tänkt att användas på webben. RDF/XML får flera fördelar genom att stödja sig på XML. Det är lätt att kontrollera att syntaxen är korrekt. Syntaxen är inte bara läsbar för systemet utan även för användaren. Man får även möjlighet att representera komplexa strukturer.

RDF-schema

RDF erbjuder möjligheten för olika organisationer att beskriva semantik för att användas med RDF. Till exempel kan begreppet ”Författare” ha vidare eller smalare betydelse inom olika organisationer beroende på vilka krav organisationen ställer. Semantiken för olika begrepp och beskrivningen hur de ska användas, ligger i så kallade RDF-scheman. För att särskilja de olika organisationernas semantik åt använder sig RDF/XML av XML-Namespace som binder ett valt namn till en viss organisations RDF-schema.

I figur 3 nedan är boken *The Hitchhiker’s Guide to the Galaxy* modellerad som en resurs. Modellen använder sig av Dublin Cores egenskapstyp *Creator* för att ange vem som skrivit boken, vilken är *Douglas Adams*. Figur 4 nedan innehåller delar av det RDF-schema där Dublin Cores egenskapstyp Creator är definierad. På rad 17 till 26 framgår definitionen av Creator. Anledningen till att Dublin Core använder egenskapstypen Creator för författare framgår av kommentaren på rad 19, vilken är ”An entity primarily responsible for making the content of the resource”.



Figur 3. Modellering av författare till en bok.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE rdf:RDF>
3: <rdf:RDF xmlns:dcterms=http://purl.org/dc/terms/
            xmlns:dc=http://purl.org/dc/elements/1.1/
            xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">


4:    <rdf:Description rdf:about="http://purl.org/dc/elements/1.1/">
5:      <dc:title xml:lang="en-US">The Dublin Core Element Set v1.1 namespace
            providing access to its content by means of an RDF Schema</dc:title>
6:      <dc:publisher xml:lang="en-US">The Dublin Core Metadata
            Initiative</dc:publisher>
7:      <dc:description xml:lang="en-US">The Dublin Core Element Set v1.1 namespace
            provides URIs for the Dublin Core Elements v1.1. Entries are declared
            using RDF Schema language to support RDF applications.</dc:description>
8:      <dc:language xml:lang="en-US">English</dc:language>
9:      <dcterms:issued>1999-07-02</dcterms:issued>
10:     <dcterms:modified>2003-03-24</dcterms:modified>
11:     <dc:source rdf:resource="http://dublincore.org/documents/dces/" />
12:     <dc:source rdf:resource="http://dublincore.org/usage/decisions/" />
13:     <dcterms:isReferencedBy rdf:resource=
            "http://www.dublincore.org/documents/2001/10/26/dcmi-namespace/" />
14:     <dcterms:isRequiredBy rdf:resource="http://purl.org/dc/terms/" />
15:     <dcterms:isReferencedBy rdf:resource="http://purl.org/dc/dcmitype/" />
16:   </rdf:Description>


17:   <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/creator">
18:     <rdfs:label xml:lang="en-US">Creator</rdfs:label>
19:     <rdfs:comment xml:lang="en-US">An entity primarily responsible for making
            the content of the resource.</rdfs:comment>
20:     <dc:description xml:lang="en-US">Examples of a Creator include a person, an
            organisation, or a service. Typically, the name of a Creator should be
            used to indicate the entity.</dc:description>
21:     <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
22:     <dcterms:issued>1999-07-02</dcterms:issued>
23:     <dcterms:modified>2002-10-04</dcterms:modified>
24:     <dc:type rdf:resource=
            "http://dublincore.org/usage/documents/principles/#element" />
25:     <dcterms:hasVersion rdf:resource=
            "http://dublincore.org/usage/terms/history/#creator-004" />
26:   </rdf:Property>
27: </rdf:RDF>
```

Figur 4. RDF-schema innehållande definitionen av Dublin Cores egenskapstyp Creator.

LOM/RDF-bindningen

I det här kapitlet kommer den framtagna LOM/RDF-bindningen för LOM att beskrivas. Dels genom att beskriva hur och varför den är framtagen och dels genom att beskriva hur den är uppbyggd.

Bakgrund

I grunden ligger en vision inom KMR om att vem som helst ska kunna skapa, söka och använda metadata för de olika lärobjekt som finns [7]. KMR har utvecklat flera verktyg för att stödja och driva utvecklingen mot denna vision. Ett par exempel på dessa verktyg är ImseVimse och Edutella vilket är ett RDF-baserat peer-to-peer verktyg för att söka och utbyta lärobjekt. Verktyg som Edutella är helt beroende av metadata för att fungera bra [2]. Allt eftersom RDF har fått en större spridning och vuxit sig starkare, har behovet av att kunna beskriva lärobjekt med RDF ökat. Detta är helt naturligt eftersom RDF från grunden är konstruerat att vara en metadata-arkitektur för Internet.

Utvecklingsarbetet

Utvecklingsarbetet av bindningen inleddes år 2000 inom ramen för IMS Global Learning Consortium[9] och stödde sig till stor del på IMS metadatastandard [5]. Senare överfördes arbetet till LTSC[10] vid IEEE.

Utvecklingsarbetet leds av Mikael Nilsson vid KMR på CID. KMR-gruppens arbete sker även i samarbete med Knowledge Based System[11] på institutionen Information Systems vid Hannovers universitet samt projektet UNIVERSAL[12] vid Wiens universitet.

För tillfället ligger LOM/RDF-bindningen för omröstning om den ska accepteras som den är.

Uppbyggnad

LOM/RDF-bindningen är uppbyggd så att den är direkt kompatibel med Dublin Core[13] och vCard[14] standardernas RDF-bindningar. Ett av målen vid utvecklingen av LOM/RDF-bindningen har varit att i så hög grad som möjligt återanvända dessa standarders RDF-scheman utan att för den skull förlora i överensstämmelse med LOM-standaren [10]. Samtidigt är det viktigt att den valda designlösningen är självförklarande eftersom LOM-instanserna i många fall ska hanteras av programvaror som inte känner till LOM-standarden [1].

---

[9] http://www.imsglobal.org

[10] http://ltsc.ieee.org/

[11] http://www.kbs.uni-hannover.de/ (Tyska)

[12] http://nm.wu-wien.ac.at/universal/universal_wu_memo.shtml (Tyska)

[13] http://dublincore.org/documents/2002/07/31/dcmes-xml/

[14] http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222/

Det lärobjekt som skall beskrivas utgörs av en RDF-resurs som är rot till de övriga RDF-konstellationerna, en så kallad *rotresurs*. De olika LOM-elementen är representerade genom olika RDF-konstellationer. Eftersom varje LOM-element måste vara unikt identifierbart måste de olika RDF-konstellationerna ha en unik uppbyggnad. Se [9] för en fullständig beskrivning av hur de olika LOM-elementen är modellerade.

Namespace använda av LOM/RDF-bindningen

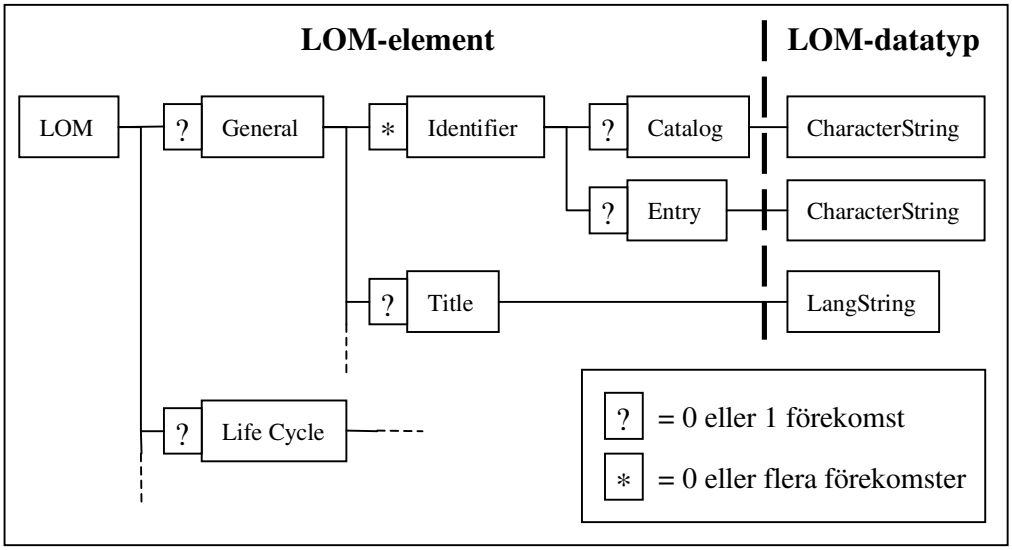
Jag kommer här att lista de olika fördefinierade namespace i LOM/RDF-bindningen som jag använder mig av i rapporten. Jag gör det för att förenkla mina beskrivningar i texten framöver, och inte behöva skriva ut en hel URI varje gång jag nämner en egenskapstyp. Exempelvis skulle `dc:description` motsvara URI `http://purl.org/dc/elements/1.1/description`.

Här följer de namespace jag använder senare i rapporten:

- `rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs = http://www.w3.org/2000/01/rdf-schema#`
- `dc = http://purl.org/dc/elements/1.1/`
- `dcterms = http://purl.org/dc/terms/`
- `lom = http://ltsc.ieee.org/2002/09/lom-base#`
- `lom-life = http://ltsc.ieee.org/2002/09/lom-lifecycle#`
- `lom-edu = http://ltsc.ieee.org/2002/09/lom-educational#`
- `lom-ann = http://ltsc.ieee.org/2002/09/lom-annotation#`
- `lom-cls = http://ltsc.ieee.org/2002/09/lom-classification#`

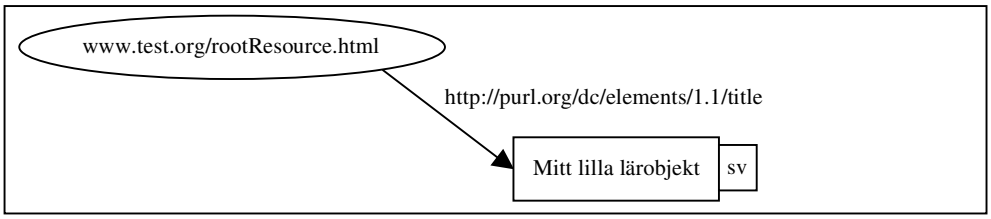
Exempel på LangString

I figur 5 nedan återges definitionen för LOM-elementen *Identifier* och *Title* under LOM-kategorin *General*. Om man tittar närmare på Title kan man se att den använder sig av LOM-datatypen LangString för sitt värde. Som nämnts tidigare kan en LangString innehålla flera semantiskt ekvivalenta textsträngar.

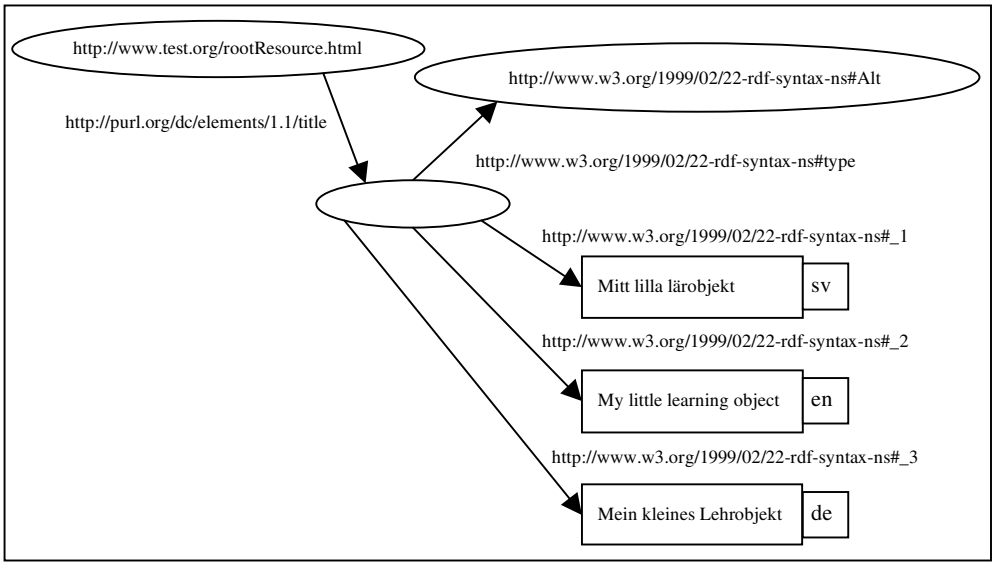


Figur 5. Definitionen av LOM-element 1.1 Identifier och 1.2 Title under LOM-kategorin General.

I LOM/RDF-bindningen modelleras Title genom att den aktuella rotresursen har egenskapstypen `dc:title` som pekar på antingen en Literal eller en mängd Literaler innehållande titeln. Anledningen till att man kan ange flera titlar på lärobjektet är att man ska kunna ange titeln på olika språk. Figur 6 nedan visar hur det ser ut med en Literal och Figur 7 visar hur det ser ut med en mängd Literaler. En mängd representeras med hjälp av RDF:s ALT-resurs, vilket är en konstruktion man använder sig av i RDF när man vill modellera flera alternativ. Jag kommer fortsättningsvis att kalla en uppbyggnad av en eller flera Literaler för en *textbeskrivning*.



Figur 6. Textbeskrivning med en Literal.

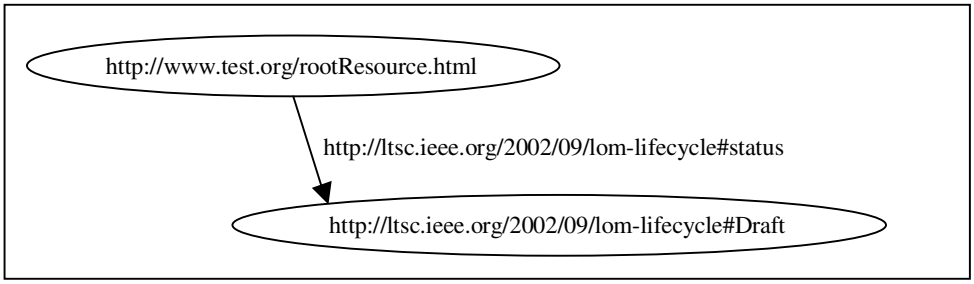


Figur 7. Textbeskrivning med flera Literaler.

Exempel på vokabulärer

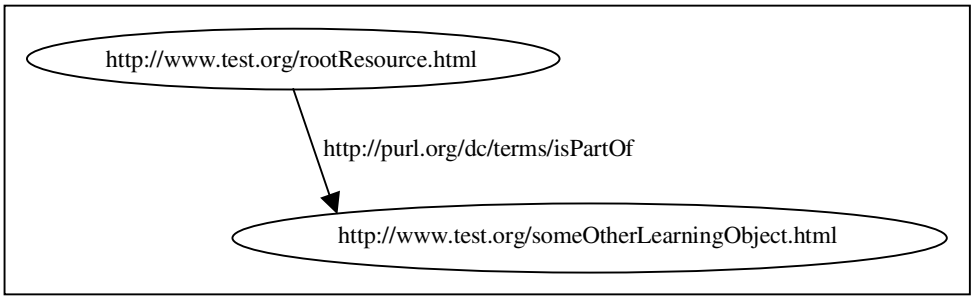
En vanligt förekommande datatyp i LOM-standarden är vokabulären. Den är som tidigare nämnts tudelad med en källa och ett värde. I LOM/RDF-bindningen har man valt att representera en vokabulär med hjälp av en URI. Då utgör källan den del av URI:n som är *path* och värdet den del av URI:n som är *local name*. Till exempel så skulle en URI som representerar en vokabulär och har följande utseende, `http://www.test.com/Tidskrift`, innebära att källan på vokabulären är `http://www.test.com/` och värdet är `Tidskrift`.

I LOM/RDF-bindningen är vokabulärer modellerade på två olika sätt. Antingen utgör de URI:n för en resurs knuten till en viss egenskapstyp eller så utgör de URI:n för en egenskapstyp. I figur 8 är LOM-elementet Status, vilket är en vokabulär för att beskriva tillståndet på lärobjektet, under LOM-kategorin Life Cycle modellerat som en resurs knuten till egenskapstypen `lom-life:status`. Resursens URI `http://ltsc.ieee.org/2002/09/lom-lifecycle#Draft` utgör vokabulären.



Figur 8. Vokabulär modellerat med hjälp av en resurs.

I figur 9 är LOM-elementet Kind, vilket är en vokabulär för att beskriva typen av relation lärobjektet har gentemot ett annat lärobjekt, under LOM-kategorin Relation modellerat med egenskapstypen `dcterms:isPartOf` knutet till en resurs som representerar det relaterade lärobjektet.



Figur 9. Vokabulär modellerat med hjälp av en egenskapstyp.

Implementeringen av LOM/RDF-bindningen

I det här kapitlet ska jag beskriva hur implementeringen av LOM/RDF-bindningen har gått till och vilka problem jag har stött på under arbetets gång.

Arbetsgången

Implementeringsarbetet har varit indelat i tre delar.

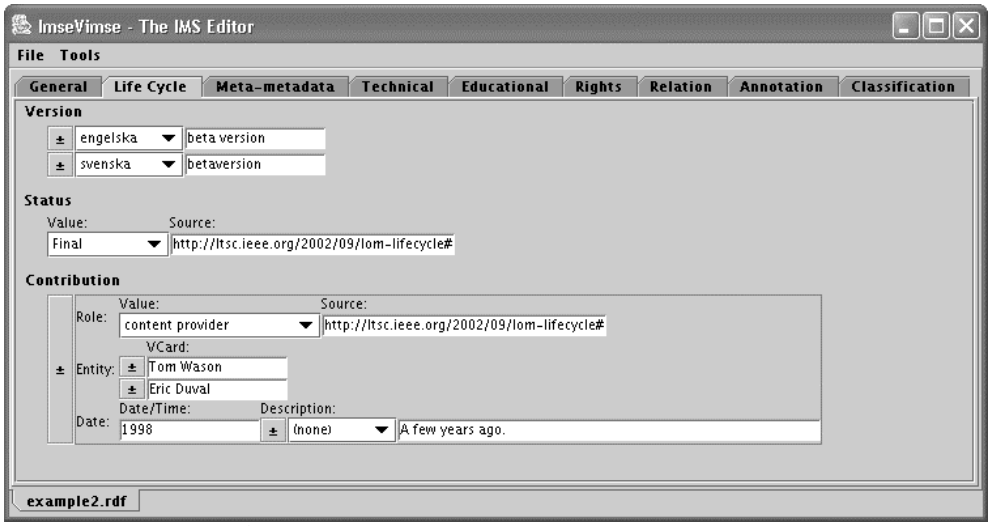
Den första delen av implementeringsarbetet bestod i att planera implementeringen. Här har arbetet främst bestått i att analysera och förstå ImseVimses befintliga programkod och programstruktur. Ett av kraven som har ställts på mitt arbete var just att jag skulle följa den befintliga programstrukturen både vad gäller den logiska och semantiska uppbyggnaden men även vad gäller syntaxen i form av namngivning och liknande. Ett annat krav på arbetet var att jag skulle använda mig av Jena för att hantera RDF-modellen i ImseVimse. Jag var därför tvungen att lära mig hur Jena fungerar.

Den andra delen av implementeringsarbetet bestod i att uppdatera ImseVimse så att det stödjer LOMv1.0. ImseVimse stödde ett tidigare utkast av LOM-standarden, vilken nu befinner sig i version 1.0. Eftersom LOM/RDF-bindningen är designad för LOMv1.0 var det nödvändigt att uppdatera ImseVimse så att den hanterade rätt version av LOM. Denna uppdatering innebar förändringar både i det grafiska gränssnittet, men också i den interna LOM-modellen. En följd av detta arbete blev att ImseVimses befintliga implementering av XML-bindningen inte längre fungerade. Det behöver göras några modifieringar av denna för att den ska stödja LOMv1.0. Detta arbete faller dock utanför ramen av mitt examensarbete.

Den tredje och sista delen av implementeringsarbetet bestod i att implementera LOM/RDF-bindningen i ImseVimse. Den här delen var den klart största av implementeringsarbetet. Inläsning från en RDF-modell och skrivning till en RDF-modell måste göras så att de följer designen av LOM/RDF-bindningen. Jag har hela tiden noterat de olika problem jag stött på för att ha ett bra underlag inför rapportarbetet. Efter anvisning från min handledare har jag inte implementerat LOM-kategorin Meta-Metadata.

Systemarkitektur

ImseVimse är skrivet i Java och har utvecklats av KMR. ImseVimse finns numera tillgängligt via SourceForge, vilket är den största förvaringsplatsen för öppen källkod på Internet. Figur 10 nedan visar det grafiska gränssnittet för ImseVimse. Figuren visar de LOM-element som finns under LOM-kategorin Life Cycle.



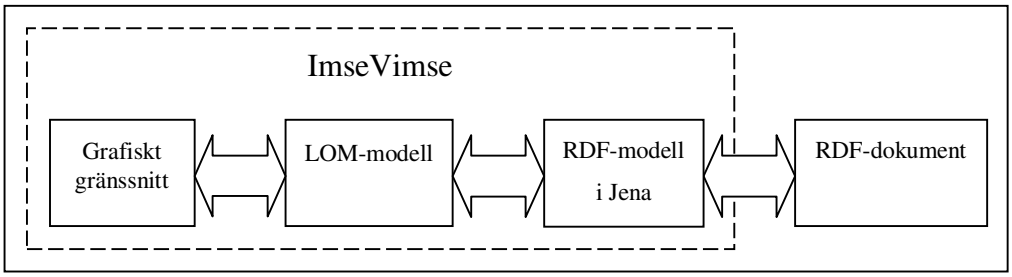
Figur 10. Det grafiska gränssnittet för ImseVimse.

När ImseVimse utvecklades fanns det planer på att det ska kunna stödja flera olika LOM-bindningar. Därför är den interna LOM-modellen och de komponenter den använder separerad från de olika LOM-bindningarna. Den interna LOM-modellen är uppbyggd i en Java-klass med en variabel, med tillhörande modifierare, för varje LOM-element. Dessa variabler består i sin tur av olika klasser, uppbyggda efter hur LOM-elementet de ska representera ser ut. För varje enskild LOM-instans i ImseVimse, skapas det en instans av denna Java-klass. När användaren gör förändringar på något LOM-element i gränssnittet, görs motsvarande förändringar i objektet för den LOM-instansen.

Jena

Jena är ett Java API för att hantera RDF-modeller. Jena har en inbyggd RDF/XML-parser, vilken jag använder under inläsningen av ett RDF-dokument. Parsern returnerar en RDF-modell. Denna RDF-modell stegas igenom och data extraheras ur RDF-konstellationerna för de olika LOM-elementen. Detta görs med hjälp av den funktionalitet Jenas API erbjuder. Jena erbjuder även funktioner för att bygga upp en RDF-modell, vilket kommer väl till pass när användaren vill spara till ett RDF-dokument.

Figur 11 visar det abstrakta flödet av data vid inläsning och skrivning av en LOM-instans.



Figur 11. Det abstrakta dataflödet i ImseVimse för LOM/RDF-bindningen.

Inläsning av RDF-dokument

Det första som sker när ett RDF-dokument öppnas är att Jenas inbyggda parser gör en syntaxanalys av dokumentet och bygger upp en RDF-modell av innehållet. Eftersom ett RDF-dokument kan innehålla metadata för flera olika lärobjekt får användaren därefter välja den rotresurs som motsvarar den LOM-instans som skall läsas in. Sedan letar programmet igenom den valda LOM-instansen efter RDF-

konstellationer som matchar dem som är definierade i LOM/RDF-bindningen, och bygger upp LOM-modellen utefter vad den finner. Avslutningsvis fylls det grafiska gränssnittet med de matchade LOM-elementen.

Utskrift till RDF-dokument

När användaren väljer att spara, överförs innehållet i det grafiska gränssnittet till LOM-modellen. Därefter stegar programmet igenom LOM-modellen och bygger upp en RDF-modell enligt LOM/RDF-bindningens definition för de olika LOM-elementen. Avslutningsvis skrivs RDF-modellen till ett RDF-dokument med hjälp av en utskriftsfunktion i Jena.

Verifiering av implementeringen

Jag måste verifiera att implementeringen är korrekt för både inläsning och utskrift av RDF-dokument.

Inläsning

För att verifiera att inläsningen fungerar har jag använt mig av tre RDF-dokument. Det första RDF-dokumentet innehåller en komplett LOM-instans och har använts för att verifiera att det går att läsa in alla typer av LOM-element. Detta dokument har jag även fått modifiera vid verifieringen, eftersom vissa LOM-element kan vara modellerade på olika vis enligt LOM/RDF-bindningen. Det andra RDF-dokumentet innehåller även det en LOM-instans, och har använts för ett relaterat lärobjekt, för att verifiera LOM-kategorin *Relation*. Det tredje RDF-dokumentet innehåller ett klassificeringssystem, och har använts för att verifiera LOM-kategorin *Classification*.

Utskrift

För att verifiera att utskriften av alla LOM-element fungerar, har jag använt mig av det utskrivna RDF-dokumentet vid inläsning, och på så vis kontrollerat att det blir korrekt uppbyggt.

Problem

Här presenterar jag de problem jag har stött på under implementeringsarbetet. Dels de problem jag har upptäckt med LOM/RDF-bindningen och dels de problem som uppstått i ImseVimse.

Generella problem

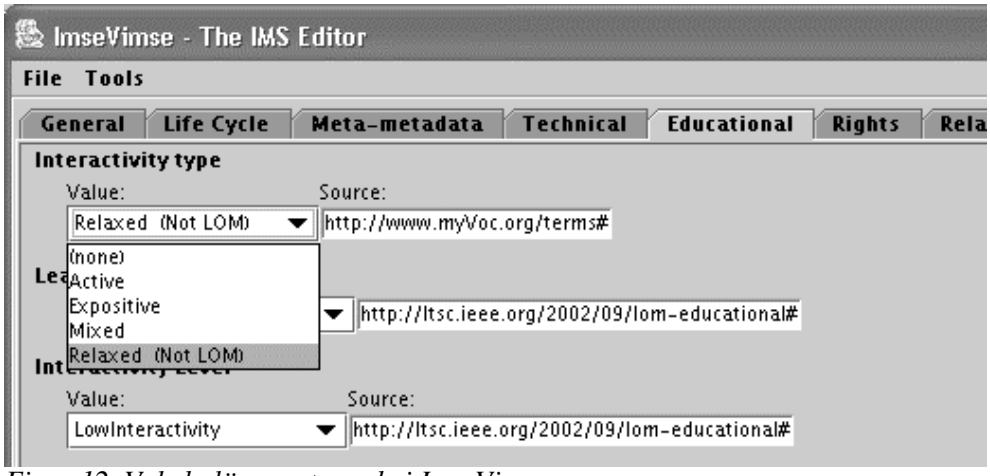
- När man öppnar ett RDF-dokument och ska välja rotresurs för en viss LOM-instans kan namnet på rotresursen, vilket utgörs av rotresursens URI, se underligt ut. Det finns två anledningar till att det kan bli så. Det kan till exempel uppkomma när användaren har skapat en ny LOM-instans och sparat den i RDF-format. Det som då sker, är att när Jena skapar rotresursen så automat-genererar den en unik URI åt rotresursen. Dessa automatgenererade URI:er är inte användarvänliga i sitt utseende. Problemet kan även uppstå om rotresursen är en så kallad anonym nod. En anonym nod är en resurs som i LOM-modellen saknar en URI. När Jena läser in en anonym nod så automatgenererar den en URI åt den anonyma noden för att kunna hantera den unikt. Därav det underliga namnet på rotresursen.

- Om man öppnar en LOM-instans från ett RDF-dokument innehållande rotresurser för flera olika LOM-instanser, och därefter sparar LOM-instansen till samma RDF-dokument, kommer man att förlora de övriga LOM-instanserna i RDF-dokumentet. Anledningen är att ImseVimse endast känner till den öppnade LOM-instansen. När Jenas utskriftsfunktion skriver RDF-modellen till RDF-dokumentet, kommer den aktuella RDF-modellen att vara det enda RDF-dokumentet innehåller.
- LOM/RDF-bindningen beskriver hur de olika RDF-konstellationerna får se ut i form av vilka egenskapstyper som ska vara knutna till vilka resurser. I flera fall är det dock tillåtet att använda sig av egenskapstyper man själv har skapat, istället för den föreslagna egenskapstypen, så länge den egna egenskapstypen ärver egenskaperna från den föreslagna egenskapstypen. Det är även tillåtet att i flera fall använda sig av en resurs man själv har skapat, istället för den föreslagna resursen, så länge den egna resursen ärver egenskaperna från den föreslagna resursen. Dessa båda arv är transitiva och måste definieras separat. Ett problem med detta är att när ImseVimse läser in en RDF-modell som använder sig av egendefinierade egenskapstyper eller resurser måste ImseVimse även ha tillgång till de RDF-dokument som beskriver dessa arv för att kunna verifiera arven. Ett annat problem är hur användaren av ImseVimse ska kunna använda sina egenutvecklade egenskapstyper och resurser under modifieringen av en LOM-instans. Då arbetet för att lösa dessa problem har ansetts för omfattande har det strukits på grund av tidsbrist.

Vokabulärer

Datatypen vokabulär som är definierad i LOM-specifikationen har tidigare inte varit implementerad i ImseVimse. ImseVimse saknade därför stöd och funktionalitet för hur vokabulärer ska hanteras. ImseVimse skulle behöva en modul för att hantera vokabulärer för de olika LOM-elementen. En modul som låter användaren lägga till, ta bort och ändra vilka värden som ska vara fördefinierade för en vokabulär. Den ska även hantera verifiering av de olika värdena för vokabulären, så att det tydligt framgår för användaren om det valda värdet är ett av de fördefinierade värdena enligt LOM-specifikationen för ett visst LOM-element. Arbetet med att utveckla en sådan modul skulle bli för tidskrävande och jag har därför inte implementerat den.

Vad jag däremot har gjort är att jag har skapat de grafiska komponenter som behövs för att representera en vokabulär. Jag har även skapat en kontroll som vid inläsning av värde och källa för en vokabulär kontrollerar om de överensstämmer med de fördefinierade i LOM-specifikationen. Som framgår av figur 12 nedan står det ”Not LOM” bredvid värdet i vokabulären om det inte överensstämmer med LOM-specifikationen. För att det ska vara så smidigt som möjligt för användaren att välja ett värde för en vokabulär så är värdena inlästa i en drop down-lista. Denna lista består av de värden som ingår i LOM-specifikationen för det aktuella LOM-elementet samt det inlästa värdet om det inte överensstämmer med LOM-specifikationen. Figur 12 visar hur en vokabulär ser ut, i detta fall är det LOM-element 5.1 Interactivity Type. När användaren växlar mellan värdena kommer källan att växla automatiskt.



Figur 12. Vokabulärens utseende i ImseVimse.

Denna enklare lösning för vokabulärer är inte helt problemfri. Det går bra att läsa in en vokabulär som inte överensstämmer med LOM-specifikationen. Det går även bra att skriva denna vokabulär till ett RDF-dokument när man sparar. Men det är inte möjligt för användaren att skapa ett nytt värde och en ny källa för LOM-elementet i ImseVimse.

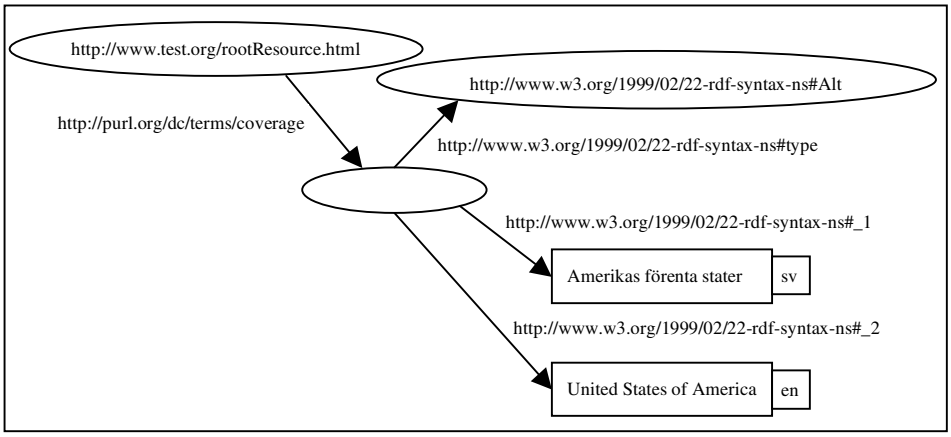
Ett annat problem som inte är löst är när ett värde för en vokabulär i LOM-specifikationen är modellerad i LOM/RDF-bindningen med en URI vars *local name* inte är det samma som värdet i LOM-specifikationen. Detta inträffar för LOM-elementet *Contribute* under LOM-kategorin Life Cycle, där värdet för vokabulären Role kan vara ”author”. I LOM/RDF-bindningen är ”author” modellerat med `http://purl.org/dc/elements/1.1/creator`. Anledningen till att detta är ett problem, är att om man läser in ovan nämnda URI kommer värdet för vokabulären i det grafiska gränssnittet att vara ”creator”. Då ställs helt plötsligt kravet att användaren måste känna till hur vokabulärerna i LOM-specifikationen är modellerade i RDF, vilket är ett orimligt krav. Av de 108 fördefinierade värdena för de olika vokabulärerna i LOM-specifikationen inträffar detta problem tre gånger. Förutom det nyss nämnda exemplet påverkas även värdena ”discipline” och ”idea” för vokabulären i LOM-elementet *Purpose* under LOM-kategorin Classification. Detta problem skulle möjligtvis kunna lösas genom att mappa alla vokabulärer definierade i LOM-specifikationen mot motsvarande URI:er i LOM/RDF-bindningen. Mappningsfunktionaliteten bör i så fall hamna inom den modul som behövs för att ImseVimse ska kunna stödja vokabulärer fullt ut.

Specifika problem för olika LOM-element

Nedanstående problem är ordnade efter de nummer respektive LOM-element har i LOM-specifikationen.

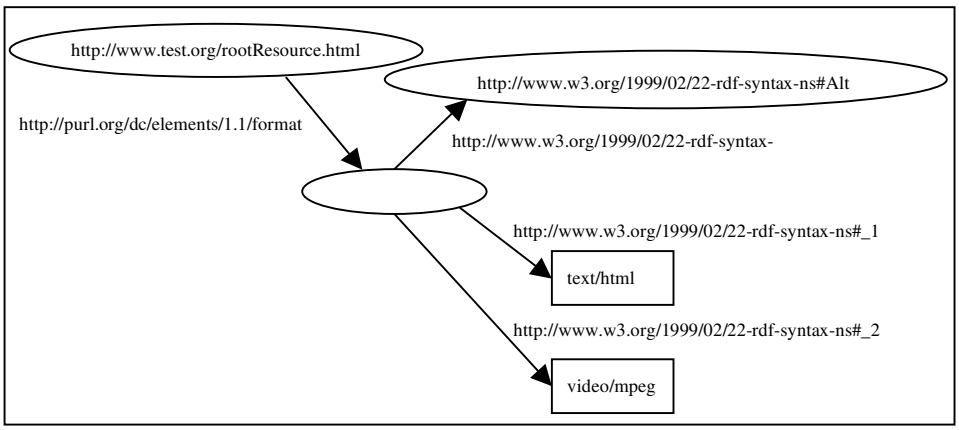
- LOM-element 1.6 *Coverage* under LOM-kategorin General anger vilken tid, kultur eller geografisk plats lärobjektet behandlar. Coverage är modellerat genom att låta egenskapstyperna `dc:coverage`, `dcterms:spatial` och `dcterms:temporal` peka på en resurs av en viss typ eller på en text-beskrivning. Coverage består i ImseVimse av LangStringar. Problemet uppstår när dessa LangStringar vid skrivning skall översättas till RDF-modellen. Eftersom man inte vet vad de innehåller måste de få en generell uppbyggnad i RDF-modellen. Vid inläsning följt av skrivning kan detta resultera i förlust av semantik för RDF-konstellationen som representerar Coverage. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att vid skrivning ska RDF-modellen använda sig av egenskapstypen `dc:coverage` pekandes på en

textbeskrivning innehållande LangStringarna. Figur 13 nedan, återger hur LOM-element Coverage modelleras vid skrivning.



Figur 13. Modelleringen av LOM-element 1.6 Coverage vid skrivning.

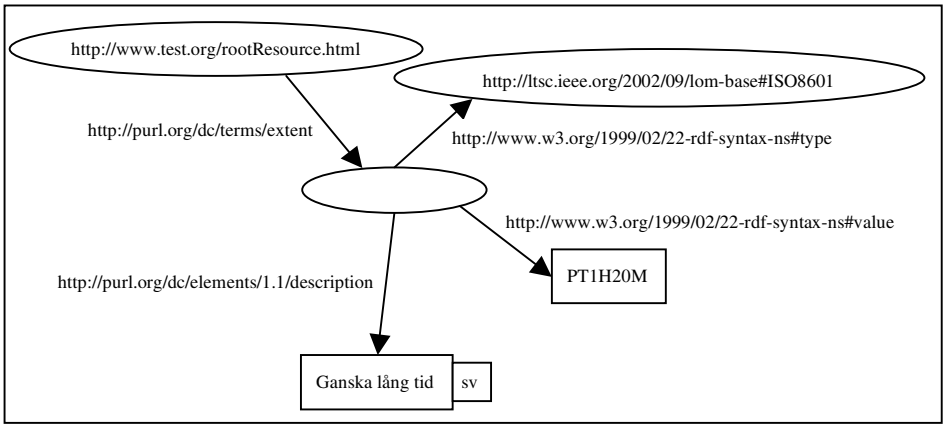
- LOM-element 4.1 *Format* under LOM-kategorin Technical anger vilket dataformat lärobjektet har. Dataformatet är antingen av MIME-typ eller icke-digital. Format är modellerat genom att låta egenskapstypen `dc:format` peka på en resurs av typen `dcterms:IMT` om dataformatet är en MIME-typ eller peka på en textbeskrivning om det är icke-digital. Format består i ImseVimse av CharacterStringar. Eftersom man inte vet vad de innehåller måste de få en generell uppbyggnad i RDF-modellen. Vid inläsning följt av skrivning resulterar detta i förlust av semantik för RDF-konstellationen som representerar en MIME-typ. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att vid skrivning ska RDF-modellen använda sig av egenskapstypen `dc:format` pekandes på en textbeskrivning innehållande CharacterStringarna. Figur 14 nedan återger hur LOM-element Format modelleras vid skrivning.



Figur 14. Modelleringen av LOM-element 4.1 Format vid skrivning.

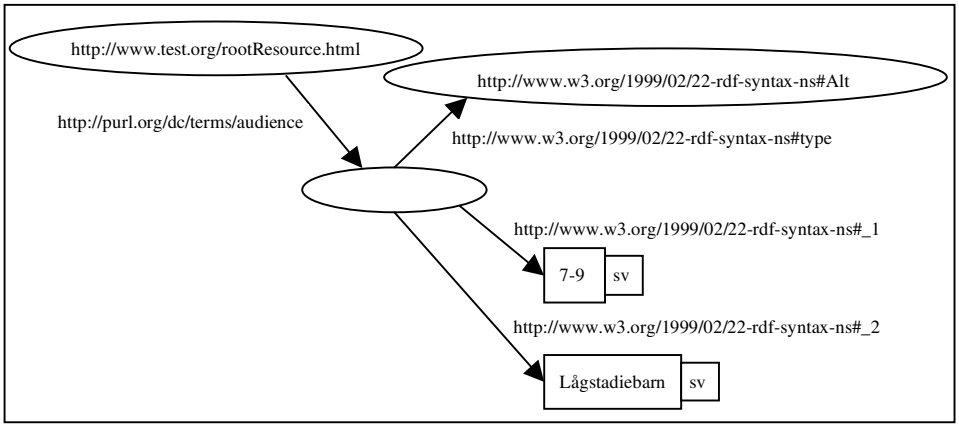
- LOM-element 4.7 *Duration* under LOM-kategorin Technical anger den tid det tar att spela upp lärobjektet. Duration är främst tänkt att användas för ljud, filmer eller animationer. Duration är modellerat genom att låta egenskapstypen `dcterms:extent` peka på en resurs av typen `lom:ISO8601` som i sin tur pekar på ett värde som anger tiden det tar att spela upp lärobjektet. Enligt LOM-standarden ska LOM-elementet Duration bestå av datatypen Duration. Datatypen Duration innehåller förutom en CharacterString som anger tiden även en LangString som beskriver tiden. Denna beskrivning är inte modellerad i LOM/RDF-bindningen. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att resursen av typen `lom:ISO8601` via egenskapstypen `dc:description` ska peka på en textbeskrivning som beskriver tiden det

tar att spela upp lärobjektet. Figur 15 nedan återger LOM-element Durations nya modellering.



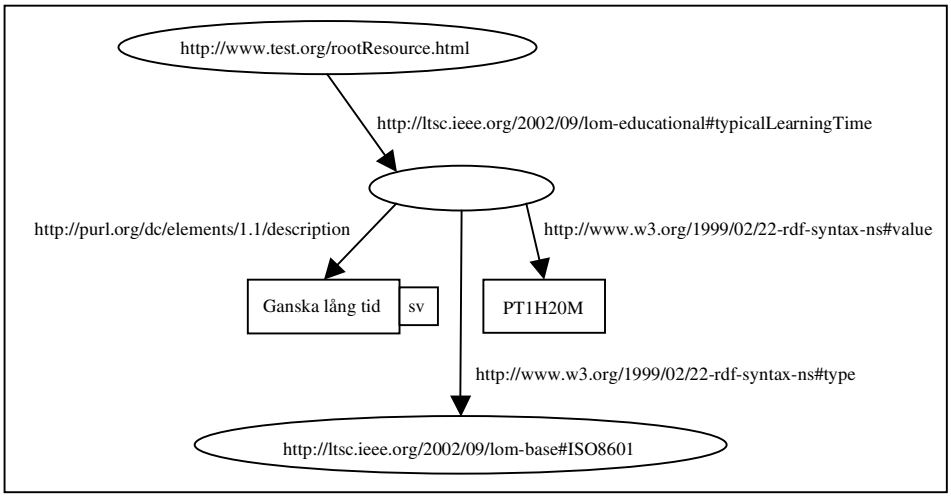
Figur 15. Den nya modelleringen av LOM-element 4.7 Duration.

- LOM-element 5.2 *Learning Resource Type* under LOM-kategorin Educational anger vilken specifik typ av lärobjektet det är, till exempel bildspel eller föreläsning. Learning Resource Type är en vokabulär som är modellerad genom att låta egenskapstypen `rdf:type` peka på en resurs vars URI är värdet på vokabulären. Enligt LOM-specifikationen får det förekomma flera olika värden på Learning Resource Type. Om det förekommer flera olika värden skall dessa vara ordnade efter hur väl typerna överensstämmer med lärobjektet. Den typ som bäst överensstämmer med lärobjektet skall vara först. Problemet med detta är att en ordnad lista i RDF modelleras med hjälp av egenskapstypen `rdf:type` pekandes på RDF:s SEQ-resurs, vilket är en konstruktion man använder sig av när man vill gruppera flera resurser och ordningen är av betydelse. Detta går dock inte eftersom egenskapstypen `rdf:type` redan är upptagen för att peka ut en ensam Learning Resource Type-resurs.
- LOM-element 5.5 *Intended End User Role* under LOM-kategorin Educational anger vilken eller vilka målgrupper lärobjektet är skapat för, till exempel lärare eller chefer. Enligt LOM-specifikationen spelar det ingen roll i vilken ordning man anger dessa målgrupper, om det är flera. Enligt LOM/RDF-bindningen skall flera målgrupper vara ordnade med RDF:s SEQ-resurs, vilket är en konstruktion man använder sig av när man vill gruppera flera resurser och ordningen är av betydelse. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att kravet på att ordna flera målgrupper med RDF:s SEQ-resurs stryks.
- LOM-element 5.7 *Typical Age Range* under LOM-kategorin Educational anger åldern för den typiske användaren av lärobjektet. Typical Age Range är modellerat genom att låta egenskapstypen `dcterms:audience` peka på en resurs av typen `lom-edu:AgeRange` eller på en textbeskrivning. Typical Age Range består i ImseVimse av LangStringar. Problemet uppstår när dessa LangStringar vid skrivning skall översättas till RDF-modellen. Eftersom man inte vet vad de innehåller måste de få en generell uppbyggnad i RDF-modellen. Vid inläsning följt av skrivning kan detta resultera i förlust av semantik för RDF-konstellationen som representerar Typical Age Range. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att vid skrivning ska RDF-modellen använda sig av egenskapstypen `dcterms:audience` pekandes på en textbeskrivning innehållande LangStringarna. Figur 16 nedan återger hur LOM-element Typical Age Range modelleras vid skrivning.



Figur 16. Modelleringen av LOM-element 5.7 Typical Age Range vid skrivning.

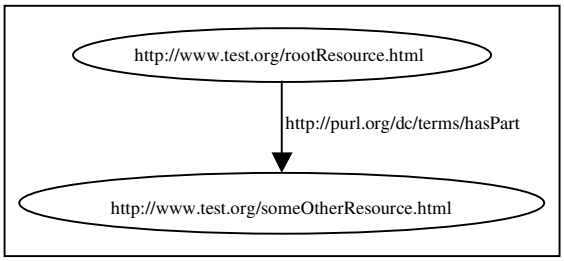
- LOM-element 5.9 *Typical Learning Time* under LOM-kategorin Educational anger den tid det tar att använda eller arbeta sig igenom lärobjektet för den tilltänkte användaren. Typical Learning Time är modellerat genom att låta egenskapstypen `lom-edu:typicalLearningTime` peka på en resurs av typen `lom:ISO8601` som i sin tur pekar på ett värde som anger tiden det tar att använda lärobjektet. Enligt LOM-standarden ska LOM-elementet Typical Learning Time bestå av datatypen Duration. Datatypen Duration innehåller förutom en CharacterString, som anger tiden, även en LangString som beskriver tiden. Denna beskrivning är inte modellerad i LOM/RDF-bindningen. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att resursen av typen `lom:ISO8601` via egenskapstypen `dc:description` ska peka på en textbeskrivning som beskriver tiden det tar att använda lärobjektet. Figur 17 nedan återger LOM-element Typical Learning Times nya modellering.



Figur 17. Den nya modelleringen av LOM-element 5.9 Typical Learning Time.

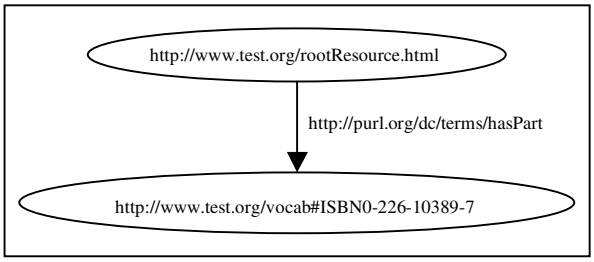
- LOM-element 5.10 *Description* under LOM-kategorin Educational anger hur lärobjektet är tänkt att användas. Enligt LOM-specifikationen är det ok att ha flera olika beskrivningar om hur lärobjektet är tänkt att användas. Enligt LOM/RDF-bindningen är det inte tillåtet att ha flera olika beskrivningar. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att kravet på att det inte får förekomma flera beskrivningar ströks.
- LOM-kategorin *Relations* definierar eventuella relationer det här lärobjektet har mot andra lärobjekt. En relation består av tre delar. De tre delarna är, vilken *typ* av relation det är, en *identifierare* för det relaterade lärobjektet och en

beskrivning av det relaterade lärobjektet. I LOM/RDF-bindningen är relationer modellerade genom en egenskapstyp som beskriver vilken typ av relation det är frågan om. Denna egenskapstyp pekar på en resurs vars URI är URI:n för det relaterade lärobjektet. När användaren öppnar en LOM-instans innehållande relationer kommer användaren via ett dialogfönster få ange i vilket RDF-dokument det relaterade lärobjektets LOM-instans finns. Denna fråga använder sig av URI:n för det relaterade lärobjektet. När användaren har angett RDF-dokument och även fått välja LOM-instans i RDF-dokumentet, extraheras det relaterade lärobjektets identifierare och beskrivning ur dess LOM-instans. När användaren väljer att spara en LOM-instans innehållande relationer modelleras en relation genom en egenskapstyp som anger vilken typ av relation det är fråga om. Denna egenskapstyp pekar på en resurs vars URI är identifieraren för det relaterade lärobjektet. Problemet med detta förfarande är att om identifieraren inte är av typen URI kommer URI:n för resursen, som representerar det relaterade lärobjektet, att vara felaktig. Detta kan även leda till missförstånd när användaren ska ange i vilket RDF-dokument det relaterade lärobjektets LOM-instans finns. I figur 18 nedan, är URI:n för resursen, som anger vilket det relaterade lärobjektet är, korrekt.



Figur 18. Korrekt utseende på en URI för ett relaterat lärobjekt.

I figur 19 nedan, är URI:n för resursen, som anger vilket det relaterade lärobjektet är, felaktig. Anledningen till det är att den del av identifieraren som säger vad det är för typ av identifierare, i det här fallet `http://www.test.org/vocab#ISBN`, har slagits ihop med den del av identifieraren som utgör värdet för identifieraren, i det här fallet `0-226-10389-7`, på ett sätt som det inte är tänkt att användas. Denna sammanslagning sker eftersom det inte finns ett bestämt sätt att hantera detta fall på.

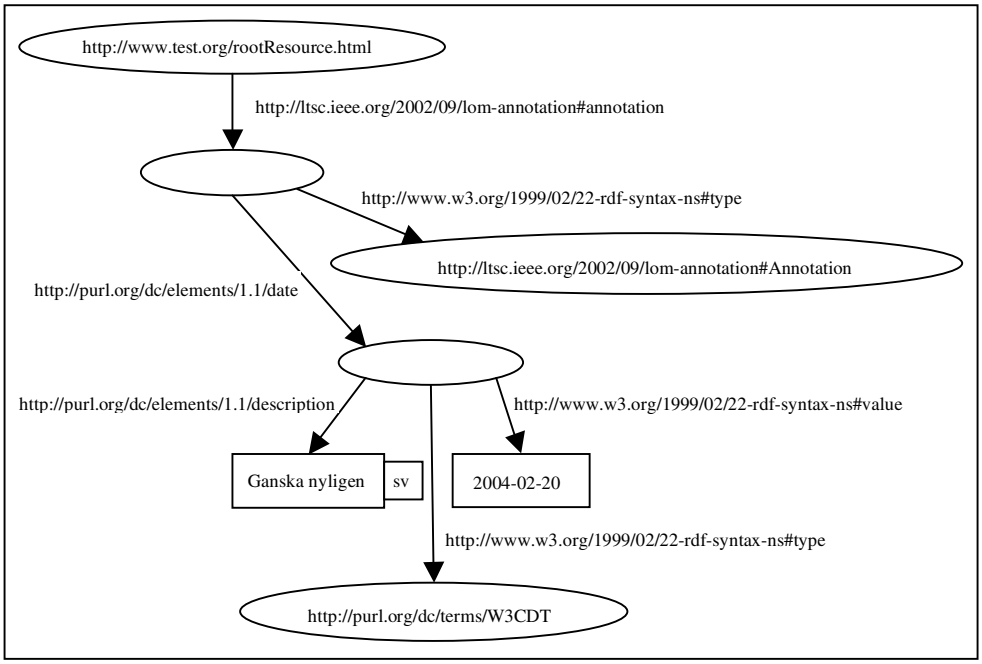


Figur 19. Felaktigt utseende på en URI för ett relaterat lärobjekt.

- LOM-element 7.1 *Kind* under LOM-kategorin Relations anger vilken typ av relation det här lärobjektet har mot ett annat lärobjekt. I LOM/RDF-bindningen finns modelleringsalternativ för två typer av relationer som inte är definierade i LOM-specifikationen. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att dessa stryks.

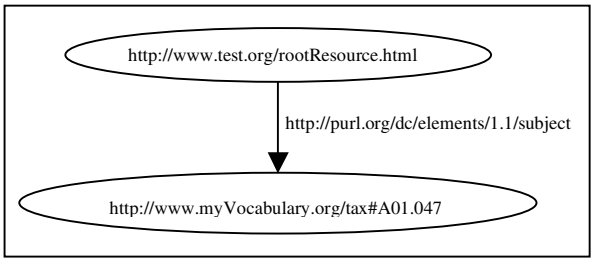
- LOM-element 8.2 *Date* under LOM-kategorin Annotation anger datumet då en kommentar om lärobjektet skapades. Date är modellerad genom att låta egenskapstypen `lom-ann:annotation` peka på en resurs av typen `lom-ann:Annotation` som i sin tur via egenskapstypen `dc:date` pekar på en resurs av typen `dcterms:W3CDTF`. Enligt LOM-standarden ska LOM-

elementet Date bestå av datatypen DateTime. Datatypen DateTime innehåller förutom en CharacterString som anger tiden även en LangString som beskriver tiden. Denna beskrivning är inte modellerad i LOM/RDF-bindningen. Efter att ha diskuterat detta med Mikael Nilsson föreslog han att resursen av typen `dcterm:W3CDTF` via egenskapstypen `dc:description` ska peka på en textbeskrivning som beskriver datumet då kommentaren om lärobjektet skapades. Figur 20 nedan återger LOM-element Dates nya modellering. Figuren modellerar bara den del av LOM-kategorin Annotation som berör datumet.



Figur 20. Den nya modelleringen av LOM-element 8.2 Date.

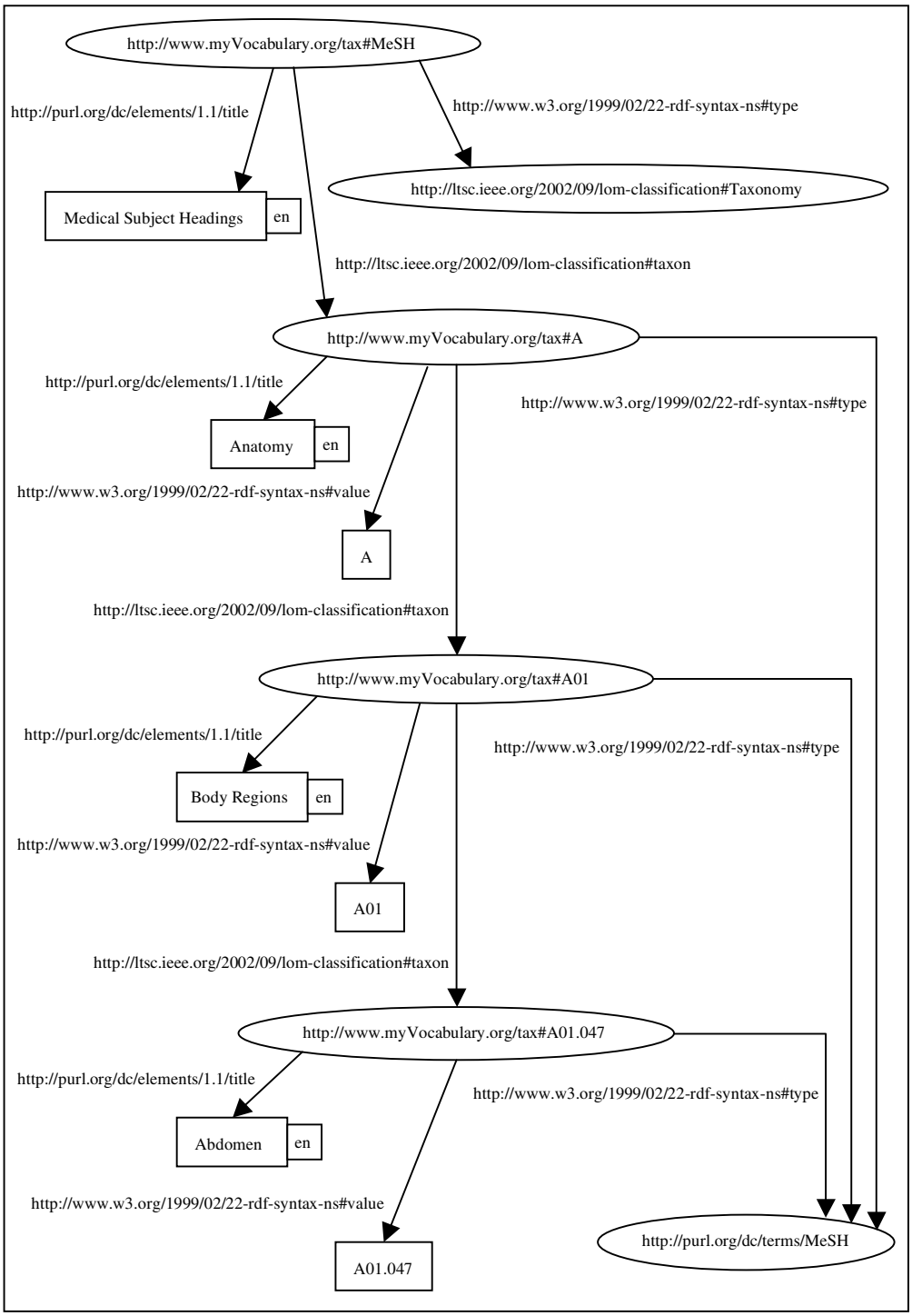
- LOM-kategorin *Classification* anger var lärobjektet hamnar inom ett visst klassificeringssystem. I LOM/RDF-bindningen är en klassificering modellerad genom en egenskapstyp, som beskriver *syftet* med att lärobjektet är klassificerat, som pekar på en resurs vars URI är *identifieraren* för en viss nivå inom ett klassificeringssystem. Figur 21 visar denna modellering.



Figur 21. Modellering av klassificering av ett lärobjekt.

När användaren öppnar en LOM-instans innehållande klassificeringar kommer användaren via ett dialogfönster få ange i vilket RDF-dokument klassificeringssystemet för identifieraren finns. Ett RDF-dokument kan innehålla flera klassificeringssystem. Programmet letar igenom RDF-dokumentet efter de resurser som utgör basen för varje klassificeringssystem. Därefter stegas klassificeringssystemen igenom rekursivt från basen tills man lyckas matcha den givna identifieraren med en viss nivå inom ett klassificeringssystem. Ett klassificeringssystem är indelat i nivåer. Varje nivå utgörs av en resurs vars URI är den nivåns identifierare. Nivåresurserna pekar i sin tur på en resurs vars URI anger vad det är för typ av klassificeringssystem. Denna typ motsvarar LOM-elementet 9.2.1 Source i LOM-specifikationen.

Varje nivåresurs pekar på en Literal innehållande den delen av nivåresursens URI som utgör *local name*. Varje nivåresurs kan peka på en textbeskrivning som förklarar vad nivån representerar. Avslutningsvis kan nivåresursen via egenskapstypen `lom-cls:taxon` peka på nästa nivåresurs i klassificeringssystemet om det finns en sådan. Figur 22 nedan visar denna modellering av ett klassificeringssystem.



Figur 22. Modellering av ett klassificeringssystem.

Det finns några begränsningar i hur LOM/RDF-bindningen modellerar ett klassificeringssystem jämfört med hur LOM-specifikationen definierar det. För det första är det inte beskrivet i LOM/RDF-bindningen hur LOM-elementen 9.3 Description och 9.4 Keyword ska modelleras. Och sedan kan LOM-

element 9.2.1 Source vilket är en LangString endast ha ett värde eftersom det är modellerat genom en URI.

Resultat

I det här kapitlet analyserar jag de olika problemen jag stött på och kategoriserar dem efter vad det är för typ av problem.

Problemanalys

Jag har delat in de olika problemen jag presenterat i föregående kapitel i olika kategorier, för att ge en tydligare bild av deras omfattning och vad de innebär.

Modelleringsproblem

Problemen inom den här kategorin beror på hur LOM/RDF-bindningen modellerar de olika LOM-elementen. Flera av problemen är av enklare art men några är lite mer komplicerade och kan innebära lite huvudbry för arbetsgruppen som tar fram LOM/RDF-bindningen.

Enkla problem

De enklare problemen är de problem som jag efter att ha diskuterat med Mikael Nilsson, har fått ett förslag på lösning på och har kunnat åtgärda direkt. Lösningsförslagen har antingen bestått av hur jag ska implementera LOM/RDF-bindningen eller inneburit en förändring i LOM/RDF-bindningen. De problem som faller inom denna kategori är:

- Problemet med hur skrivningen av LOM-element 1.6 Coverage ska gå till.
- Problemet med förlust av semantik för MIME-typer vid skrivning av LOM-element 4.1 Format.
- Problemet med avsaknad av modellering av beskrivningen av tiden för LOM-element 4.7 Duration.
- Problemet med modelleringen av ordning av målgrupper för LOM-element 5.5 Intended End User Role.
- Problemet med förlust av semantik för åldersgrupperingar för LOM-element 5.7 Typical Age Range.
- Problemet med avsaknad av modellering av beskrivningen av tiden för LOM-element 5.9 Typical Learning Time.
- Problemet med flera beskrivningar av hur lärobjektet ska användas för LOM-element 5.10 Description.
- Problemet med förekomst av relationstyper som inte ingår i LOM-specifikationen för LOM-element 7.1 Kind.
- Problemet med avsaknad av modellering av beskrivningen av datumet för LOM-element 8.2 Date.

Kvarstående problem

För dessa problem finns det ingen färdig lösning. Om och hur dessa problem skall hanteras är upp till arbetsgruppen som utvecklar LOM/RDF-bindningen att avgöra.

- Problemet med ordningen av flera typer för LOM-element 5.2 Learning Resource Type.

- LOM-kategori Relations:
  Problemet med att spara identifieraren som en URI för ett relaterat lärobjekt. Anledningen till att detta problem uppstår är en kombination av hur LOM/RDF-bindningen är modellerad och hur den är tänkt att implementeras. Det vill säga det sätt identifieraren sparas och hur identifieraren används när användaren ska säga var RDF-dokumentet för det relaterade lärobjektet finns.
- LOM-kategorin Classification:
  Problemen med LOM-element 9.3 Description och LOM-element 9.4 Keyword är att de inte är modellerade i LOM/RDF-bindningen. Jag antar att det inte ska vara något större problem att bestämma hur dessa modelleringar ska se ut när man väl åtgärdar det.
  Enligt LOM-specifikationen är LOM-element 9.2.1 Source en LangString. Men genom den valda modelleringen kan den inte anta olika översättningar utan endast ha ett värde.

Problem som är beroende av ny funktionalitet

Dessa problem kräver ny funktionalitet i ImseVimse för att lösas. Problemen i sig är inte helt olika. Lösningen för det ena kan säkert ge uppslag för hur det andra ska lösas.

- Problemet med att kunna använda sig av egenutvecklade egenskapstyper och resurser. Vid inläsning krävs det funktionalitet för att hantera de filer i vilka definitionerna för de egenutvecklade egenskapstyperna och resurserna finns. Och vid modifiering av en LOM-instans i ImseVimse krävs det funktionalitet för att kunna använda sig av de egenutvecklade egenskapstyperna och resurserna.
- Problemet med stöd för vokabulärer. ImseVimse skulle behöva en modul för att hantera vokabulärer för de olika LOM-elementen så användaren kan använda sig av andra vokabulärer än LOM:s fördefinierade. Detta problem är inte ett specifikt problem för LOM/RDF-bindningen utan generellt för ImseVimse. Däremot skulle det vara bra om denna modul innehöll en mappningsfunktion för att mappa namnen på vokabulären mot URI:n för vokabulären i LOM/RDF-bindningen eftersom dessa inte alltid är samma.

Problem med LOM-instanserna

Dessa problem uppstår vid hanteringen av LOM-instanser.

- Problemet med underligt namn på rotresursen vid öppnandet av en LOM-instans. Jag har inget förslag på vad man ska göra för att lösa detta problem.
- Problemet med förlust av LOM-instanser vid öppnande och sparande till samma RDF-dokument. En lösning skulle kunna vara att ImseVimse läser in hela RDF-dokumentet. När man väl sparar måste sedan rätt LOM-instans i RDF-dokumentet tas bort och ersättas med den modifierade. Frågan är i så fall vad som ska ske om användaren skapar en helt ny LOM-instans i ImseVimse och vill spara den till ett befintligt RDF-dokument. Detta förfarande brukar i vanliga fall resultera i en fråga om man vill ersätta dokumentet. Jag har inget bra förslag för hur detta problem ska lösas som inte resulterar i att användaren blir förvirrad.

Slutsats

Här följer en summering av mitt arbete och en kort beskrivning av vad jag har kommit fram till.

Målet med examensarbetet var att finna vilka problem som uppstår när ImseVimse utökas med LOM/RDF-bindningen. För att finna dessa problem valde jag att försöka implementera LOM/RDF-bindningen i ImseVimse. Innan jag kunde påbörja det arbetet, var jag först tvungen att uppdatera ImseVimse så att det följde LOMv1.0. Denna uppdatering innebar förändringar i både det grafiska gränssnittet och den interna LOM-modellen i ImseVimse. Själva implementeringen av LOM/RDF-bindningen för de olika LOM-elementen gick smidigare än jag trodde innan jag påbörjade arbetet. ImseVimse kan nu läsa och skriva även till RDF-formatet.

De problem jag har funnit, har jag analyserat och delat in i de tre kategorierna:

- *Modelleringsproblem*: Dessa problem beror på hur de olika LOM-elementen är modellerade i LOM/RDF-bindningen. Några av problemen är till exempel att ett LOM-element saknar fullständig modellering, att RDF-modellen förlorar semantik för ett LOM-element vid skrivning till ett RDF-dokument, eller att modelleringen av ett LOM-element står i konflikt med möjligheten att uppfylla LOM-standarden.
- *Problem som är beroende av ny funktionalitet*: Den funktionalitet dessa problem är beroende av, är möjligheten för användaren att använda sig av egenutvecklade egenskapstyper och resurser, samt möjligheten för användaren att använda sig av egna vokabulärer.
- *Problem med LOM-instanserna*: Dessa problem uppkommer när användaren ska spara eller öppna en LOM-instans. Ett av problemen uppkommer i form av svårtydda namn på rotresurserna när användaren öppnar en LOM-instans. Ett annat problem är att det inte är uppenbart att endast den aktuella LOM-instansen skrivs till RDF-dokumentet när användaren sparar. Problemen kommer troligtvis att skapa förvirring hos användaren.

Har jag då funnit alla problem som uppstår när ImseVimse utökas med LOM/RDF-bindningen?

Svaret på den frågan är antagligen nej. Eftersom några problem är beroende av ny funktionalitet i ImseVimse, kommer det troligtvis att uppstå nya problem när den nya funktionaliteten implementeras.

För närvarande ligger LOM/RDF-bindningen för omröstning för standardisering, hos IEEE. Hur det arbetet fortskrider kommer naturligtvis att påverka behovet av nya uppdateringar för ImseVimse.

Litteraturlista

Här följer min refererade litteratur.

[1] BRASE, J., NILSSON, M., PALMÉR, M. (2003) *The LOM RDF Binding – Principles and Implementation*
http://rubens.cs.kuleuven.ac.be:8989/ariadne/CONF2003/papers/MIK2003.pdf
Verifierad mars 2004.

[2] DECKER, S., NAEVE, A., NEJDL, W., NILSSON, M., PALMÉR, M., QU, C., RISCH, T., SINTEK, M., WOLF, B. (2002) *Edutella: A P2P Networking Infrastructure Based on RDF*
http://edutella.jxta.org/reports/edutella-whitepaper.pdf
Verifierad mars 2004.

[3] *Draft Standard for Learning Object Metadata* (2002) LTSC VID IEEE.
http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
Verifierad mars 2004.

[4] *IEEE 1484.12.3/D1 Draft Standard for eXtensible Markup Language (XML) Binding for Learning Object Metadata Data Model* (2003) LTSC VID IEEE.
http://www.cs.kuleuven.ac.be/~erikd/LOM/20030214/ballot.pdf
Verifierad mars 2004.

[5] *IMS Learning Resource Meta-Data Best Practice and Implementation Guide* (2001) IMS.
http://www.imsglobal.org/metadata/imsmdv1p2p1/imsmd_bestv1p2p1.html
Verifierad mars 2004.

[6] MANDOLA, F., MILLER, E. (2004) *RDF Primer*
http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
Verifierad mars 2004.

[7] NILSSON, M. (2003) *The Edutella P2P Network – Supporting Democratic E-learning and Communities of Practice*
http://kmr.nada.kth.se/papers/SemanticWeb/Edutella-chapter.pdf
Verifierad mars 2004.

[8] NILSSON, M. (2001) *The Semantic Web: How RDF will Change Learning Technology Standards*
http://www.cetis.ac.uk/content/20010927172953
Verifierad mars 2004.

[9] NILSSON, M. (2002) *IEEE Learning Object Metadata RDF Binding*
http://kmr.nada.kth.se/el/ims/md-lomrdf.html
Verifierad mars 2004.

[10] NILSSON, M. (2003) *Semantic Issues with the LOM RDF Binding*
http://kmr.nada.kth.se/el/ims/md-lom-semantics.html
Verifierad mars 2004.