Ontologies de dispositius mòbils

Projecte de Fi de Carrera

Joan Cervós Escribà

A la meva dona Zakia
i la meva filla Jana

Índex de continguts

1 Introducció

1.1 Justificació i context

Des dels seus inicis, la web està formada per documents HTML (HyperText Markup Language). Aquests contenen tant el contingut com el format que hom pot visualitzar en un navegador d'Internet. Per a aconseguir separar el contingut de la forma es van crear els fulls d'estil CSS (*Cascade Style Sheets*). D'aquesta manera el document HTML només té el seu el contingut i les marques que en configuren la seva estructura lògica, mentre que tota la informació sobre com ha de ser la presentació es troba en els fulls d'estil.

Avui en dia, la majoria de pàgines web estan formades per documents que no tan sols estan escrits en HTML, sinó que aquests contenen a més scripts de client i de servidor. Els scripts de client són fragments de codi que s'executen en el navegador del client, i que aporten una millor experiència a l'usuari en la seva interacció amb la web, ja que permeten accions com validació de dades o canvis en la visualització en funció de la interacció de l'usuari, i tot això sense necessitat de comunicar-se amb el servidor. Els scripts de servidor s'executen en el servidor i permeten servir pàgines de forma dinàmica, és a dir, pàgines en què el seu contingut varia segons les dades emmagatzemades en el servidor, generalment en una base de dades.

Una nova millora introduïda en les aplicacions web consisteix en permetre que el servidor no sols serveixi documents HTML als clients, sinó que també serveixi dades concretes a petició dels usuaris, de manera asíncrona, és a dir, sense interrompre la interacció de l'usuari amb la web. D'aquesta manera, la comunicació de dades entre client i servidor es redueix i els navegadors poden incorporar les dades rebudes als documents sense necessitat d'obtenir un document de nou i recarregar-lo. Aquestes dades es transfereixen de manera estructurada utilitzant el llenguatge XML (*Xtensible Markup Language*). Aquesta tècnica de desenvolupament web es coneix com a AJAX (*Asynchronous JavaScript And XML).*

Però avui en dia ja no sols els usuaris interactuen amb la web, sinó també moltes aplicacions. En el moment en què la informació disponible en el web ha de ser processada per un programa informàtic, aquesta ha d'estar estructurada en la forma adequada per a aquesta tasca. Per a aconseguir aquesta interoperabilitat entre aplicacions o màquines apareixen els serveis web, que defineixen una sèrie de protocols i missatges basats en XML.

La web semàntica té per objectiu afegir coneixement a la web, és a dir, emmagatzemar la informació de manera que un programa, anomenat raonador, sigui capaç d'inferir informació a partir de la base de coneixement emmagatzemada. Els diferents llenguatges utilitzats per a representar el coneixement en la web semàntica estan basats en XML, i són:

- XML Schema, que defineix l'estructura dels documents XML
- RDF (*Resource Description Framework*), que defineix un model de dades per als recursos i les relacions que hi ha entre ells
- RDF Schema defineix l'estructura dels recursos RDF mitjançant classes i propietats, i relacions i herència entre elles
- OWL (*Ontology Web Language*) afegeix més semàntica a les classes i relacions. Aquest llenguatge està basat en RDF i s'utilitza per a la definició d'ontologies, que és una representació de coneixement d'un domini concret. L'ontologia consta de classes, propietats,

relacions i restriccions, i quan es disposa d'un conjunt d'instàncies d'una ontologia diem que tenim una base de coneixement.

- SWRL (*Semantic Web Rule Language*) afegeix regles a les ontologies, proporcionant així més riquesa semàntica.
- SPARQL (*SPARQL Protocol and RDF Query Language)* és un llenguatge de consultes sobre documents RDF.
- SQWRL (Semantic Query-enhanced Web Rule Language) està basat en SWRL i permet fer consultes en una ontologia.

En aquest projecte farem ús de l'editor d'ontologies Protégé i dels llenguatges OWL i SWRL per a crear una ontologia de dispositius mòbils, la instanciarem per crear la base de coneixements, la testejarem i implementarem un prototipus que en farà ús.

1.2 Objectius

Els objectius que volem assolir amb aquest projecte són els següents:

- Conèixer els conceptes bàsics de la web semàntica.
- Conèixer què és una ontologia.
- Conèixer el llenguatge de definició d'ontologies OWL.
- Conèixer el llenguatge de definició de regles SWRL.
- Conèixer el llenguatge de consulta d'ontologies SPARQL.
- Conèixer l'eina d'edició i gestió d'ontologies Protegé.
- Definir i instanciar una ontologia per a descriure les característiques i funcionalitats dels dispositius mòbils. L'ontologia abastarà el màxim coneixement sobre aquest domini, no obstant, serà prioritari que aquesta satisfaci els requeriments de l'enunciat, és a dir, permetre identificar per a cada dispositiu mòbil:
    1) quins tipus d'informació es poden enviar/rebre (sols text, àudio, video, formats supotats i aconsellats de video...)
    2) disponibilitat de funcionalitats de geoposicionament
    3) disponibilitat de connexió de dades (GPRS, 3G...)
- Desenvolupar una aplicació web que consultant l'ontologia crei una pàgina web amb enllaços als fitxers que poden ser visualitzats en un dispositiu mòbil concret. Els fitxers estaran disponibles a mitjan semestre en un servidor de la UOC.

1.3 Enfocament i mètode seguit

La web semàntica és un camp dins de la informàtica relativament nou, i durant els estudis d'Enginyeria Informàtica no es veu en cap de les assignatures que es cursen. És per això que per a la realització d'aquest projecte caldrà primer de tot assolir uns coneixements bàsics en aquesta àrea.

Així doncs, el segon capítol d'aquesta memòria tractarà els llenguatges i eines que fan falta per a crear una ontologia i una aplicació que en faci ús. No es pretén fer una descripció exhaustiva dels

llenguatges sinó fer-ne una introducció. De totes maneres, es facilitaran les referències a les fonts utilitzades, on es pot ampliar la informació que es desitgi.

Abans de definir una ontologia cal veure si ja hi ha alguna ontologia existent i que puguem aprofitar. Per tant abans de començar l'ontologia des de zero farem una recerca d'ontologies existents sobre dispositius mòbils.

Quan es defineix una ontologia, es fa pensant en l'ús que en farà determinada aplicació, però també pensant en que l'ontologia ha de poder ser reutilitzable. Així doncs, definirem l'ontologia pensant en que haurà de servir per a determinar quins fitxers es poden servir als dispositius mòbils, però al mateix temps mirarem de fer que sigui prou àmplia com per a que pugui ser utilitzada per a altres aplicacions.

Un cop definida i provada l'ontologia ja estarem en condicions de començar a desenvolupar l'aplicació. Realitzarem un estudi de les diferents biblioteques de programació (APIs) que hi ha disponibles per a programar aplicacions semàntiques i n'escollirem la que considerem més adequada.

Per al desenvolupament de l'aplicació realitzarem l'anàlisi i definició de requeriments. Després escollirem la plataforma de desenvolupament i d'execució i desenvoluparem l'aplicació. Finalment farem el test de l'aplicació i elaborarem un manual d'instal·lació i d'ús de la mateixa.

1.4 Planificació

Tasques

Tasca	Descripció	Prec.	Inici	Fi
1	**Estudi de les ontologies**		**3/10/2011**	**23/10/2011**
1.1	Estudi del llenguatge OWL: Lectura de les característiques del llenguatge i realització d'exemples.		2/10/2011	9/10/2011
1.2	Estudi del llenguatge SWRL: Lectura de les característiques del llenguatge i creació de regles en els exemples realitzats en la tasca 1.1.	1.1	10/10/2011	16/10/2011
1.3	Estudi del llenguatge SPARQL: Lectura de les característiques del llenguatge i realització de consultes en els exemples realitzats en la tasca 1.2.	1.2	17/10/2011	23/10/2011
1.4	Aprenentatge de Protegé: Aquesta tasca es realitza al mateix temps que les tasques 1.1, 1.2 i 1.3. A mesura que s'estudien els diferents llenguatges es van realitzant proves amb l'editor d'ontologies.		3/10/2011	23/10/2011
1.5	Finalització del capítol 1: Aquest és el capítol d'introducció, per al qual aprofitarem aquest pla de treball i el completarem durant aquesta primera fase del projecte on adquirirem coneixements que ens donaran un millor visió global del projecte.		3/10/2011	23/10/2011

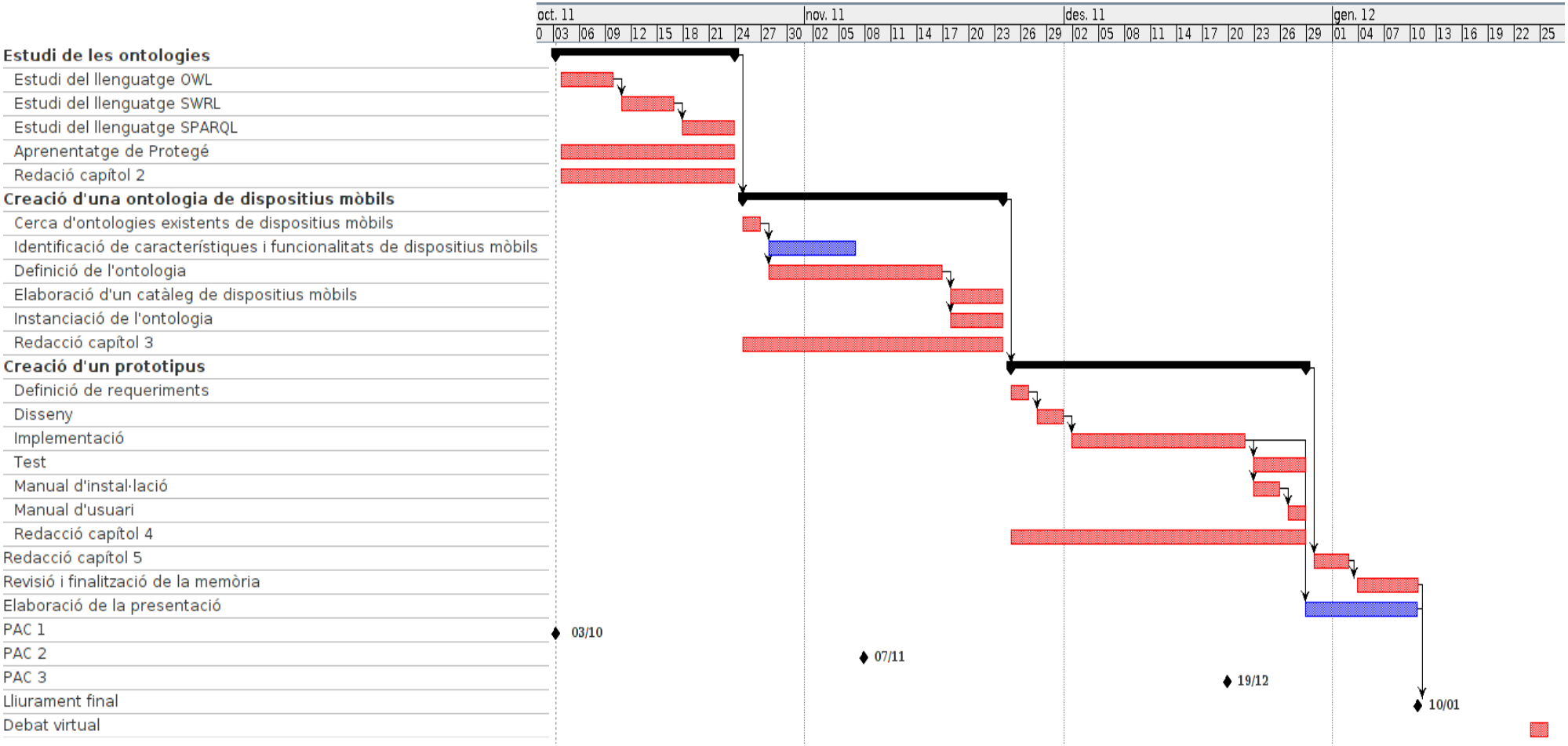
Tasca	Descripció	Prec.	Inici	Fi
1.6	Redacció capítol 2: Durant l'estudi dels diferents llenguatges i eines de les tasques 1.1, 1.2, 1.3 i 1.4, anirem elaborant la memòria corresponent.		3/10/2011	23/10/2011
2	**Creació d'una ontologia de dispositius mòbils**	**1**	**24/10/2011**	**23/11/2011**
2.1	Cerca d'ontologies existents: Es realitzarà una recerca d'ontologies ja existents sobre aquest domini de coneixement, i en cas que n'hi hagi, s'aprofitaran per a la creació de l'ontologia objecte d'aquest projecte.		24/10/2011	26/10/2011
2.2	Identificació de característiques i funcionalitats de dispositius mòbils: Es farà una recerca dels dispositius mòbils disponibles avui en dia en el mercat i quines característiques i funcionalitats ofereixen, i identificarem què és el que ha de representar l'ontologia que definirem.		27/10/2011	6/11/2011
2.3	Definició de l'ontologia: Amb la informació recollida a la tasca 2.2, definirem l'ontologia amb Protegé. Es poden iniciar aquesta tasca i la 2.2 al mateix temps, però acabarem abans la 2.2. per acabar centrant-nos exclusivament en la definició de l'ontologia.		27/10/2011	16/11/2011
2.4.	Elaboració d'un catàleg de dispositius mòbils: Es realitzarà una selecció de dispositius de diferents tipus, marques i prestacions que serviran per a instanciar l'ontologia.	2.3	17/11/2011	23/11/2011
2.5	Instanciació de l'ontologia: Amb la informació recollida en la tasca 2.4 instanciarem l'ontologia i realitzarem consultes per comprovar-ne el funconament. Farem les dues tasques al mateix temps.	2.3	17/11/2011	23/11/2011
2.6.	Redacció capítol 3: Al mateix temps que anem realitzant les tasques 2.1, 2.2, 2.3, 2.4 i 2.5, realitzarem el capítol de la memòria corresponent a la creació de l'ontologia.		24/10/2011	23/11/2011
3	**Creació d'un prototipus**	**2**	**24/11/2011**	**28/12/2011**
3.1	Definició de requeriments: S'analitzaran els requeriments i es farà la seva definició amb la informació proporcionada pel consultor.		24/11/2011	26/11/2011
3.2	Disseny: Es farà el disseny de l'aplicació segons els requeriments definits en la tasca 3.1.	3.1	27/11/2011	30/11/2011
3.3	Implementació: Es desenvoluparà el prototipus i es desplegarà en el servidor d'aplicacions. En aquesta tasca incloem també l'elecció, instal·lació i configuració del servidor d'aplicacions.	3.2	1/12/2011	21/12/2011

Tasca	Descripció	Prec.	Inici	Fi
3.4	Test: Amb el joc de proves proporcionats a mitjan semestre es comprovarà el correcte funcionament de l'aplicació.	3.3	22/12/2011	28/12/2011
3.5	Manual d'instal·lació: Redacció d'un manual detallat per a la realització de la instal·lació de l'aplicació.	3.3	22/12/2011	25/12/2011
3.6	Manual d'usuari: Redacció d'un manual per a l'usuari on s'indicarà amb detall l'ús de l'aplicació i les seves opcions.	3.3	26/12/2011	28/12/2011
3.7	Redacció capítol 4: Al mateix temps que anem realitzant les tasques 3.1, 3.2, 3.3 i 3.4, realitzarem el capítol de la memòria corresponent a l'anàlisi, disseny i implementació del prototipus.		24/12/2011	28/12/2011
4	**Redacció capítol 5**: Un cop finalitzat el prototipus, realitzarem el darrer capítol, que correspon a les conclusions i línies de futur.	**3**	**29/12/2011**	**2/1/2012**
5	**Revisió i finalització de la memòria**: Es farà una lectura detallada de cada capítol de la memòria, es faran les correccions corresponents i s'ensamblaran tots els capítols, s'afegirà la bibliografia i es generaran els índexs de continguts i de figures.	**4**	**2/1/2012**	**10/1/2012**
6	**Elaboració de la presentació**: Es seleccionarà la informació a incloure en la presentació, es realitzaran les transparències i es gravarà la veu.	**4**	**28/12/2011**	**10/1/212**
7	Debat virtual			

Fites

Fita	Data	Lliuraments
PAC1	3/10/2011	• Pla de treball
PAC2	7/11/2011	• Capítol 1 • Capítol 2
PAC3	19/12/2011	• Capítol 3 • Catàleg de dispositius • Ontologia instanciada
Lliurament final	10/1/2011	• Memòria completa • Prototipus amb l'ontologia instanciada • Manual d'instal·lació • Manual d'usuari • Presentació
Inici del debat virtual	23/1/2011	
Final del debat virtual	26/1/2011	

Diagrama de Gantt



Gestió de riscos

Risc	Mesura
Pèrdua accidental de la informació	Disposarem d'un ordinador de sobretaula i un Netbook, i desarem el nostre projecte en un espai d'emmagatzematge remot mitjançant l'aplicació Dropbox
Avaries de maquinari	
Malaltia o imprevistos	Els caps de setmana serà prioritari dedicar-se al projecte a fi de corregir possibles endarreriments

1.5 Recursos

Per a la realització d'aquest projecte disposarem dels següents recursos de maquinari i programari:

- Ordinador de sobretaula amb processador Phenom II X4 945 i 4 GB de RAM
- Ordinador portàtil Netbook amb processador Atom N270 i 4 GB de RAM
- Connexió a Internet
- Sistema operatiu Ubuntu 11.04
- Paquet ofimàtic LibreOffice 3
- Programari de gestió de projectes OpenProj 1.4
- Programari de diagramació UML StarUML
- Editor d'ontologies Protegé 4.1
- Programari d'emmagatzematge remot Dropbox 1.1.35
- Servidor d'aplicacions Tomcat
- Entorn de desenvolupament d'aplicacions (IDE) Eclipse Indigo
- Biblioteca de programació (API) per a la web semàntica

Es consultarà la bibliografia i webgrafia recomanada en el pla docent i es farà recerca d'altres fonts a Internet.

1.6 Productes obtinguts

Els productes resultants d'aquest projecte són:

- La memòria de desenvolupament del projecte
- Una ontologia de dispositius mòbils
- Una aplicació web que utilitza l'ontologia de dispositius mòbils per a decidir quins fitxers es poden servir a un usuari en funció del seu dispositiu mòbil
- Un manual d'instal·lació de l'aplicació.
- Un manual d'utilització de l'aplicació.

1.7 Descripció dels capítols posteriors

- Capítol 2: Ontologies

  Farem una introducció a la web semàntica i als diferents llenguatges per a la definició d'ontologies.

- Capítol 3: Definició d'una ontologia de dispositius mòbils

  Es descriuran ontologies de dispositius mòbils ja existents i s'explicarà amb detall l'ontologia creada per a aquest projecte.

- Capítol 4: Desenvolupament d'un prototip

  S'explicaran tots els passos del procés de desenvolupament de l'aplicació, això és, l'anàlisi i definició de requeriments, el disseny, el desenvolupament i les proves.

- Capítol 5: Conclusions

  S'exposaran les conclusions, el grau d'assoliment dels objectius i les línies de futur i possibles millores del projecte.

2 Ontologies

2.1 Introducció

Com ja hem dit en el primer capítol, una ontologia és una base de coneixement sobre un determinat domini, i està formada per un conjunt de definicions de classes, propietats, relacions i restriccions, i un conjunt d'instàncies (objectes) que s'ajusten a aquestes definicions. Aquesta ontologia serà normalment utilitzada per una aplicació, la qual hi accedirà a través d'una sèrie de funcions d'una biblioteca de programació (API).

El llenguatge utilitzat per a la representació d'ontologies és el llenguatge OWL, que actualment es troba en la seva versió 2. Aquest llenguatge té una definició formal i una sèrie de representacions, de les quals la més habitual és la basada en XML/RDF. Existeixen altres sintaxis, com la OWL Manchester, que són més fàcils d'utilitzar per una persona.

El dissenyador d'ontologies, però, té a l'abast una sèrie d'eines que fan menys feixuga la feina d'escriure ontologies. En aquest projecte utilitzarem una d'aquestes eines, el Protégé 4.1.

En aquest capítol farem una breu introducció de XML i XML Schema, de RDF i RDF Schema, que són la base de OWL. Explicarem a grans trets com s'utilitza el llenguatge OWL per a definir ontologies, centrant-nos en la sintaxi RDF/XML, i també comentarem la sintaxi OWL Manchester, que utilitza el programari Protége 4.1. A més, veurem com es pot enriquir l'ontologia amb regles, utilitzant el llenguatge de definició de regles SWRL.

Després veurem com podem fer consultes en una ontologia. L'opció bàsica és SPARQL, que està pensat per a fer consultes en qualsevol tipus de document RDF. Tenim però, altres opcions que treballen directament sobre el llenguatge OWL com són el llenguatge SQWRL, basat en SWRL, i la sintaxi Manchester.

Finalment, farem una comparativa de les diferents biblioteques de programació que podem fer servir per a implementar una aplicació que utilitzi una ontologia.

2.2 XML i XML Schema

XML és un llenguatge de marques derivat del SGML que va ser creat per a poder representar la informació de manera estructurada per a ser processada per un computador. Els documents XML no contenen informació sobre el format i la presentació de la informació. Les diferents etiquetes dels documents XML estan definides per l'usuari o aplicació que els ha d'utilitzar.

Estructura d'un document XML

Un document XML conté una capçalera i a continuació un número d'elements. La capçalera consta de la declaració i una referència a documents externs.

La declaració defineix la versió i la codificació de carters, per exemple:

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Un exemple de referència a documents externs seria la que mostrem a continuació, que en aquest cas és una referència a un document DTD, que defineix l'estructura del document XML:

```
<!DOCTYPE devices SYSTEM “devices.dtd”>
```

Els elements del document XML són els objectes que aquest representa. Cada element està format

per una etiqueta d'obertura, el contingut i una etiqueta de tancament. El contingut pot ser text o bé altres elements. Per exemple:

```
<device>
      <manufacturer>Samsung</manufacturer>
      <model>Galaxy ACE</model>
</device>
```

Un element també pot tenir atributs, els quals es defineixen dins de l'etiqueta d'obertura. Per exemple:

```
<hardware type=”display”>
      <property name=”color” value=”yes”/>
      <property name=”resolution” value “300x200”/>
</hardware>
```

També pot haver-hi elements buits, com l'element `property` de l'exemple anterior, que podem abreviar escrivint només l'etiqueta d'obertura tancada amb acabada amb `/>`.

Es diu que un document XML és ben format si:

- Només conté un element que no estigui dins d'un altre element. Aquest s'anomena element arrel
- Tots els elements tenen obertura i tancament, amb la forma abreviada o no.
- Els elements no estan sobreposats, com seria en el cas:

```
<device><model>Galaxy ACE</device></model>
```

- Els elements d'un atribut tenen noms únics.

DTD i XML Schema

L'estructura d'un document XML està definida per un altre document que conté les regles sintàctiques que ha de seguir. Hi ha dos tipus de documents que defineixen l'estructura dels documents XML: DTD i XML Schema.

Els DTDs defineixen els tipus d'elements, el lloc i l'ordre en què apareixen, i els atributs que tenen. Per als atributs i valors, també permet definir-ne diverses classes, però DTD no permet especificar tipus de dades.

XML Schema afegeix molta més riquesa per a definir l'estructura dels documents XML. El llenguatge XML Schema permet:

2. Definir la cardinalitat dels elements que apareixen.
3. Definir atributs opcionals.
4. Establir tipus de dades per als valors. N'hi ha una gran varietat de pre-definits.
5. Definir nous tipus simples i complexos.
6. Estendre nous tipus a partir de tipus existents.
7. Definir restriccions als tipus de dades, com per exemple enumeracions.

Els documents XML Schema estan escrits seguint la sintaxi XML, és a dir, són documents XML. Aquest és un exemple de document *XML Schema* extret del tutorial de www.w3schools.com:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:element name="note">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="to" type="xs:string"/>
                  <xs:element name="from" type="xs:string"/>
                  <xs:element name="heading" type="xs:string"/>
                  <xs:element name="body" type="xs:string"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>

</xs:schema>
```

Es tracta d'un document XML que té com a element arrel un element `xs:schema` i dins d'ell els elements per a definir l'estructura d'un document XML. L'atribut `xmlns` de l'element `xs:schema` defineix l'espai de noms corresponent al volcabulari de *XML Schema*, les paraules del qual contenen el prefix `xs:`. Un XML vàlid per a aquest esquema seria:

```
<?xml version="1.0"?>
<note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
</note>
```

2.3 RDF i RDF Schema

RDF (Resource Description Framework)

RDF és un model de dades per a descriure recursos que utilitza la sintaxi XML per a la seva representació. Els conceptes fonamentals en RDF són:

- Recursos: Un recurs és alguna cosa de la qual volem parlar, per exemple un llibre, un lloc, un dispositiu, etc. Cada recurs té un URI (Universal Resource Identifier) associat que l'identifica.
- Propietats: Descriuen relacions entre recursos, per exemple “es troba a”, “conté”, etc. Les propietats també s'identifiquen mitjançant URIs.
- Declaracions: Diuen propietats dels recursos. Estan formades per un recurs del qual es diu alguna cosa, la propietat que se'n diu i el valor que s'assigna a la propietat. Aquest valor pot ser un text o bé un altre objecte.

Un document RDF és en definitiva un conjunt de declaracions, que expressades en llenguatge natural serien de la forma: el recurs A té en la propietat B el valor C. És a dir, un document RDF està format per un conjunt de triples (recurs, propietat, valor). Les propietats i valors poden ser també altres recursos RDF.

Els documents RDF són documents XML que contenen com a element arrel l'element `rdf:RDF`., que conté elements del tipus `rdf:Description`. Els elements `rdf:Description` tenen l'atribut `rdf:about` o l'atribut `rdf:ID`, que tenen com a valor l'URI del recurs que es descriu. El contingut d'aquest element són un seguit d'elements amb valor, que estableixen propietats i valors del recurs. Veiem un exemple de document RDF:

```
<?xml:version=”1.0” encoding=”UTF-8”?>

<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:md="http://www.mobiledevices.org/md-ns#">
<rdf:Description rdf:about=”http://www.mobiledevices.org/devices#Galaxy_ACE”>
      <md:isManufacturedBy>Samsung</md:isManufacturedBy>
</rdf:Description>
</rdf:RDF>
```

Els atributs `rdf:about` i `rdf:ID` tenen el mateix significat, és a dir, identificar el recurs sobre el qual es descriuen propietats. Com que pot haver-hi descripcions d'un mateix recurs en llocs diferents, es sol posar `rdf:ID` en el lloc on es defineix el recurs i `rdf:about` en la resta de llocs on es descriuen altres propietats del mateix recurs.

Com es veu en l'exemple, en l'etiqueta de l'element arrel `rdf:RDF` es fa referència a espais de noms. En RDF els espais de noms no només proveeixen vocabulari, sinó que poden ser altres documents RDF que contenen recursos que es poden reutilitzar.

Com hem dit més amunt, les propietats poden ser un text o bé una referència a un altre recurs. Les referències de propietats a altres recursos es fan afegint l'atribut `rdf:resource` a l'etiqueta d'obertura de la propietat. Seguint amb l'exemple anterior, el modifiquem per a fer referència a un recurs i queda així:

```
<rdf:Description rdf:about=”http://www.mobiledevices.org/devices#Galaxy_ACE”
      rdf:type=”http://www.mobiledevices.org/md-ns/Device”>
      <md:isManufacturedBy
            rdf:resource=”http://www.mobiledevices.org/brands#Samsung”/>
</rdf:Description>
```

Es pot especificar el tipus d'un recurs amb l'atribut `rdf:type`, com mostrem en l'exemple anterior. També és possible utilitzar el mètode abreviat, que seria com mostrem a continuació:

```
<md:Device rdf:about=”http://www.mobiledevices.org/devices#Galaxy_ACE”>
      <md:isManufacturedBy
            rdf:resource=”http://www.mobiledevices.org/brands#Samsung”/>
</md:Device>
```

RDF també disposa d'elements contenidors, que es poden especificar com a valors de les propietats, i que contenen diversos elements. Són els següents:

- `<rdf:Bag>` en què l'ordre no és important.
- `<rdf:Seq>` on els elements estan ordenats.
- `<rdf:Alt>` en què els elements són un conjunt d'alternatives.

Els elements anteriors contenen un conjunt d'elements, però no permeten especificar que no pot haver-hi cap més element. Per a això existeix l'element `rdf:List`.

RDF permet també la reificació, és a dir, fer declaracions sobre altres declaracions. Això es pot fer si definim una declaració amb l'element rdf:Statement, que té com a atribut l'identificador `rdf:about`, i com a elements els tres implicats en la declaració objecte:

- `rdf:subject` amb la referència al recurs
- `rdf:predicate` amb la referència a la propietat
- `rdf:object` amb el valor de la propietat

Per exemple, definim una declaració de la següent manera:

```
<rdf:Statement rdf:about=”declaracio1”>
      <rdf:subject
      rdf:resource=”http://www.mobiledevices.org/devices#Galaxy_ACE”/>
      <rdf:predicate rdf:resource=”md:isManufacturedBy”/>
      <rdf:object>Grigoris Antoniou</rdf:object>
</rdf:Statement>
```

Aleshores podem utilitzar l'identificador “declaracio1” en qualsevol altra declaració.

RDF Schema

Amb RDF Schema es pot definir el vocabulari utilitzat en els models RDF, especificar quines propietats es poden aplicar a quins tipus d'objectes i quins valors poden tenir.

RDF Schema permet definir classes d'objectes, i d'aquesta manera restringir el rang i el domini de les declaracions RDF. Entenem per domini el conjunt de recursos que poden estar en l'atribut `rdf:about` d'una declaració, i per rang el conjunt de valors que pot prendre cada una de les propietats de la declaració.

També és possible l'herència i d'aquesta manera establir una jerarquia de classes. Així doncs, un objecte que és d'una classe X que és subclasse de Y, és també un objecte de la classe Y.

L'herència també es pot aplicar a les propietats. P és una subpropietat de Q si es compleix Q(x,y) sempre que es compleix P(x,y).

Les definicions RDF Schema es fan utilitzant RDF, i mitjançant l'espai de noms `rdfs` es proveeixen les seves primitives, que llistem a continuació.

Classes

- `rdfs:Resource`, la classe de tots els recursos
- `rdfs:Class`, la classe de totes les classes
- `rdfs:Literal`, la classe de tots els literals
- `rdf:Property`, la classe de totes les propietats
- `rdf:Statement`, la classe de totes les declaracions reificades

Propietats per a definir relacions

- `rdf:type`, per a definir la classe d'un recurs
- `rdfs:subClassOf`, per a relacionar una classe amb una de les seves superclasses
- `rdfs:subPropertyOf`, per a relacionar una propietat amb una de les seves superpropietats

Propietats per a restringir propietats

- `rdfs:domain`, per a especificar el domini d'una propietat
- `rdfs:range`, per a especificar el rang d'una propietat
- `rdfs:ConstraintResource`, la classe de totes les restriccions
- `rdfs:ConstraintProperty`, que conté totes les propietats que defineixen restriccions

Propietats per a reificació

- `rdf:subject`
- `rdf:predicate`
- `rdf:object`

Classes contenidores

- `Rdf:Bag`
- `rdf:Seq`
- `rdf:Alt`
- `rdfs:Container`, que és una superclasse de les classes anteriors

Propietats d'utilitats

- `rdfs:seeAlso` relaciona un recurs amb un altre que l'explica
- `rdfs:isDefinedBy` relaciona un recurs amb el lloc on està definit
- `rdfs:Comment` permet associar un comentari a un recurs
- `rdfs:label` permet posar una etiqueta a un recurs

Com a exemple mostrarem un fragment de la descripció RDFS de CC/PP (Composite Capabilities/Preference Profiles), que descriu capacitats de dispositius:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-
20020710#">
<rdf:Description ID="Component">
      <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
      <rdfs:label>Component</rdfs:label>
      <rdfs:comment>
      A Component within the CC/PP Schema is a class of related properties
      that describe the capabilities and preferences information.
      </rdfs:comment>
</rdf:Description>
<!-- ************************************************************** -->
<!-- ***** Properties shared among the components***** -->
<rdf:Description ID="component">
      <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
      <rdfs:label>component</rdfs:label>
      <rdfs:comment>
      The component attribute links the various components to the root node
      (profile).
      </rdfs:comment>
</rdf:Description>

<rdf:Description ID="defaults">
      <rdfs:type rdf:resource="http://www.w3.org/2000/01/rdf-
```

```
schema#Property"/>
      <rdfs:domain rdf:resource="#HardwarePlatform"/>
      <rdfs:domain rdf:resource="#SoftwarePlatform"/>
      <rdfs:domain rdf:resource="#WapCharacteristics"/>
      <rdfs:domain rdf:resource="#BrowserUA"/>
      <rdfs:domain rdf:resource="#NetworkCharacteristics"/>
      <rdfs:domain rdf:resource="#PushCharacteristics"/>
      <rdfs:comment>
     An attribute used to identify the default capabilities.
      </rdfs:comment>
</rdf:Description>

<!-- ************************************************************** -->
<!-- ***** Component Definitions ***** -->
<rdf:Description ID="HardwarePlatform">
      <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#Component"/>
      <rdfs:label>Component: HardwarePlatform</rdfs:label>
      <rdfs:comment>
      The HardwarePlatform component contains properties of the device's
      Hardware, such as display size, supported character sets, etc.
      </rdfs:comment>
</rdf:Description>

<rdf:Description ID="SoftwarePlatform">
      <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#Component"/>
      <rdfs:label>Component: SoftwarePlatform</rdfs:label>
      <rdfs:comment>
      The SoftwarePlatform component contains properties of the device's
      application environment, operating system, and installed software.
      </rdfs:comment>
</rdf:Description>
```

Podem veure com es descriu la classe `Component` com a subclasse de `Resource`, i que té les classes `HardwarePlatform` i `SoftwarePlatform` com a sublcasses. Cada classe té una etiqueta i un comentari.

Es descriuen també les propietats `component` i `defaults`, també amb una etiqueta i un comentari. Per a la classe `defaults` se'n defineix el domini, és a dir, les classes d'objectes als quals es pot aplicar.

2.4 OWL

Amb RDF i RDF Schema és possible la realització de models de domini, però el llenguatge té poca expressivitat. Hi trobem a faltar les característiques següents:

- Àmbit local de les propietats
- Classes disjuntes
- Combinacions de classes
- Restriccions de cardinalitat
- Característiques de les propietats, com la transitivitat, la unicitat o la inversa.

Per a poder definir ontologies es fa necessari l'ús d'un altre llenguatge amb molta més riquesa

semàntica, i aquest és el OWL (Ontology Web Language). Aquest està basat en RDF i RDF Schema i utilitza la sintaxi XML, utilitzant els elements dels espais de noms de `xsd`, `rdf` i `rdfs`, i afegeix nou vocabulari amb l'espai de noms `owl`. També existeix una sintaxi abstracta molt més fàcil de llegir, una representació gràfica basada en UML (Unified Modeling Language) i la sintaxi Manchester.

Sintaxi RDF/XML. Estructura d'un document OWL

Capçalera

Un document OWL és un document RDF, amb l'element `rdf:RDF` com a element arrel, que especifica els següents espais de noms:

```
<rdf:RDF
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

A continuació tenim l'element owl:Ontology, que conté la versió, comentaris i importacions d'altres ontologies.

```
<owl:Ontology rdf:about="http://www.mobiledevices.org/md.owl">
      <rdfs:comment>Ontologia de dispositius mòbils</rdfs:comment>
      <owl:importsdf:resouce="http://www.cenitmio.es/ontologies/Device.owl"/>
</owl:Ontology>
```

Elements de classe

Les classes es defineixen amb l'element `owl:Class`, que contenen altres elements de diferents tipus que en defineixen les característiques. Per exemple, definim una classe que és subclasse d'una altra classe:

```
<owl:Class rdf:about="&ontologies;dispositiusmobils.owl#AudioCodec">
      <rdfs:subClassOf
            rdf:resource="&ontologies;dispositiusmobils.owl#Codec"/>
</owl:Class>
```

Altres característiques que podem definir d'una classe són les seves classes disjuntes (`owl:Disjoint`) i les seves classes equivalents (`owl:Equivalent`).

Existeixen les classes predefinides `owl:Thing` i `owl:Nothing`. La primera és la classe de la qual deriven totes les classes, i la segona és la classe buida de la qual totes les classes són superclasse.

Elements de propietat

Les propietats es defineixen amb els elements `owl:DatatypeProperty` i `owl:ObjectProperty`. El primer relaciona un objecte amb un valor d'un tipus de dades determinat, mentre que el segon relaciona un objecte amb un altre objecte. En OWL2 totes les propietats d'objecte són subpropietats de la propietat `topObjectProperty`, i per tant relaciona cada parell d'objectes existents.

```
<owl:ObjectProperty
      rdf:about="&ontologies;dispositiusmobils.owl#digitalCompass">
   <rdfs:domain rdf:resource="&context;hardware.owl#DeviceHardware"/>
   <rdfs:subPropertyOf
      rdf:resource="&context;hardware.owl#hardwareComponent"/>
</owl:ObjectProperty>
```

Restriccions de propietats

En OWL es defineixen restriccions sobre les propietats d'una classe declarant que la classe és subclasse d'una classe anònima que compleix una sèrie de condicions. S'utilitza l'element `rdfs:subClassOf` en combinació amb l'element `owl:Restriction`.

```
<owl:Class rdf:about=”#Device">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasHardware"/>
                <owl:allValuesFrom rdf:resource="#Hardware"/>
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

En aquest cas hem utilitzat el quantificador universal `owl:allValuesOf` per a definir la classe `Device` com una subclasse d'una classe anònima d'objectes en què tots els objectes amb què es relaciona mitjançant la propietat `hasHardware` són del tipus `Hardware`. És a dir, un dispositiu és un objecte que pot tenir components de maquinari. Es poden definir altres tipus de restriccions utilitzant el quantificador existencial `owl:someValuesFrom`, o restriccions de cardinalitat amb els elements `owl:minCardinality` o `owl:maxCardinality`.

Característiques de les propietats

Es poden definir algunes característiques especials de les propietats, que són:

- `owl:TransitiveProperty` defineix una propietat com a transitiva, és a dir, que si P(x,y) i P(y,z) aleshores P(x,z)
- `owl:SymmetricProperty` defineix una propietat simètrica, és a dir, que si P(x,y) aleshores P(y,x)
- `owl:FunctionalProperty` defineix una propietat funcional, és a dir, que té un sol valor per a cada objecte
- `owl:InverseFunctionalProperty` defineix una propietat funcional inversa, és a dir, que dos objectes no poden tenir el mateix valor per a aquesta propietat

Combinacions booleanes

Es poden crear restriccions que són combinació de diferents condicions. Això es fa utilitzant els elements `owl:unionOf`, `owl:intersectionOf`, que contenen una col·lecció de condicions que conformen la restricció combinades amb l'operador unió i intersecció respectivament. També es pot utilitzar l'element `owl:complementOf` per a expressar el complementari d'una condició.

Tot seguit un exemple d'ús de `owl:intersecionOf,` en què definiim una pantalla tàctil com a una intersecció entre una pantalla i un dispositiu d'entrada.

```
<owl:Class rdf:about="&context;hardware.owl#TactileDisplay">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
                         rdf:about="&context;hardware.owl#Display"/>
                    <rdf:Description
                         rdf:about="&context;hardware.owl#InputDevice"/>
                </owl:intersectionOf>
```

```
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>
```

Enumeracions

Es pot definir una classe com una llista de tots els seus elements, utilitzant l'element `owl:oneOf`. En l'exemple següent l'utilitzem per a definir la classe `BitrateType` com a enumeració dels objectes `VBR` i `CBR`.

```
<owl:Class rdf:about="&ontologies;dispositiusmobils.owl#BitRateType">
        <owl:equivalentClass>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
                         rdf:about="&ontologies;dispositiusmobils.owl#VBR"/>
                    <rdf:Description
                         rdf:about="&ontologies;dispositiusmobils.owl#CBR"/>
                </owl:oneOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf
             rdf:resource="&context;software.owl#Context_SoftwareEntity"/>
</owl:Class>
```

Instàncies

Les instàncies es declaren de la mateixa manera que en RDF:

```
<Device rdf:ID="iPhone 4"/>
```

En OWL 2 s'utilitza l'element NamedIndividual:

```
<owl:NamedIndividual rdf:about="iPhone 4">
      <rdf:type rdf:resource=”Device"/>
</owl:NamedIndividual>
```

Sintaxi Manchester

La sintaxi RDF/XML no és gaire fàcil de llegir i escriure per un ésser humà. A l'hora de crear ontologies s'utilitzen eines que fan més fàcil aquesta tasca, com *Protégé 4.1*, que és la que utilitzarem en aquest projecte. *Protégé 4.1* utilitza la sintaxi *Manchester* per a definir les característiques de les classes, i a més facilita la seva edició amb una interfície que permet seleccionar classes, propietats i paraules claus amb un clic de ratolí evitant així haver d'escriure. Comentarem breument en aquest apartat com és la sintaxi Manchester.

A continuació veiem quines són les paraules clau equivalents en la sintaxi Manchester per a cada element RDF/XML.

Restriccions

OWL RDF/XML	OWL Manchester	Exemple
`someValuesFrom`	`some`	`supportedNetworkModes some NetworkMode`
`allValuesFrom`	`only`	`audioCodec only AudioCodec`
`hasValue`	`value`	`resolutionWidth value 300`
`minCardinality`	`min`	`battery min 1 Battery`

`cardinality`	`exactly`	`operatingSystem exactly 1 OperatingSystem`
`maxCardinality`	`max`	`battery max 1 Battery`

Combinacions booleanes

OWL RDF/XML	OWL Manchester	Exemple
`intersectionOf`	`and`	`Display and InputDevice`
`unionOf`	`or`	`name value "IEEE 802.11a" or name value "IEEE 802.11b"`
`complementOf`	`not`	not Device

Valors i tipus de dades

Podem utilitzar diferents tipus de dades depenent de la implementació. Per exemple

```
resolutionWidth some int
```

Si volem indicar un valor especificant que és d'un tipus determinat ho fem de la següent manera

```
resolutionWidth value “300”^^int
```

També es poden especificar restriccions sobre els tipus de dades, per exemple

```
resolutionWidth some int[>=300]
```

Aquests són els tipus de restriccions que es poden especificar, tot i que no tots els raonadors les suporten.

Restricció	Significat
`< x, <= x`	Menor que, menor o igual que x
`> x, >= x`	Major que, major o igual que x
`length x`	Per a una cadena de caràcters, la longitud igual x
`maxLength x`	Per a una cadena de caràcters, la longitud màxima de x
`minLength x`	Per a una cadena de caràcters, la longitud mínima de x
`pattern regexp`	Coincideix amb el patró expressat amb una expressió regular
`totalDigits x`	Per a un número, el número de dígits és x
`fractionDigits x`	Per a un número, el número de dígits decimals és x

Expressions complexes

Podem combinar les expressions exposades més amunt per a formar expressions complexes de classe, com per exemple:

```
DeviceHardware and primaryCamera some (Camera and cameraFlash value true)
```

Diferències entre OWL i OWL 2

- Algunes millores en l'expressivitat de la sintaxi:
  - Conjunts de classes disjuntes. En OWL 1 s'havia de definir parell a parell.
  - Unió de classes disjuntes. En OWL2 hi ha l'element `owl:disjointUnionOf`. En OWL1 s'havien de definir primer les classes com a disjuntes i després definir una superclasse

com a unió.

- Especificar valors que una propietat d'un objecte no té amb l'element `owl:NegativePropertyAssertion`.

- Noves construccions per a les propietats:
  - owl:hasSelf Restricció que un objecte té a ell mateix com a valor d'una propietat.
  - Restriccions de cardinalitat qualificada.
  - Propietats reflexives, irreflexives i asimètriques.
  - Propietats disjuntes.
  - Definir una propietat com a composició d'unes altres
  - Permet definir propietats com a claus úniques de cada instància
- OWL 1 utilitza els tipus de dades de *XML Schema*. OWL 2 incorpora molts més tipus de dades i restriccions als tipus de dades, com per exemple rangs de valors, longituds de cadenes, etc. També es poden combinar rangs amb els operadors unió, intersecció i complement.
- OWL 2 permet el mateix nom per a una classe i una instància.
- El mecanisme d'anotacions (etiquetes) és més complet, permetent crear jerarquies de tipus d'anotacions i assignar dominis i rangs als diferents tipus.
- Totes les propietats són subpropietats de `topObjectProperty` i de `topDataProperty`
- Usa IRIs (Internationalized Resource Identifiers) enlloc de URIs (Uniform Resource Locators), amb la qual cosa es permeten noms amb caràcters no només de l'alfabet anglès.
- Es pot utilitzar la inversa d'una propietat sense haver-la definit.

OWL 1 està implementat en la biblioteca *Protege-OWL* que utilitza *Protégé 3.4*, mentre que OWL 2 està implementat en la biblioteca OWL-API que utilitza *Protégé 4.1*. La biblioteca *Jena* suporta OWL 1.

2.5 SWRL

El llenguatge SWRL (Semantic Web Rule Language) és un llenguatge de definició de regles basat en OWL. Una regla SWRL té la següent forma:

$$atom \wedge atom \wedge \ldots \rightarrow atom \wedge atom \wedge \ldots$$

on distingim el cos o antecedent i la capçalera o conseqüent, ambdós formats per conjuncions d'àtoms. La regla indica que si es compleixen els àtoms que formen el cos, aleshores també es compleixen els àtoms de la capçalera.

Els àtoms són predicats amb un nombre determinat d'arguments, i en trobem de diferents tipus:

- De classe: el predicat és una definició de classe OWL i hi ha un sol argument que representa un objecte.
- De propietat tipus objecte: el predicat és una propietat tipus objecte i hi ha dos arguments que representen els dos objectes relacionats per la propietat.

- De propietat amb valor de dada: el predicat és una propietat tipus dada i hi ha dos arguments, un que representa l'objecte a què s'aplica la propietat i l'altre que representa el valor de la propietat.
    - D'objectes diferents: el predicat és el símbol `differentFrom` i els arguments són dos objectes, i s'avalua a cert quan els objectes són diferents.
    - D'objectes iguals: el predicat és el símbol `sameAs` i els arguments són dos objectes, i s'avalua a cert quan els objectes són el mateix.
    - Predefinits del llenguatge: SWRL disposa de funcions matemàtiques i de tractament de cadenes que es poden utilitzar. Estan definits en l'espai de noms `swrlb`.
- De rang: el predicat és un tipus de dades o un conjunt de valors, i hi ha un únic argument que representa un valor.

El llenguatge SWRL es pot escriure o bé utilitzant una sintaxi XML o bé utilitzant una sintaxi RDF. Les regles SWRL escrites amb la sintaxi RDF es poden desar en les ontologies OWL. Per exemple, la regla:

```
nextVersion(?x,?z),supportedMediaFormats(?x,?y)->supportedMediaFormats(?z,?y)
```

s'escriuria utilitzant la sintaxi RDF:

```
<swrl:Imp>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate
rdf:resource="&ontologies;dispositiusmobils.owl#supportedMediaFormats"/>
                        <swrl:argument2 rdf:resource="urn:swrl#y"/>
                        <swrl:argument1 rdf:resource="urn:swrl#z"/>
                    </swrl:IndividualPropertyAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
        <swrl:body>
            <swrl:AtomList>
                <rdf:first>
                    <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate
rdf:resource="&ontologies;dispositiusmobils.owl#nextVersion"/>
                        <swrl:argument1 rdf:resource="urn:swrl#x"/>
                        <swrl:argument2 rdf:resource="urn:swrl#z"/>
                    </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:rest rdf:resource="&rdf;nil"/>
                        <rdf:first>
                            <swrl:IndividualPropertyAtom>
                                <swrl:propertyPredicate
rdf:resource="&ontologies;dispositiusmobils.owl#supportedMediaFormats"/>
                                <swrl:argument1 rdf:resource="urn:swrl#x"/>
                                <swrl:argument2 rdf:resource="urn:swrl#y"/>
                            </swrl:IndividualPropertyAtom>
```

```
                        </rdf:first>
                    </swrl:AtomList>
                </rdf:rest>
            </swrl:AtomList>
        </swrl:body>
    </swrl:Imp>
```

Aquest llenguatge està implementat a la biblioteca *Protegé-OWL*, en què es basa *Protégé 3.4*, i es pot utilitzar fàcilment a través d'un plugin anomenat *SWRLTab*. També està implementat a la biblioteca *OWL-API*, en què es basa *Protégé 4.1*, i es pot utilitzar activant la vista *Rules*, tot i que es tracta d'un editor més bàsic que el plugin de *Protégé 3.4*.

2.6 SPARQL

SPARQL (*Simple Protocol and RDF Query Language*) és el llenguatge de consultes per a RDF. Una consulta SPARQL conté una clàusula `SELECT` amb les variables que es volen obtenir com a resultats de la consulta, seguida d'una clàusula `WHERE` que conté una tripla RDF (recurs, propietat, valor) en què algun dels seus elements pot ser una variable. A continuació mostrem un exemple:

```
PREFIX md: <http://www.semanticweb.org/ontologies/2011/9/mobiledevices.owl#>
   1. SELECT  ?x ?y
WHERE
  { ?x  md:operatingSystem  ?y .}
```

Aquesta consulta retornaria tots els dispositius existents i el seu sistema operatiu. Una consulta pot tenir diverses triples que van separades pel caràcter '.' i el resultat de la consulta seran totes les coincidències que hi hagi substituint les variables pels valors existents en el document RDF. Si en la consulta es volen especificar diverses triples amb el mateix subjecte, es poden especificar separades pel caràcter ';' i sense haver de repetir el subjecte. Per exemple:

```
SELECT  ?x ?y
WHERE
  { ?x      md:operatingSystem          ?y ;
            md:hasHardware              ?z .
}
```

Quan es vol especificar un literal cal posar el seu valor entre cometes i a continuació els símbols "^^" seguits del tipus de dades. En l'exemple següent consultem les pantalles que tenen una resolució de 300 columnes.

```
SELECT ?display
WHERE { ?display md:resolutionWidth "300"^^xsd:int . }
```

Es pot utilitzar la funció `FILTER` per a restringir els valors de coincidència de les variables. Per a restringir valors numèrics ho fem com en el següent exemple:

```
SELECT ?display
WHERE { ?display md:resolutionWidth ?columns .
            FILTER (?columns >= 300) }
```

Si els valors que volem restringir són cadenes de text, utilitzarem la funció `FILTER` en combinació amb la funció `regex`, com en l'exemple següent en què volem obtenir totes les instàncies en que la propietat `model` és un text que comença per "Sam":

```
SELECT ?brand
WHERE { ?brand md:model ?name .
            FILTER regex(?name, "^Sam") }
```

Aquest llenguatge es troba implementat en les biblioteques *Jena*, *Protege-OWL* i *OWL-API*. *Protégé 3.4* i *Protégé 4.1* disposen d'editors de consultes SPARQL. Cal dir que a *Protégé 4.1* hi ha

algunes funcions del llenguatge com `FILTER` que no funcionen.

2.7 SQWRL

Com que el llenguatge OWL es pot escriure utilitzant RDF, el llenguatge SPARQL és vàlid per a obtenir informació d'una ontologia OWL. No obstant, SPARQL no té coneixement de les estructures de OWL, sinó únicament de la seva representació en RDF. A més, la representació d'una ontologia OWL en RDF no és única, i per tant les consultes SPARQL hauran d'adaptar-se a les diferents formes de representació XML que usen les diferents eines d'edició d'ontologies.

El llenguatge SQWRL (*Semantic Query-enhanced Web Rule Language*, *squirrel*) està basat en el llenguatge SWRL incorporant-hi noves construccions per a la realització de les consultes, que estan definides en l'espai de noms *sqwrl*. Les consultes definides amb SQWRL es p

odran desar per tant en les ontologies OWL. Es pot utilitzar també amb el plugin *SWRLTab* de *Protégé 3.4*.

L'operador principal del llenguatge és *sqwrl:select*, que sol trobar-se en el conseqüent d'una regla i els paràmetres del qual poden ser variables. Aquest operador retorna una taula amb les diferents instanciacions possibles de les variables que té com a arguments, i que fan que els àtoms de l'antecedent de la regla s'avaluïn a cert. Per exemple, si volem obtenir tots els dispositius que tenen algun maquinari del tipus Sensor farem la consulta:

```
hardwareComponent(?x, ?y) ∧ Sensor(?y) → sqwrl:select(?x)
```

Existeixen altres operadors amb un funcionament semblant a SQL, que permeten especificar l'ordre en què es presenten els resultats, agrupacions, realitzar operacions com mitjanes, comptabilitzar elements, etc.

El fet d'estar basat en SWRL fa que el llenguatge tingui limitacions. Per exemple, no es pot utilitzar la disjunció. Per això es proveeixen una sèrie d'operadors predefinits per a poder realitzar aquest i altres tipus de consultes.

2.8 DL Query

L'editor d'ontologies *Protégé 4.1* incorpora el plugin *DL Query*, el qual permet fer cerques en l'ontologia a partir d'una expressió de classe utilitzant la sintaxi *Manchester*, com si de definir una classe es tractés. Així doncs una consulta seria una definició d'una classe a partir de restriccions de les seves propietats. Per exemple, si volem obtenir les instàncies que tenen algun component de hardware:

```
hardwareComponent some HardwareComponent
```

2.9 Biblioteques de programació

A l'hora d'implementar una aplicació que faci ús d'una ontologia, podem utilitzar diferents biblioteques de programació. *Protégé 3.4* està basat en la biblioteca *Protege-OWL*, mentre que *Protégé 4.1* està basat en la biblioteca *OWL-API*. També tenim l'opció d'utilitzar la biblioteca *Jena*. En la següent taula llistem les característiques més importants de cada una d'elles.

Jena	Protege-OWL	OWL-API
Suport OWL 1.0	Suport OWL 1.0	Suport OWL 2.0

Suport SPARQL	Suport SPARQL	Suport SPARQL
	Suport SWRL	Suport SWRL
	Suport SQWRL	Suport OWL Manchester

3 Ontologia de dispositius mòbils

3.1 Ontologies existents

Repositori WURFL (*Wireless Universal Resource File)*

WURFL és un repositori de descripció de dispositius. Es pot trobat tota la informació del projecte en la seva pàgina web:

http://wurfl.sourceforge.net/

Es tracta d'un fitxer en XML amb informació sobre dispositius i una API de programació per a accedir a les dades. L'objectiu d'aquest projecte és proporcionar als programadors d'aplicacions WAP un conjunt de funcions per a determinar les característiques dels dispositius que hi accedeixen a través dels seus navegadors, i d'aquesta manera personalitzar els continguts a mostrar.

Bàsicament el fitxer XML conté llistats de funcionalitats per als diferents dispositius. Els dispositius estan agrupats per famílies totes elles descendents d'un tipus de dispositiu genèric. Al mateix temps, cada família té famílies descendents més especialitzades. Cada família hereta totes les funcionalitats de les seves ascendents. És a dir, un dispositiu o família té totes les característiques especificades com a pròpies més les de totes les famílies de les quals és descendent. Quan alguna propietat apareix en més d'una de les famílies ascendents, es considera com a vàlid el valor de la propietat de la família més especialitzada.

L'estructura del document XML és la següent:

L'element arrel és `<wurfl>`. Aquest conté l'element `<version>` i l'element `<devices>`. El primer conté informació sobre la versió de WURFL i el segon conté la informació dels dispositius mòbils. L'element `<devices>` conté un nombre indefinit d'elements `<device>`, que corresponen a un dispositiu o família de dispositius. Cada dispositiu té els següents atributs:

- id, que és el nom que identifica el dispositiu o família
- user_agent
- fall_back, que indica la família a la qual pertany i de la qual hereda totes les funcionalitats.

Cada element `<device>` té un nombre indeterminat d'elements `<group>`, que corresponen a grups de funcionalitats, i cada element `<group>` té un nombre indeterminat d'elements `<capability>`, que corresponen a una funcionalitat del dispositiu. Cada element `<capability>` té els atributs “name” i “value”, que són el nom i el valor de la funcionalitat.

A continuació veiem un exemple de dispositiu:

```
<device id="nokia_generic_series30" user_agent="Nokia 30"
      fall_back="nokia_generic_series20">

      <group id="product_info">
            <capability name="mobile_browser" value="Nokia"/>
            <capability name="nokia_series" value="30"/>
            <capability name="nokia_edition" value="1"/>
            <capability name="can_skip_aligned_link_row" value="true"/>
      </group>
      <group id="wml_ui">
```

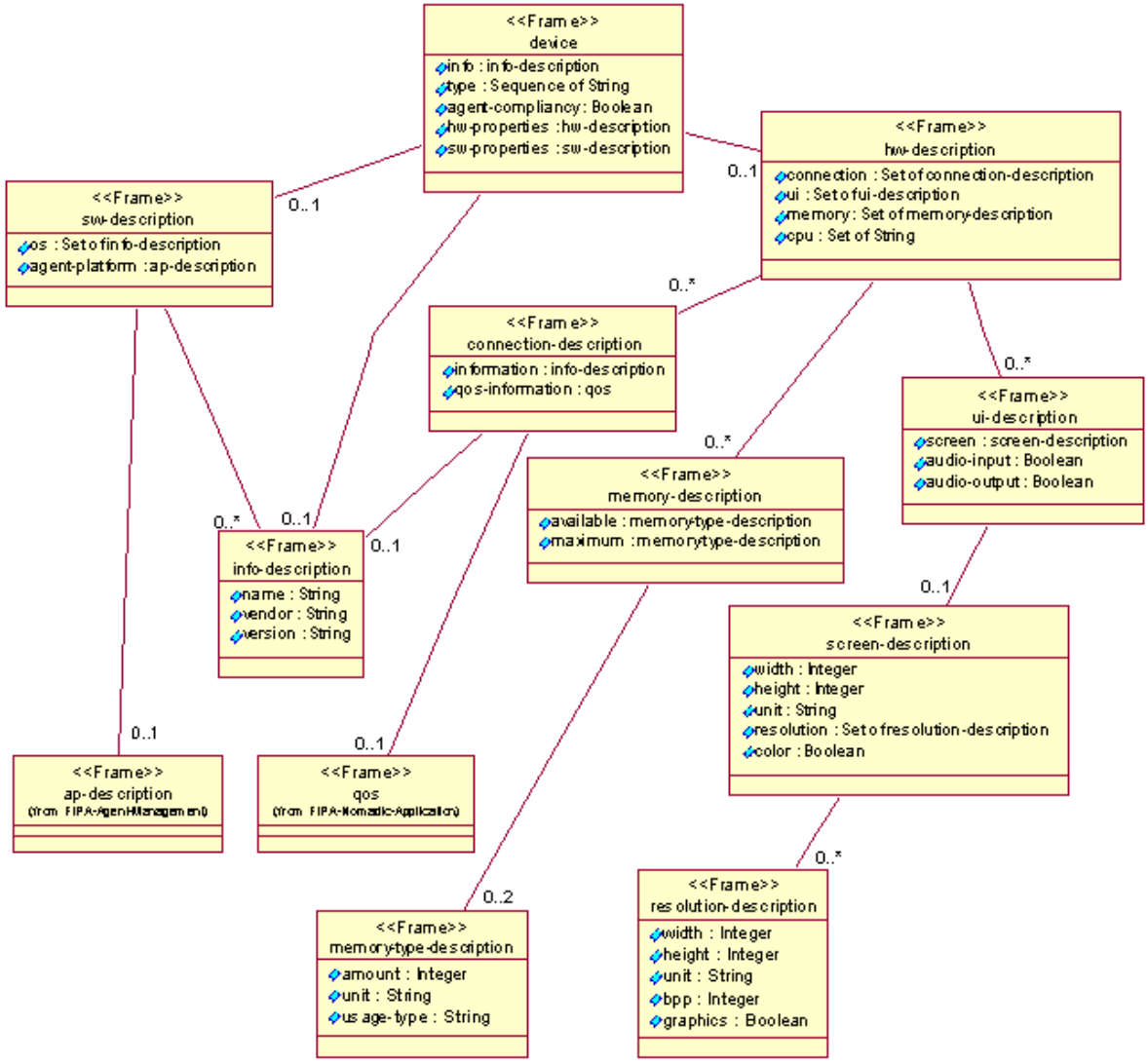
```
            <capability name="softkey_support" value="true"/>
        </group>
        <group id="markup">
            <capability name="wml_1_2" value="true"/>
            <capability name="wml_1_3" value="true"/>
            <capability name="preferred_markup" value="wml_1_3"/>
        </group>
        <group id="xhtml_ui">
            <capability name="xhtml_preferred_charset" value="iso8859"/>
        </group>
        <group id="display">
            <capability name="rows" value="4"/>
            <capability name="max_image_width" value="96"/>
            <capability name="resolution_height" value="65"/>
            <capability name="resolution_width" value="96"/>
            <capability name="max_image_height" value="45"/>
        </group>

        ...

    </device>
```

FIPA Device Ontology Specification

La FIPA (Foundation for Intelligent Physical Agents) és una organització internacional dedicada a desenvolupar especificacions per a la interoperabilitat entr agents intel·ligents.

L'any 2002 va especificar una ontologia de dispositius amb títol *FIPA Device Ontology Specification*, que podem consultar a la URL:

http://www.fipa.org/specs/fipa00091/



Il·lustració 1: Ontologia FIPA-Device

L'ontologia pretén descriure les característiques de dispositius amb l'objectiu de ser utilitzada per a que diferents dispositius connectats entre ells s'intercanviïn informació sobre els serveis que poden prestar o consumir.

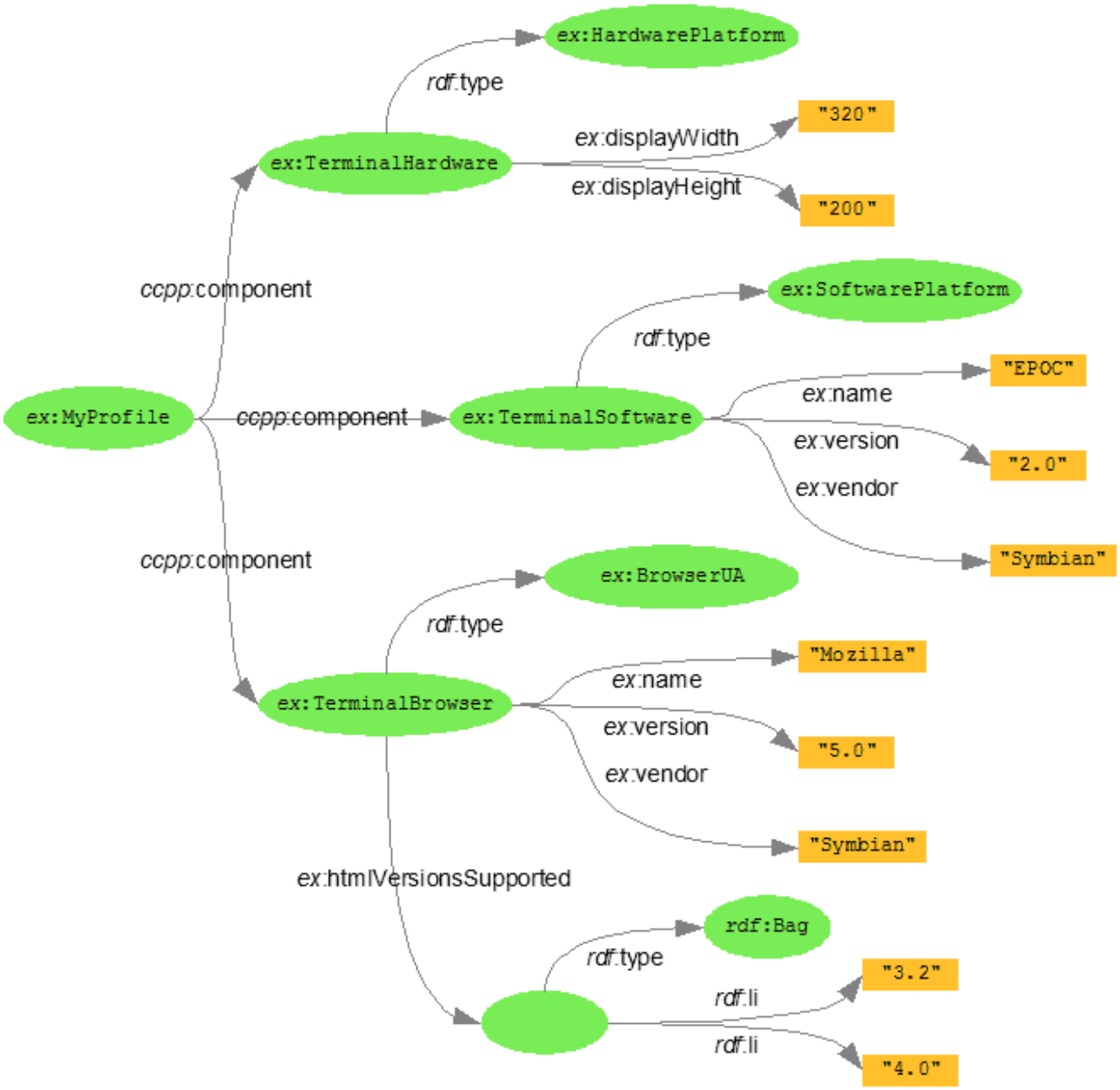
Composite Capability/Preference Profiles (CC/PP)

El CC/PP es tracta d'una descripció en RDF de capacitats de dispositius mòbils i preferències d'usuari, creada pel grup de treball Device Independence del World Wide Web Consortium.

http://www.w3.org/TR/CCPP-struct-vocab2/

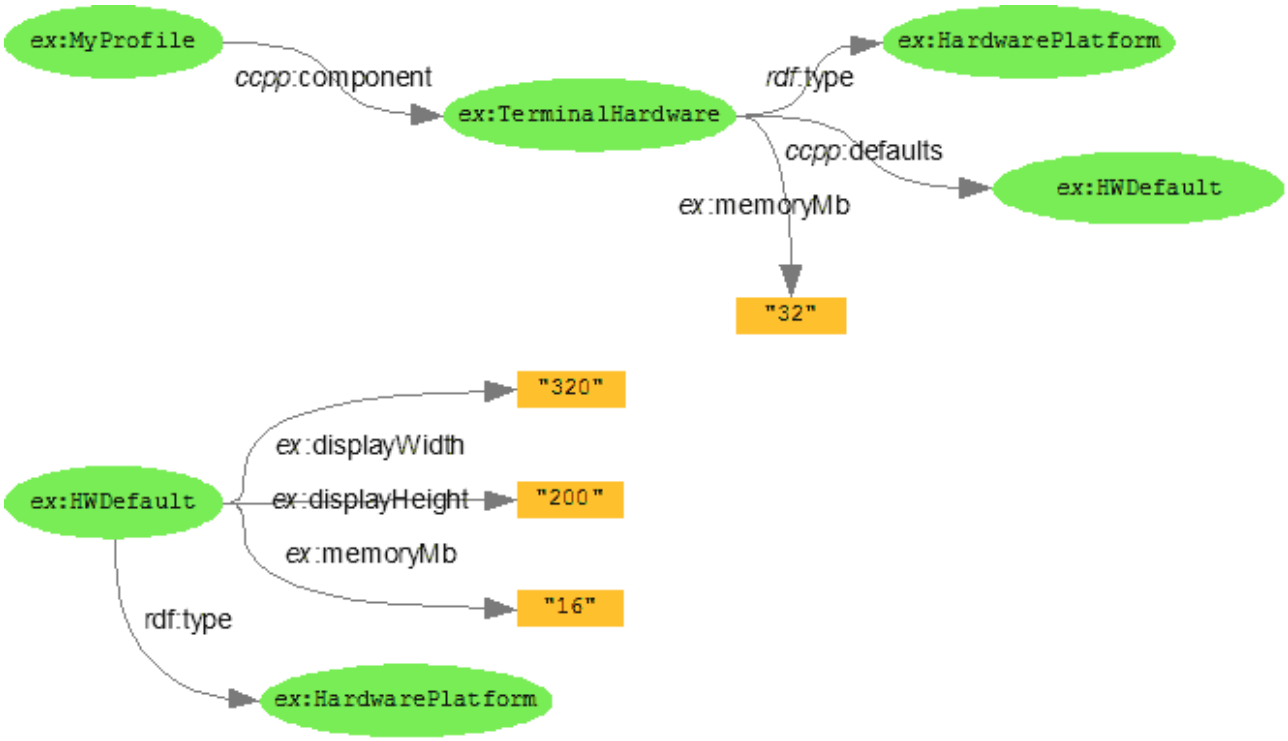
Un perfil CC/PP conté un o més components, i cada component conté un o més atributs. L'element `ccpp:component` és una propietat RDF que relaciona el perfil amb els seus components, que es representen amb recursos de tipus `ccpp:Component` o una subclasse de `ccpp:Component`. Els

atributs són propietats RDF en què el subjecte és un component i el predicat és el valor que pren la propietat, que pot ser un tipus de dades o un recurs, o un tipus compost `rdf:Bag` o `rdf:Seq` (explicat al capítol 2).



Il·lustració 2: Exemple de perfil CC/PP

Enlloc d'especificar els atributs d'un component es pot referenciar un recurs que contingui els valors per defecte, mitjançant l'element de propietat `ccpp:defaults`. Quan un atribut apareix directament en el component i també en el recurs referenciat per defecte, té preferència el primer.



Il·lustració 3: Valors per defecte en CC/PP

Mitjançant els espais de noms es pot estendre el vocabulari específic per a representar dispositius específics com memòries, dispositius de xarxa, d'imatge, etc.

User Agent Profile (*UAProf*)

UAProf és una especificació per a descriure dispositius mòbils creada pel *Wireless Application Protocol Forum* que té per objectiu que els dispositius mòbils siguin capaços de comunicar el seu perfil (descripció) als servidors a què es connecten de manera que aquests servidors puguin adaptar el contingut servit segons el perfil del dispositiu client. Utilitza el model de dades CC/PP, incorporant espais de noms addicionals com `prf` amb vocabulari específic.

http://www.uaprof.com

Un perfil *UAProf* es podrà representar per tant mitjançant un document RDF que seguirà l'esquema CC/PP. L'estructura del document serà la següent:

L'element `<rdf:RDF>` arrel contindrà un únic element `<rdf:Description>` amb l'atribut ID especificant el nom del perfil i un conjunt de 6 propietats `<prf:component>` amb els següents recursos com a valors:

- HardwarePlatform: Conté un conjunt de propietats relacionades amb el maquinari del dispositiu.
- SoftwarePlatform: Conté un conjunt de propietats relacionades amb el sistema operatiu i programari del dispositiu.
- BrowserUA: Conté un conjunt de propietats relacionades amb el navegador HTML del dispositiu.

- NetworkCharacteristics: Conté un conjunt de propietats relacionades amb la infraestructura de xarxa.
- WapCharacteristics: Conté un conjunt de propietats que descriuen les capacitats WAP del dispositiu.
- PushCharacteristics: Conté un conjunt de propietats que descriuen les capacitats Push suportades pel dispositiu.

Aquest vocabulari estàndard pot ser ampliat amb més tipus de components i més propietats. Les propietats poden ser simples cadenes de text o col·leccions utilitzant l'element `<rdf:Bag>`.

Posem com a exemple un fragment del perfil *UAProf* per al dispositiu Samsung SGH-I8320:

```
<?xml version="1.0" ?>
<!--
*************************************************************************
***************
*  SAMSUNG Mobile Handset UA Profile for SGH-I8320 (3G)
* Version:  1.0
* Created:  25/05/2009  by InBum Chang ( ibchang@samsung.com )
*
* Copyright (C) 2009 SAMSUNG Electronics. All rights reserved.
* Organization:  Linux Lab, Open OS S/W Group, SAMSUNG Electronics Co., LTD.
* Revision History
* - 1.0 : Created  ( 25/05/2009 )
*************************************************************************
***************
-->

<!DOCTYPE rdf:RDF [
      <!ENTITY prf-dt
'http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema-20030226#'>
]>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:mms="http://www.openmobilealliance.org/tech/profiles/MMS/ccppschema-
20050301-MMS1.2#"
xmlns:prf="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-
20030226#" xmlns:pss6="http://www.3gpp.org/profiles/PSS/ccppschema-PSS6#"
xmlns:prf-
dt="http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema-
20030226#">
<rdf:Description rdf:ID="I8320DeviceProfile">

      <!-- Hardware Platform Description -->
      <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type
rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppsche
ma-20030226#HardwarePlatform" />

            <!-- Vendor/model -->
            <prf:Vendor rdf:datatype="&prf-dt;Literal">SAMSUNG</prf:Vendor>
            <prf:Model rdf:datatype="&prf-dt;Literal">SGH-I8320</prf:Model>
            <prf:CPU rdf:datatype="&prf-dt;Literal">CORTEX-A8
OMAP3430</prf:CPU>
```

```
            <!-- Display -->
            <prf:ScreenSize rdf:datatype="&prf-
dt;Dimension">800x480</prf:ScreenSize>
            <prf:ColorCapable rdf:datatype="&prf-
dt;Boolean">Yes</prf:ColorCapable>
            <prf:BitsPerPixel rdf:datatype="&prf-
dt;Number">32</prf:BitsPerPixel>

      ...

<!-- Software Platform Description -->
      <prf:component>
      <rdf:Description rdf:ID="SoftwarePlatform">
      <rdf:type
rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppsche
ma-20030226#SoftwarePlatform" />

            <!--  Basic -->

            <!--  Java -->
            <prf:JavaEnabled rdf:datatype="&prf-
dt;Boolean">Yes</prf:JavaEnabled>
            <prf:JavaPlatform>
                  <rdf:Bag>
                        <rdf:li rdf:datatype="&prf-dt;Literal">CLDC
1.1</rdf:li>
                  </rdf:Bag>
            </prf:JavaPlatform>

      ....

mIO</rdf:Description>
</rdf:RDF>
```

Delivery Context

Podem consultar la documentació d'aquesta ontologia a

http://www.w3.org/2007/uwa/editors-drafts/DeliveryContextOntology/2007-08-31/DCOntology.html

Aquesta ontologia está especificada en llenguatge OWL i descriu les característiques d'un dispositiu que es connecta a la xarxa. L'ontologia conté informació sobre el maquinari, el programari i la xarxa. L'ontologia encara no està acabada i la darrera versió és de Juny de 2009.

La seva URI és la següent:

http://www.w3.org/2007/uwa/context/deliveryContext.owl

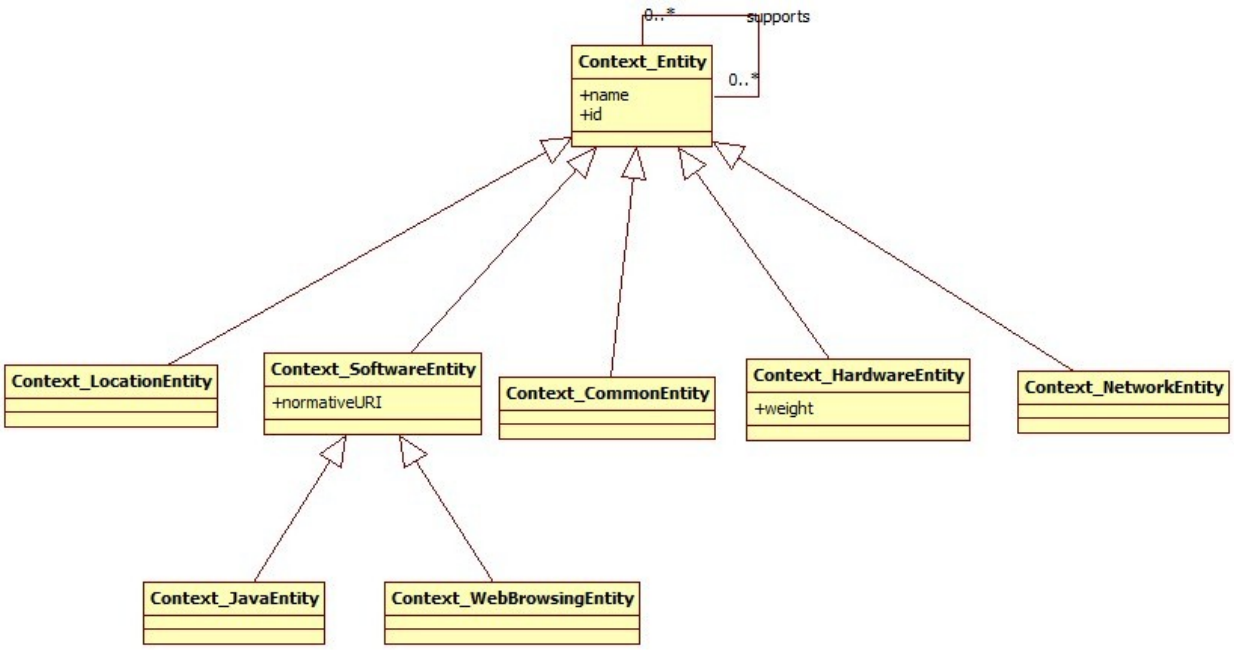
Les diferents classes que componen l'ontologia estan classificades en cinc categories diferents. Aquesta categorització es fa fent que les classes de les diferents categories són subclasse de les següents classes corresponents a cada categoria:

8. Context_CommonEntity
9. Context_HardwareEntity
10. Context_LocationEntity

11. Context_NetworkEntity
12. Context_SoftwareEntity

A més, el grup Context_SoftwareEntity també conté dos grups més:

- Context_WebBrowsingEntity
- Context_JavaEntity



Il·lustració 4: Categories de classes (Context)

A més, l'ontologia està organitzada en diferents espais de noms amb un prefix per a cada un. Són els següents, entre els quals n'hi ha un per a cada una de les categories llistades. Són els següents:

Prefix	Espai de noms	Contingut
`common`	`http://www.w3.org/2007/uwa/context/common.owl#`	Elements de propòsit general
`dcn`	`http://www.w3.org/2007/uwa/context/deliveryContext.owl#`	Elements principals del context
`hard`	`http://www.w3.org/2007/uwa/context/hardware.owl#`	Elements relacionats amb el maquinari
`java`	`http://www.w3.org/2007/uwa/context/java.owl#`	Elements relacionats amb Java
`loc`	`http://www.w3.org/2007/uwa/context/location.owl#`	Elements relacionats amb el posicionament
`net`	`http://www.w3.org/2007/uwa/context/network.owl#`	Elements relacionats amb la xarxa
`push`	`http://www.w3.org/2007/uwa/context/push.owl#`	Elements relacionats amb la tecnologia Push
`soft`	`http://www.w3.org/2007/uwa/context/software.owl#`	Elements relacionats amb el programari
`web`	`http://www.w3.org/2007/uwa/context/web.owl#`	Elements relacionats amb la tecnologia web

Unitats de mesura

Moltes de les propietats de dades en l'ontologia són valors de magnituds que tenen una unitat de mesura. En l'ontologia no està previst poder especificar les unitats de mesura, i cada propietat s'instancia amb un valor numèric que s'interpreta sempre de la mateixa manera, utilitzant una unitat de mesura fixada, que podem consultar a les anotacions de la propietat.

Descripció de classes i propietats

En la documentació de l'ontologia hi podem consultar amb detall totes les classes i propietats que conté.

De cada classe se'ns especifiquen:

- les seves superclasses
- les seves subclasses
- les propietats de les quals és domini
- les propietats en el rang de les quals es troba

De cada propietat se'ns especifiquen:

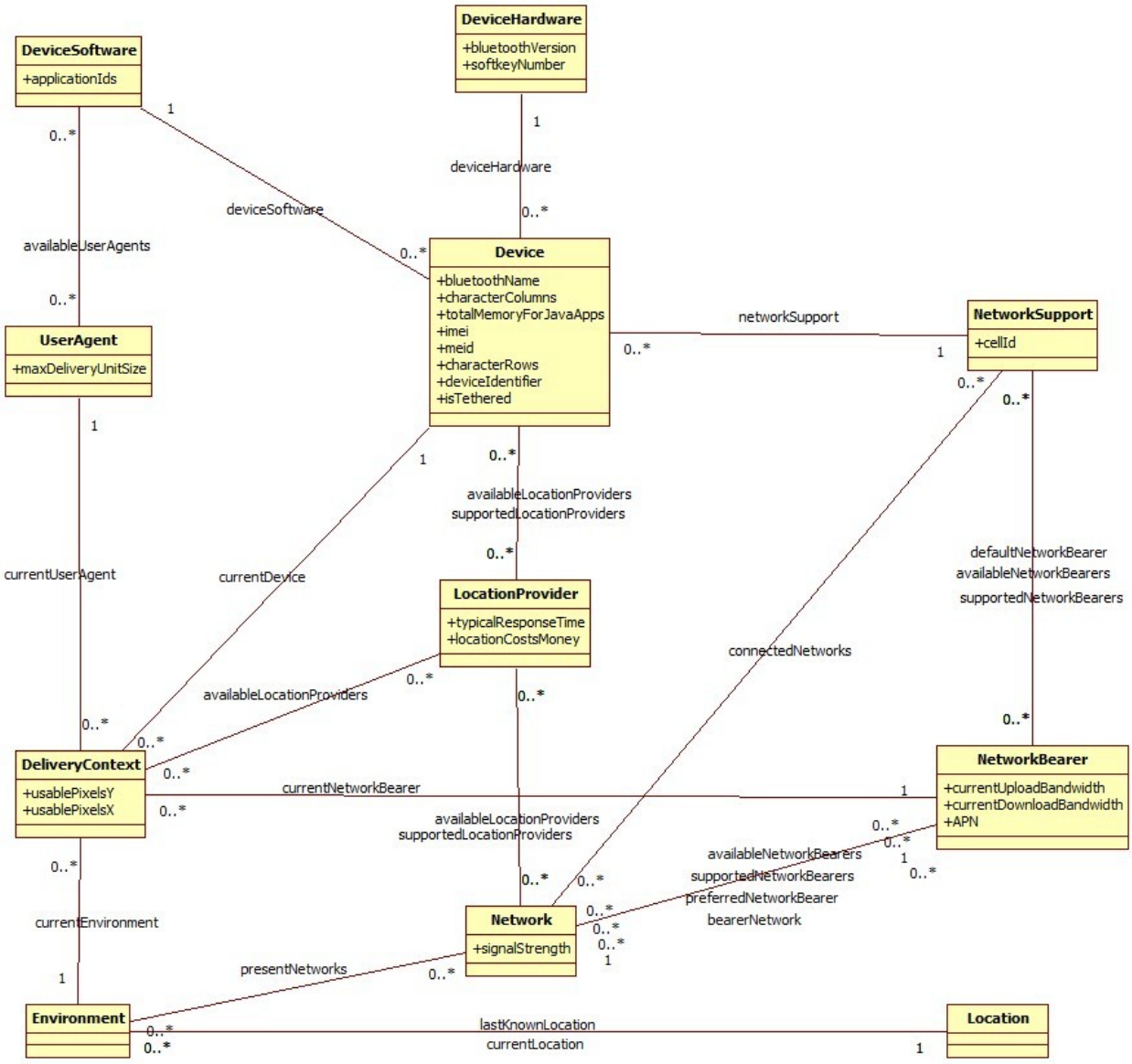
- les seves característiques: si és de dades o d'objectes, si és funcional, transitiva, etc.
- les seves superpropietats
- les seves subpropietats
- el seu domini
- el seu rang

No pretenem aquí descriure amb detall l'ontologia, sinó explicar les classes i propietats més significatives i donar una visó general i entenedora de l'ontologia.

Delivery Context

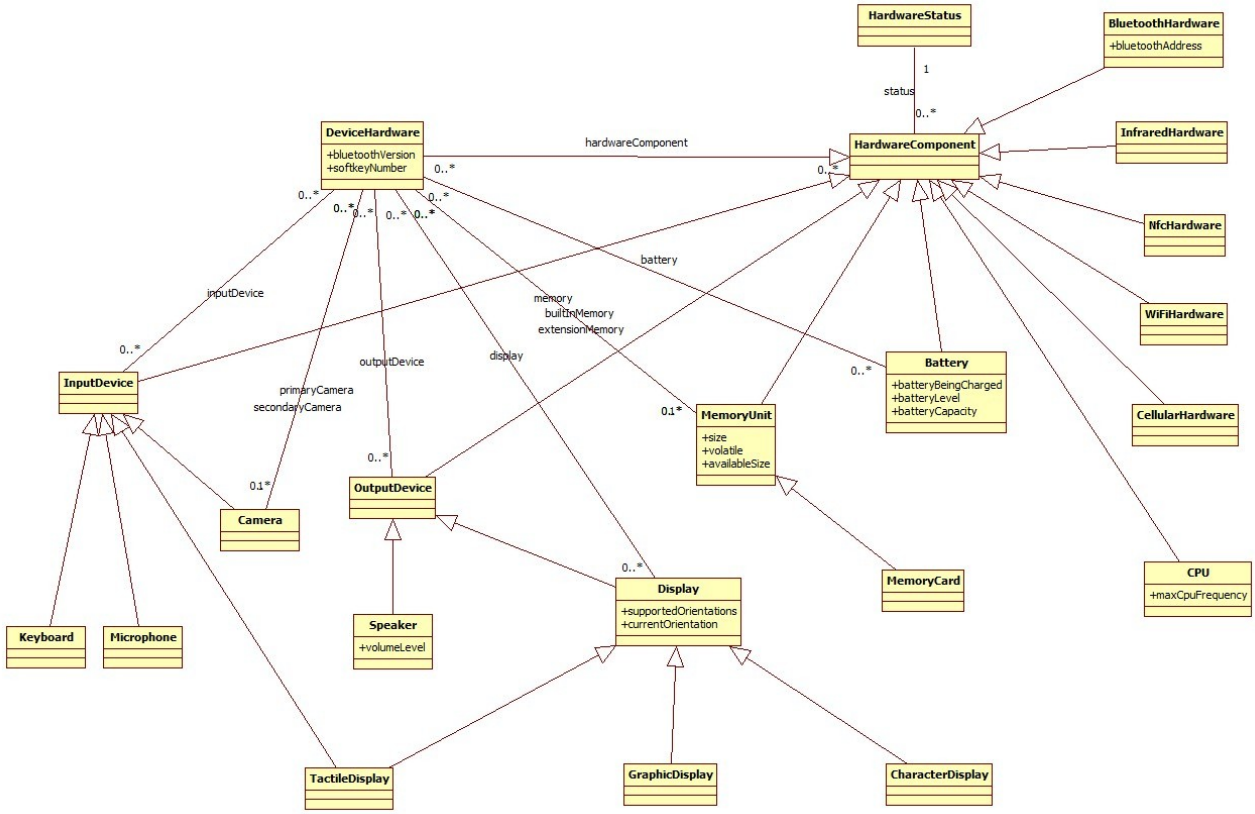
Aquesta ontologia es va dissenyar per a que els dispositius mòbils informessin en tot moment del context en el qual es produeix la comunicació. Aquest context inclou:

- el maquinari del dispositiu, els tipus de xarxes i sistemes de localització que suporta i el programari que té
- la xarxa a través de la qual s'estableix la comunicació
- l'agent d'usuari utilitzat en la comunicació, per exemple un navegador
- el sistema de localització del dispositiu, com per exemple GPS
- l'entorn on es troba el dispositiu, que inclou la seva localització i les xarxes disponibles


DeviceHardware
+bluetoothVersion
+softkeyNumber
DeviceSoftware
+applicationIds
Device
+bluetoothName
+characterColumns
+totalMemoryForJavaApps
+imei
+meid
+characterRows
+deviceIdentifier
+isTethered
NetworkSupport
+cellId
UserAgent
+maxDeliveryUnitSize
LocationProvider
+typicalResponseTime
+locationCostsMoney
DeliveryContext
+usablePixelsY
+usablePixelsX
NetworkBearer
+currentUploadBandwidth
+currentDownloadBandwidth
+APN
Network
+signalStrength
Environment
Location
deviceHardware
deviceSoftware
availableUserAgents
networkSupport
availableLocationProviders
supportedLocationProviders
currentUserAgent
currentDevice
defaultNetworkBearer
availableNetworkBearers
supportedNetworkBearers
connectedNetworks
currentNetworkBearer
currentEnvironment
presentNetworks
preferredNetworkBearer
bearerNetwork
lastKnownLocation
currentLocation

Maquinari

La classe *DeviceHardware* defineix el maquinari del dispositiu. Les seves propietats tenen com a rangs objectes que són subclasses de la classe *HardwareComponent*.



Il·lustració 5: Diagrama de classes: maquinari

Xarxa

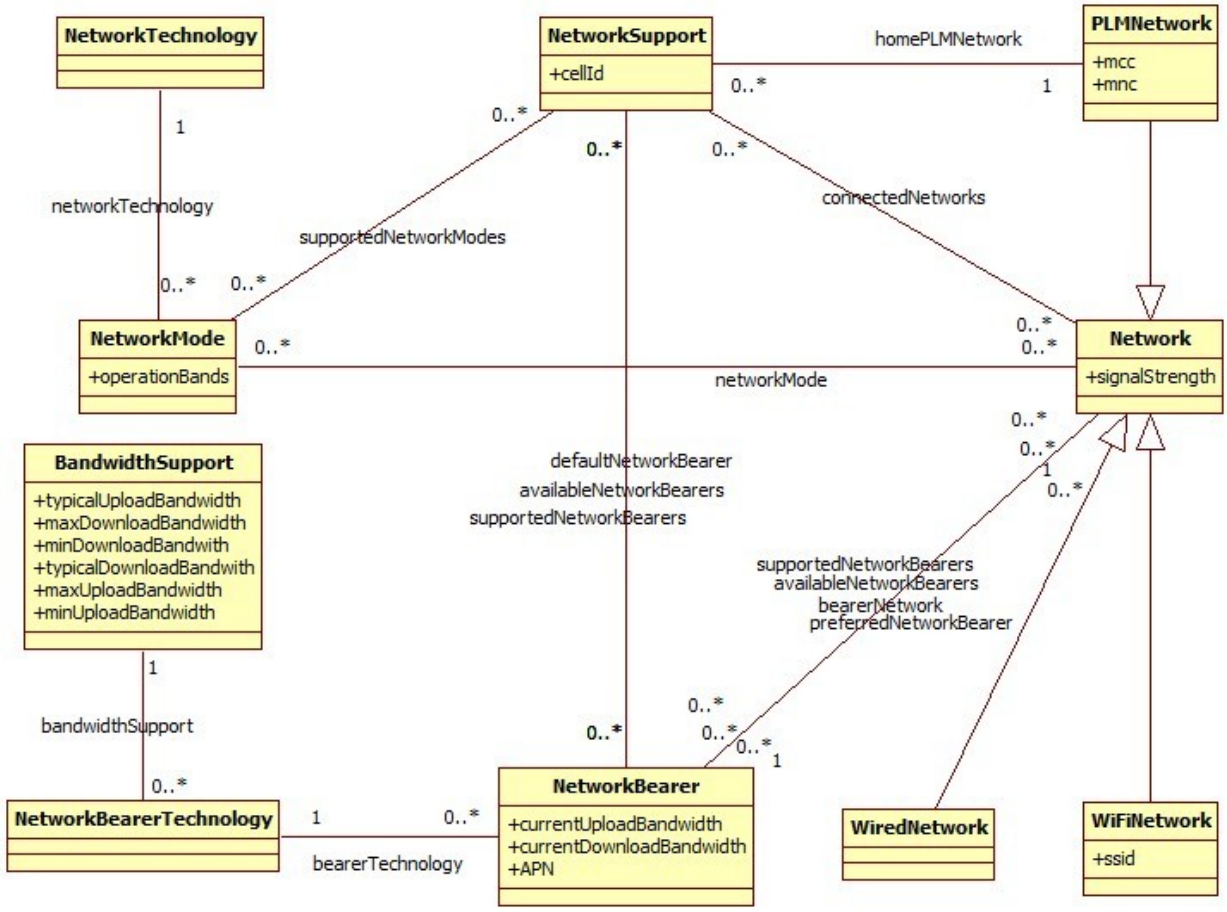
La classe *NetworkSupport* representa tots els tipus de xarxa a què es pot connectar el dispositiu, tots els serveis que pot utilitzar, tots els modes de connexió que suporta i les xarxes a què està connectat.

La classe *NetworkTecnlology* representa els tipus de xarxes, com per exemple GSM o UMTS, i la classe *NetworkMode* representa els modes de funcionament d'una xarxa, com per exemple les diferents bandes de freqüència en què pot operar.

La classe *NetworkBearer* representa un servei de comunicació de dades, i la classe *NetworkBearerTechnology* representa les tecnologies de serveis de comunicació de dades, com per exemple GPRS o HDSPA.

La classe *Network* representa una xarxa concreta a la qual es pot connectar un dispositiu i que suporta diferents modes de comunicació i diferents serveis de comunicacions. L'ontologia distingeix tres tipus de xarxes que modela en tres subclasses:

- *PLMNetwork*: xarxa mòbil terrestre pública
- *WiredNetwork*: xarxa cablejada
- *WifiNetwork*: xarxa sense fils

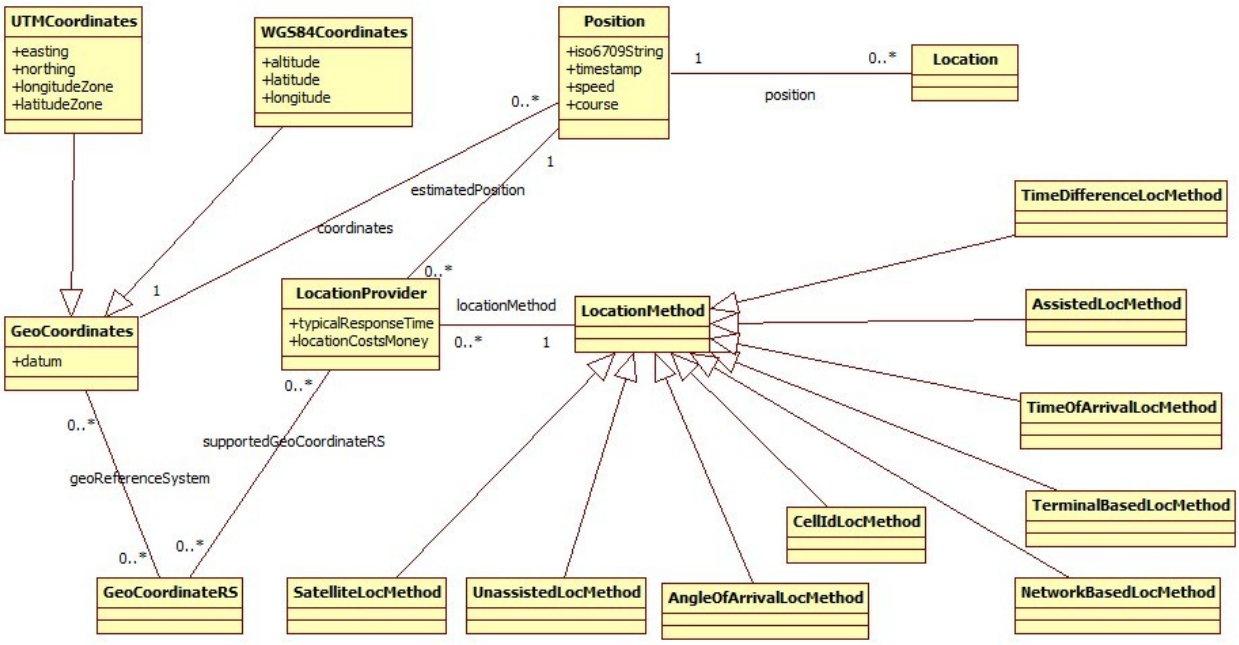


Localització

La classe *LocationProvider* representa un proveïdor de localització, el qual utilitza un mètode de localització representat en la classe *LocationMethod*, la qual pot ser de nou tipus diferents com es veu en la figura. Alguns dels tipus són disjunts entre ells, com per exemple *AssistedLocMethod* i *UnassistedLocMethod*.

La classe Position representa la posició d'un dispositiu en instant de temps donat, i altres propietats com la seva velocitat per exemple.

L'ontologia permet representar diferents tipus de georeferència i de coordenades mitjançant les classes *GeoCoordinateRS* i *GeoCoordinates* respectivament. Aquesta darrera té les dues subclasses *UTMCoordinates* i *WGS84Coordinates*, que permeten representar les coordenades de tipus UTM i WGS84 respectivament.

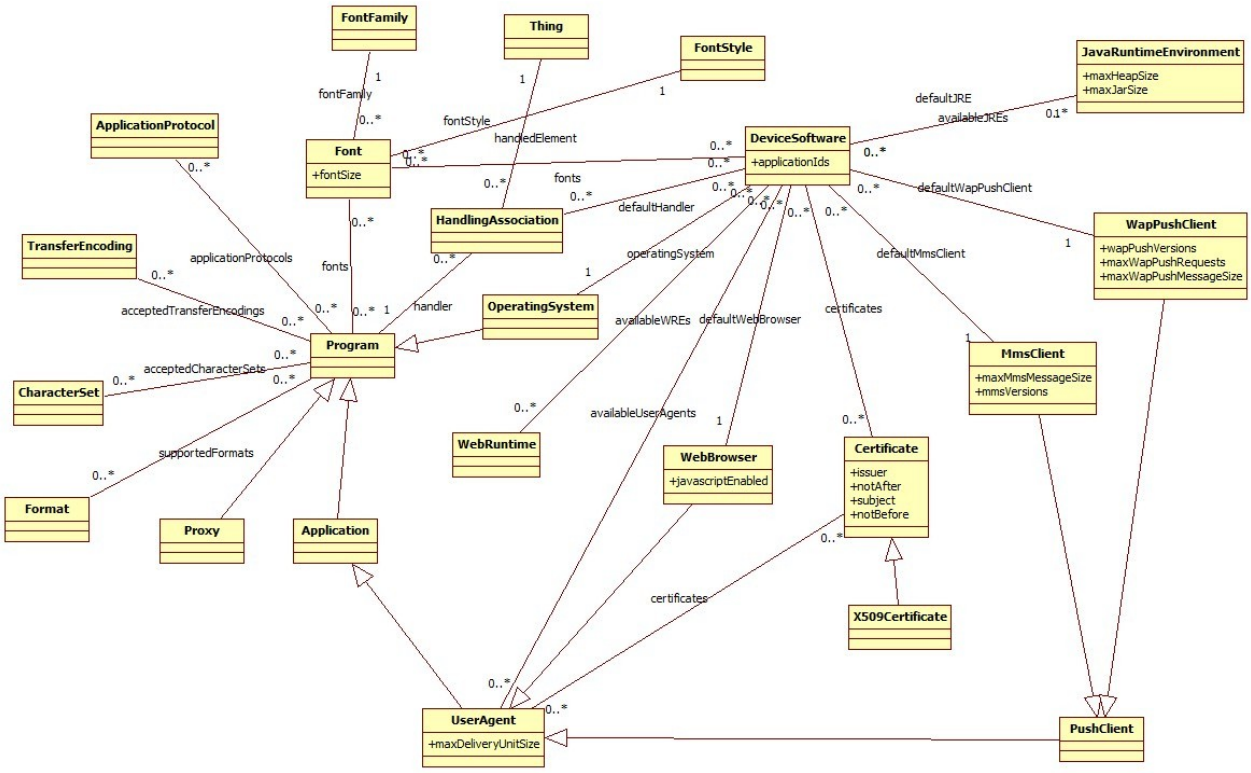


Programari

La classe *DeviceSoftware* representa tot el programari present en el dispositiu, i mitjançant diferents propietats es relaciona amb classes que representen diferents elements de programari. Són les següents:

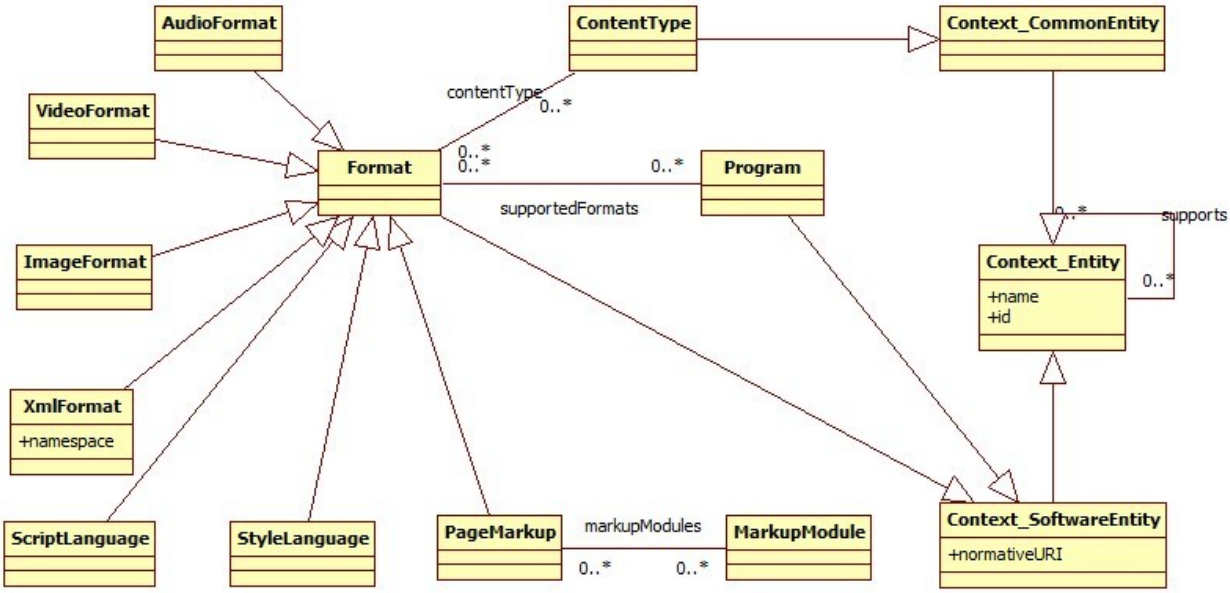
- *OperatingSystem*, que representa un sistema operatiu
- *UserAgent*, que representa un agent d'usuari. Té les següents subclasses:
  - *WebBrowser*, que representa un navegador Web i que és subclasse de UserAgent
  - *MmsClient*, que representa un client de missatges multimèdia
  - *WapPushClient*, que representa un client WAP Push
- *JavaRunTimeEnvironment*, que representa un entorn d'execució de Java
- *Certificate*, que representa un certificat per a xifratge asimètric
- *Font*, que representa un tipus de lletra
- *HandlingAssotiation*: Representa una relació en què un programa (*handler*) és capaç de manegar alguna cosa del tipus que sigui (*handledElement*)

UserAgent i *OperatingSystem* són subclasses de *Program*, que representa un programa de qualsevol tipus. Aquesta classe té diferents propietats que la relacionen amb altres classes que representen formats suportats (*Format*), protocols (*ApplicationProtocol*), jocs de caràcters acceptats (*CharacterSet*), tipus de lletra (*Font*), codificacions per a la transferència (*TransferEncoding*).



Formats

La classe *Program* té la propietat *supportedFormats* que la relaciona amb la classe *Format*, que representa un format. La classe format té diferents subclasses que representen els diferents tipus de formats que pot haver-hi: *AudioFormat*, *VideoFormat*, *ImageFormat*, *XMLFormat*, *ScriptLanguage*, *StyleLanguage*, *PageMarkup*.



Altres propietats

L'ontologia defineix un conjunt de propietats sense definir-ne el rang ni el domini, ja que es considera que es poden aplicar en diferents contextos. Destaquem com a molt freqüèntment

utilitzades les propietats *name*, *model* i *version* per a identificar elements de tot tipus.

Instàncies

L'ontologia *DeliveryContext* ve amb algunes instàncies d'exemple, i moltes instàncies útils que representen formats d'imatges, tipus de xarxes, sistemes de localització, jocs de caràcters, idiomes. No es tracta d'una instanciació molt completa però és útil per a comprendre el funcionament de l'ontologia.

mIO! Ontology Network

La xarxa d'ontologies mIO! està desenvolupada pel Ontology Engineering Group, i la podem descarregar en la seva web:

http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/82-mio-ontologies

Com bé indiquen en la web, l'objectiu de la xarxa d'ontologies mIO! és representar el coneixement relacionat amb el context d'un usuari. Aquest inclou informació tan variada com el lloc on es troba, la temperatura, els seus gustos i preferències, i també informació sobre el dispositiu que utilitza, la xarxa a què està connectat i els serveis que pot consumir o produir.

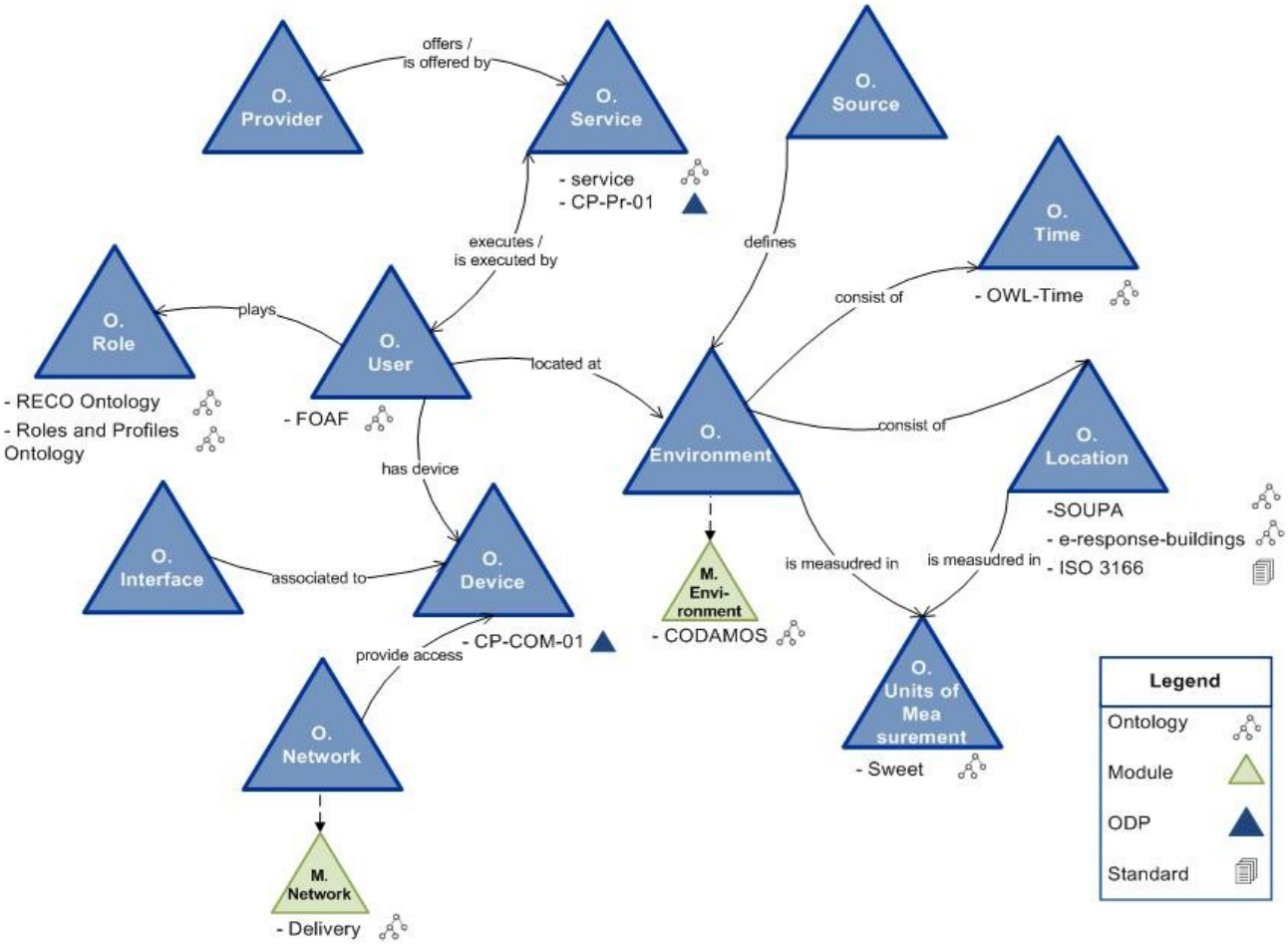
Cal dir que segons la informació que podem trobar a la web citada més amunt, l'ontologia es troba encara en desenvolupament i estarà finalitzada a finals de 2010. Hi trobem un enllaç per descarregar la darrera versió, de la qual en sabem que es tracta de la versió 3 però no en sabem la data. Com que ja estem a 2011 suposem que aquesta no és encara la definitiva i que la pàgina web no està actualitzada.

Fent una mica més de recerca per Internet trobem l'article següent

http://www.sciencedaily.com/releases/2011/05/110517091630.htm

on es diu que l'ontologia va ser presentada al CIAO Workshop de la conferència EKAW 2010 celebrada a Portugal.

Hi ha una ontologia central que relaciona entre sí un conjunt d'ontologies cada una representant el coneixement d'un subdomini concret.



Il·lustració 6: mIO! ontology network

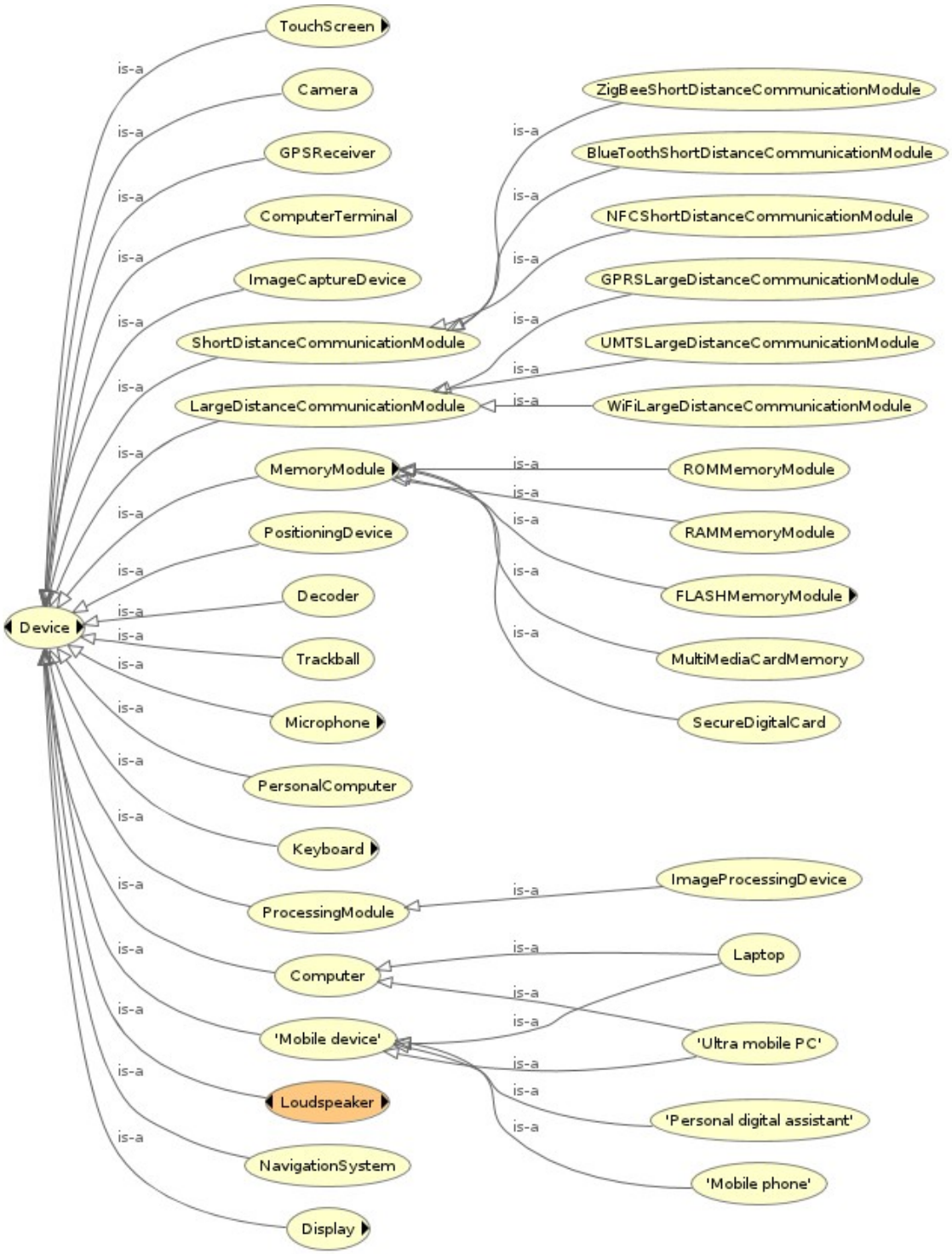
En aquest projecte ens fixarem en les ontologies Device, Network i Interface, que estan relacionades amb l'ontologia de dispositius mòbils que volem desenvolupar.

Device

Il·lustració 7: mIO! Device: Jerarquia de classes

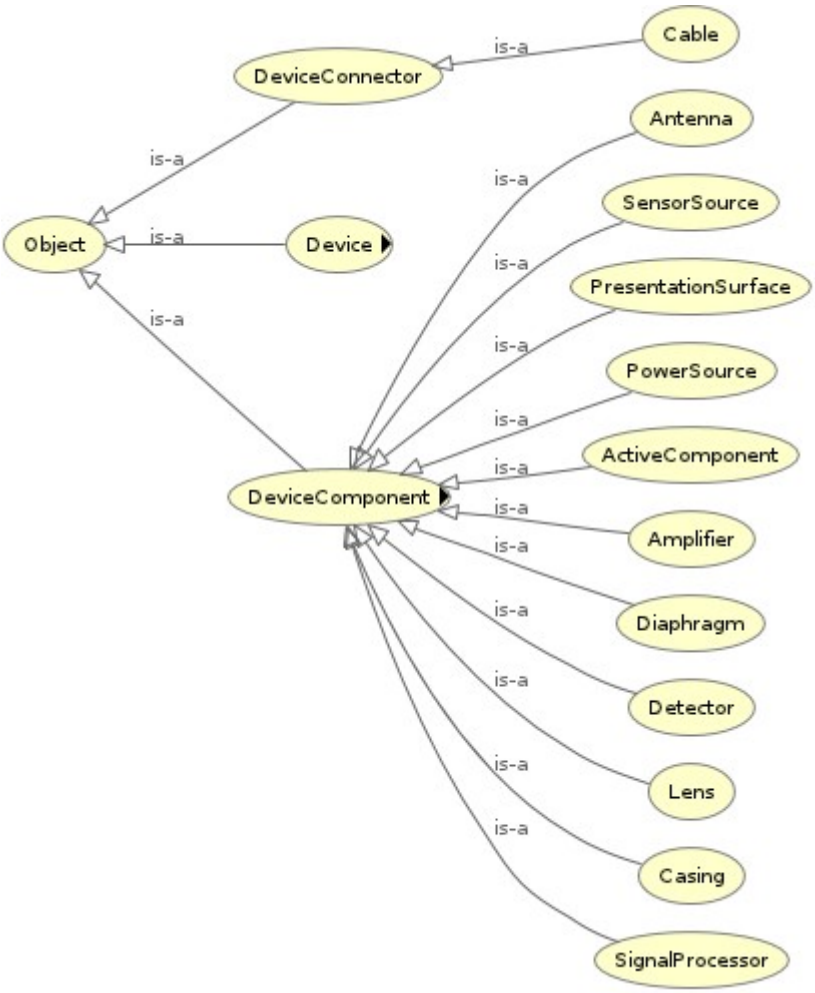
Aquesta ontologia descriu un ampli ventall de dispositius i característiques, i ha estat dissenyada utilitzant el patró *Componency*, que es basa en que un objecte forma part d'altres objectes o està format per altres objectes. Els dispositius han estat classificats en categories utilitzant el i el patró *Taxonomy*.

En la figura següent mostrem les subclasses de la classe *Device*.



Il·lustració 8: mIO! Device: Subclasses de la classe Device

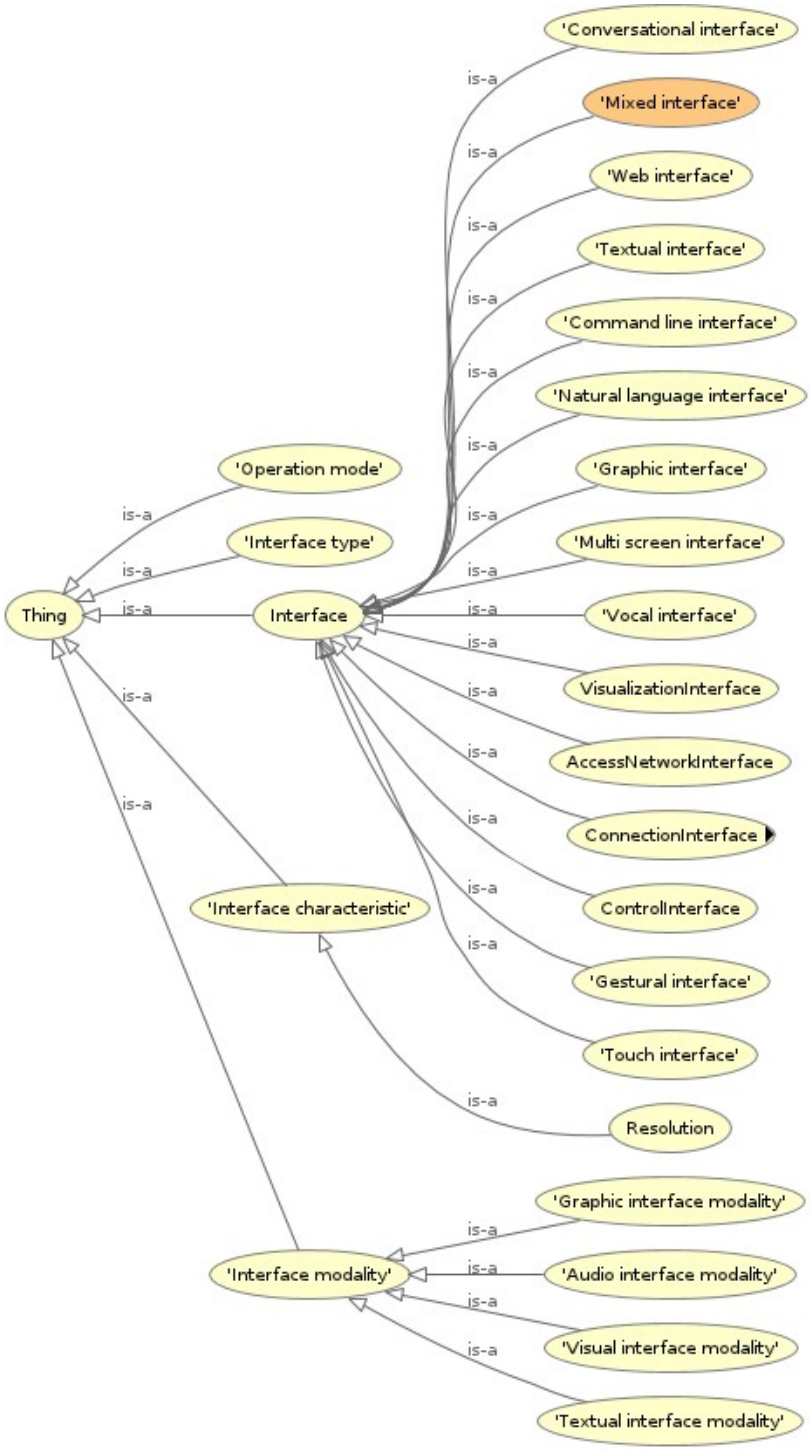
A continuació mostrem les subclasses de la classe *Object*.



Il·lustració 9: mIO! Device: Subclasses de la classe Object

Interface

Aquesta ontologia descriu les interfícies d'usuari, els seus tipus i característiques, distingint entre interfícies d'entrada i interfícies de sortida.



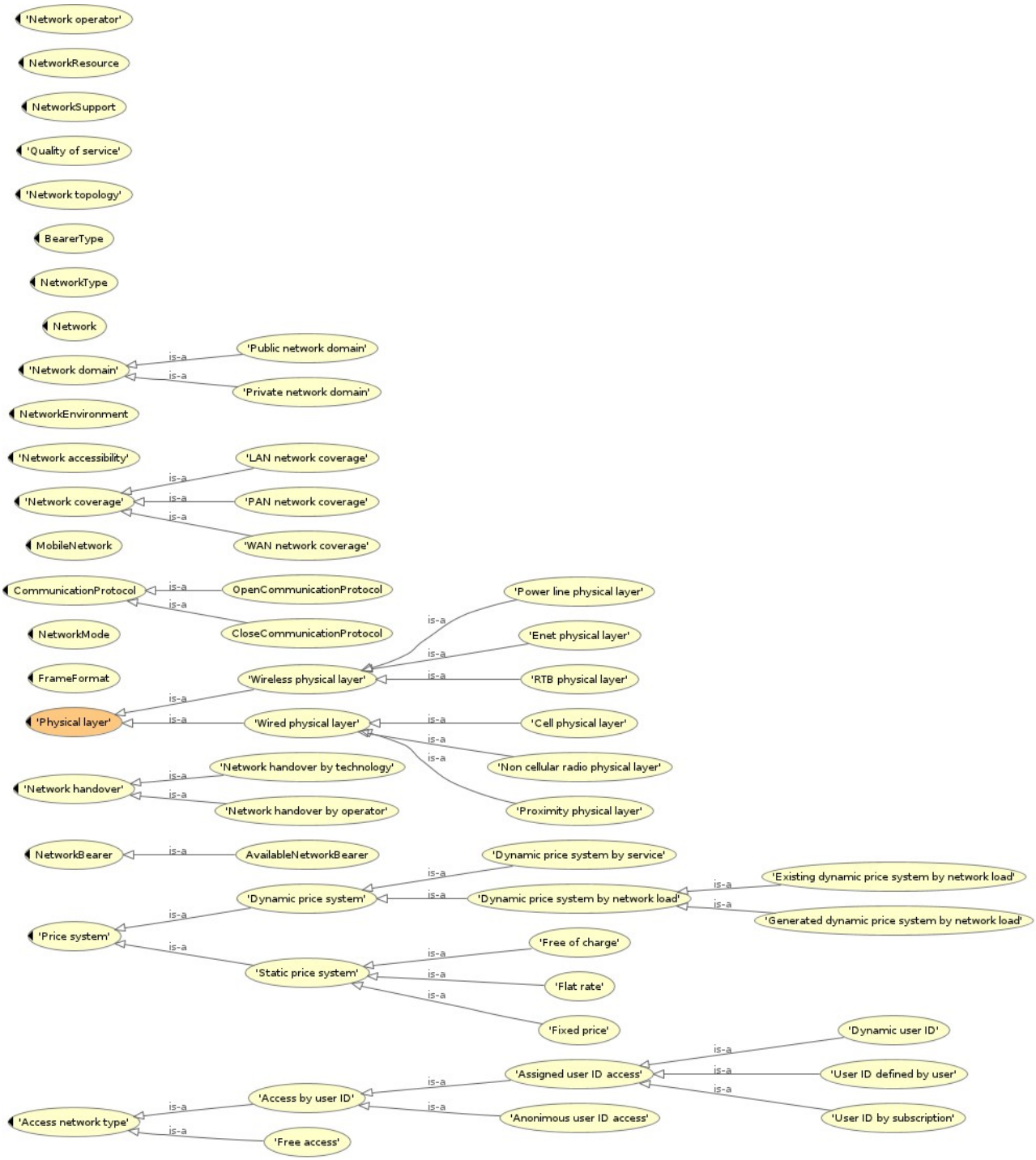
Il·lustració 10: mIO! Interface: Jerarquia de classes

Tot seguit mostrem les subclasses de la classe *ConnectionInterface*.

Il·lustració 11: mIO! Interface: Subclasses de la classe ConnectionInterface

Network

Aquesta ontologia modela el coneixement sobre xarxes de comunicacions, operadors, preus, cobertura, etc. Està basada en una part de l'ontologia *Delivery Context*.



Il·lustració 12: miO! Network: Jerarquia de classes

Per a que el gràfic sigui més clar, hem omès la classe Thing, que és la superclasse de tots els objectes de la columna de l'esquerra.

3.2 Adaptació i instanciació d'una ontologia

Ja que tenim disponibles ontologies de dispositius mòbils, no farem el disseny de l'ontologia des de zero. Tenim a la nostra disposició dues ontologies: DeliveryContext i mIO!. Anem a veure els pros i contres de cada una d'elles:

- mIO! és més actual i més completa
- mIO! té un abast molt més gran que el que necessitem per a aquest projecte
- DeliveryContext s'ajusta força a les dades que volem descriure
- DeliveryContext està ben documentada
- DeliveryContext duu algunes instàncies d'exemple i bastantes instàncies d'utilitat

Agafarem com a punt de partida l'ontologia DeliveryContext, ja que amb ella podem emmagatzemar les dades necessàries i en disposem molta més informació, tot i no ser tant completa com la mIO!.

Modificacions

Després de l'estudi detallat de l'ontologia observem que s'hi poden afegir algunes modificacions per a millorar-la. Presentarem l'ontologia amb les modificacions realitzades.

Dispositius de maquinari

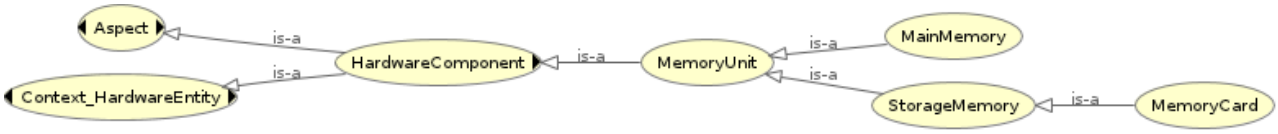
Els dispositius mòbils actuals incorporen alguns dispositius que no estan representats en l'ontologia. Són els següents:

- Sensors diversos. Actualment, molts dispositius incorporen brúixoles, giroscopis, acceleròmetres, sensors de llum i sensors de proximitat. Els representem amb la classe Sensor que és subclasse de la classe *HardwareComponent*.
- Sintonitzador de radio. El representem amb la classe Radio que és també subclasse de la classe *HardwareComponent*.

Les càmeres de fotografia i video que incorporen els dispositius mòbils tenen actualment unes molt bones prestacions que són tingudes en compte pels usuaris. Afegim les següents propietats relacionades amb la càmera

- *cameraAperture* que representa l'obertura del diafragma
- *cameraFlash* que representa la presència o no de flash
- *cameraFocusDistance* que representa la distància focal
- *cameraOpticalZoom* que representa el zoom òptic
- *cameraSensitivity* que representa la sensibilitat ISO
- *cameraShutterSpeed* que representa la velocitat del disparador

L'ontologia diferencia entre memòria integrada en el dispositiu i memòria en targes de memòria extraïbles. Creiem necessari també distingir entre memòria de treball i memòria d'emmagatzematge. La classe *MemoryUnit* i les seves subclasses:



Il·lustració 13: Jerarquia de classes per a representar la memòria

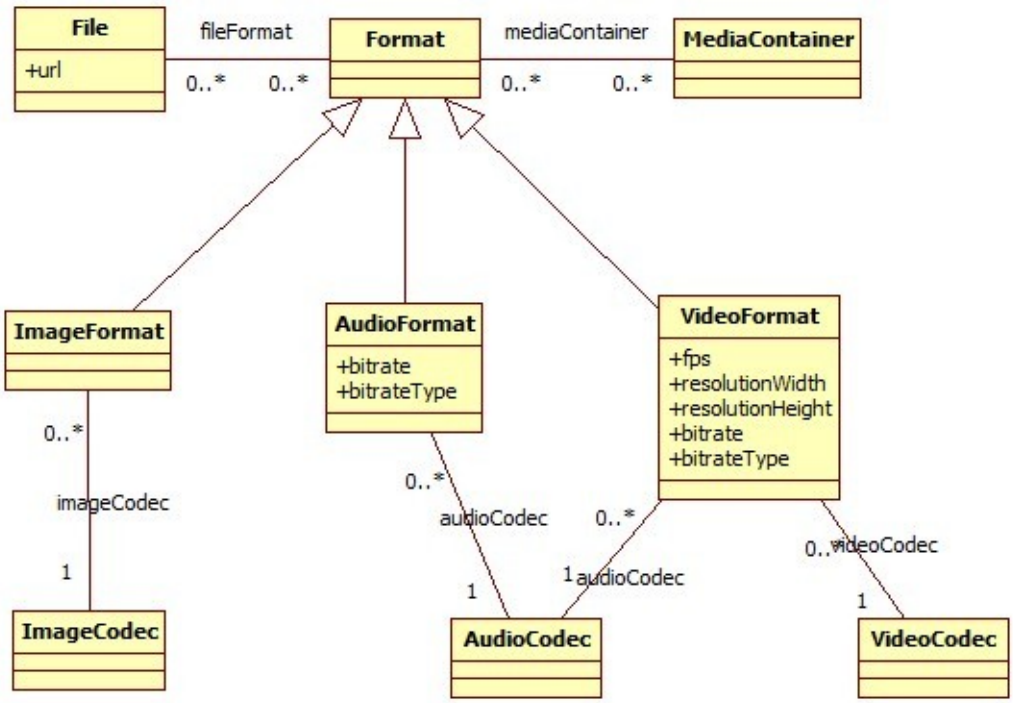
Formats d'imatge, so i video

L'ontologia *DeliveryContext* representa els formats amb la classe *Format* i les seves subclasses per a cada tipus de format. Per exemple, els tipus d'imatges els representa amb objectes de la classe *ImageFormat*. Així, l'objecte jpeg de la classe *ImageFormat* representaria el format jpg per a imatges.

Aquesta forma de representar formats és vàlida quan els fitxers poden contenir un únic format. No és vàlid, però, per a representar fitxers d'audio i de video.

Els formats d'àudio tenen d'una banda el contenidor, i d'altra banda el codec utilitzat. Per exemple, un fitxer pot tenir el contenidor WAV i el codec LPCM.

Els formats de video són encara més complexos. Tenen d'una banda el contenidor, com per exemple AVI, i d'altra banda el codec de video i el codec d'audio. Així doncs, un contenidor AVI podria tenir com a codec de video MPEG-4 i com a codec d'audio MP3.



Il·lustració 14: Diagrama de classes: Formats

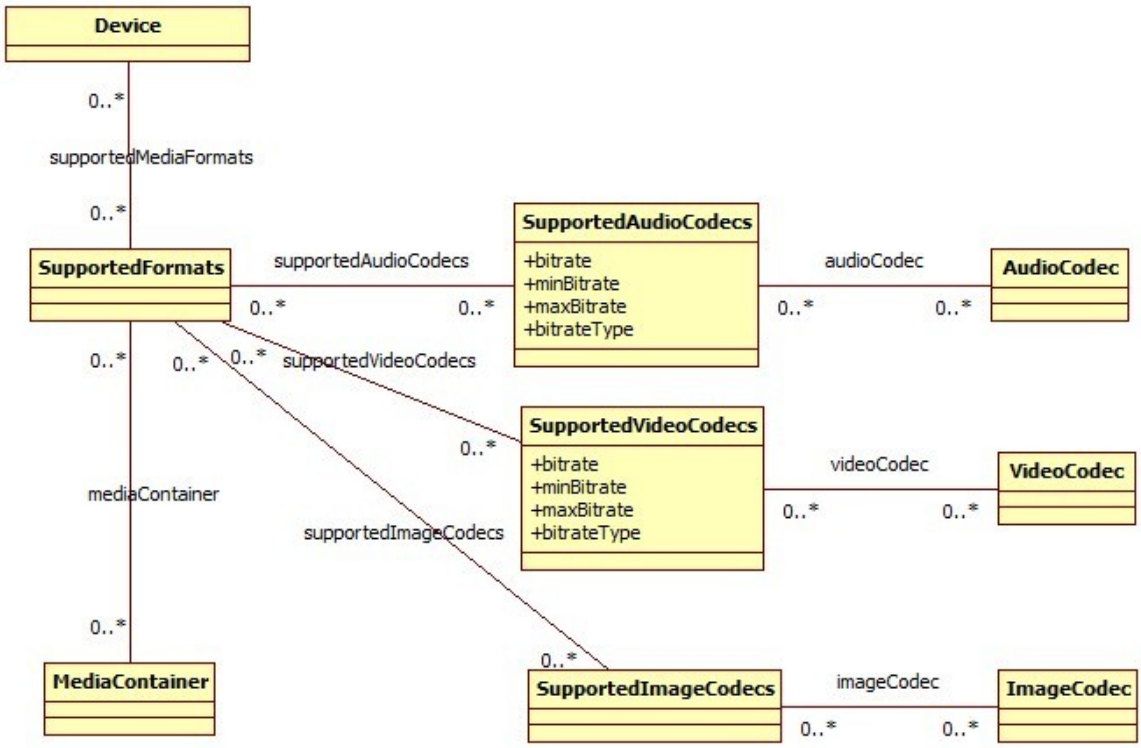
Pel que fa al suport d'audio i de video, cada sistema operatiu o dispositiu suporta unes combinacions de contenidors i codecs diferents. En alguns casos, els codecs són suportats amb algunes limitacions, com per exemple en el màxim bitrate.

Per tant hem fet algunes modificacions per a poder representar formats i el seu de cada dispositiu o sistema operatiu dels diferents contenidors, i per a cada contenidor quins codecs suporta.

Un altre tipus de fitxer que volem representar és un fitxer executable. En aquest cas cal representar quina és la seva plataforma d'execució. La plataforma d'execució la representem mitjançant la propietat platform. També representarem quins dispositius de maquinari es requereixen per a la seva execució.

Il·lustració 15: Diagrama de classes: fitxers executables

Mitjançant regles podrem determinar si un determinat sistema operatiu és capaç de suportar un fitxer determinat a partir de les seves característiques (plataforma, contenidor, codec). En cas afirmatiu s'inferirà la propietat *supports* amb origen en l'objecte que representa el sistema operatiu i amb destí el fitxer.



Il·lustració 16: Diagrama de classes: formats suportats

Programari

Hem afegit la propietat *supportedLanguage* que té com a domini la classe *Program* i com a rang la classe *Language*.

Les noves versions dels sistemes operatius solen tenir les mateixes característiques que les anteriors amb afegits de nova funcionalitat i suport. Inspirant-nos una mica en com modela aquest fet el repositori WURFL, hem relacionat les versions de sistemes operatius de manera que el suport de formats en les versions superiors s'infereixi a partir de les propietats instanciades per a versions anteriors. Per a això hem fet les següents modificacions:

- Hem afegit propietats noves per a relacionar versions de sistemes operatius: *earlierVersion*, *laterVersion*, *nextVersion* i *previousVersion*. Les propietats *nextVersion* i *previousVersion* són subpropietas de *laterVersion* i *earlierVersion* respectivament.
- Hem creat regles per a inferir propietats de sistemes operatius més nous en funció de les propietats dels sistemes operatius anteriors i no haver d'instanciar-les.
- Pel que fa als fitxers executables, també tindrem una regla que en cas de poder executar-se en un sistema operatiu determinat, també podrà executar-se en els sistemes operatius de versions posteriors.

Modelarem el fet que si un S.O. suporta determinats formats, aleshores el dispositiu que utilitza aquest sistema operatiu també els suporta. Això ho farem amb una altra regla.

Altres

Substituir el tipus *anySimpleDataType* per *Literal*, ja que el raonador *Pellet* no el reconeix.

La propietat *supports* pot ser transitiva. Per exemple, si un sistema operatiu suporta un format que suporta un *codec*, podem dir que aquest *codec* és suportat pel sistema operatiu.

Ús de l'ontologia

L'ontologia *Delivery Context* és molt completa. En aquest treball ens interessa descriure el maquinari dels dispositius, el seu sistema operatiu i el suport de fitxers. Per tant, totes aquelles classes de l'ontologia que facin referència al context de la comunicació i l'entorn no seran utilitzades.

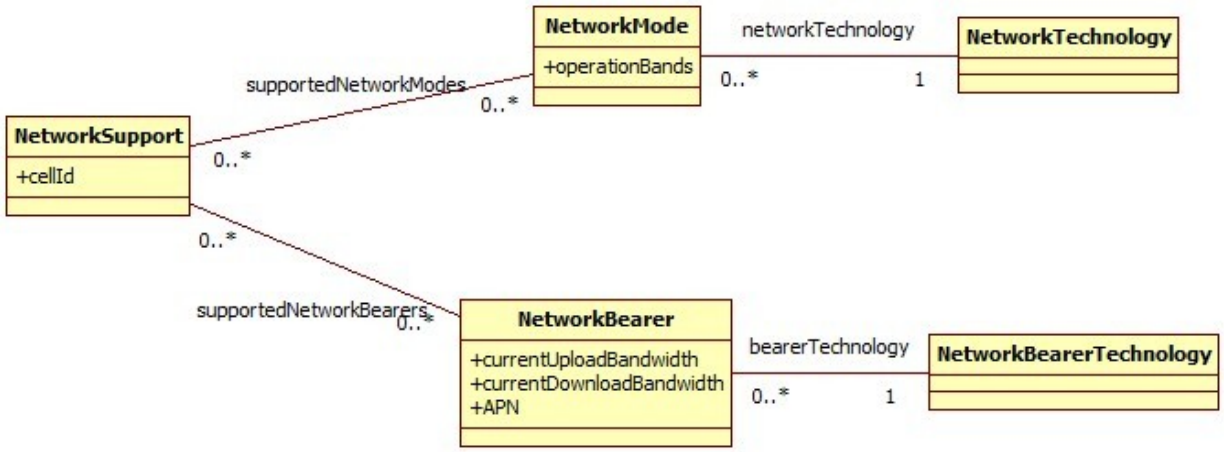
A continuació presentarem en forma de diagrames les classes utilitzades en cada una de les parts de l'ontologia. Com que ja les hem explicat en descriure l'ontologia *Delivery Context* només realitzarem alguns comentaris pertinents.

Maquinari

D'aquesta part no descartarem res, i per tant el diagrama de classes serà el de la il·lustració 5, amb l'afegit de classes per a representar sensors i sintonitzador de radio.

Xarxa

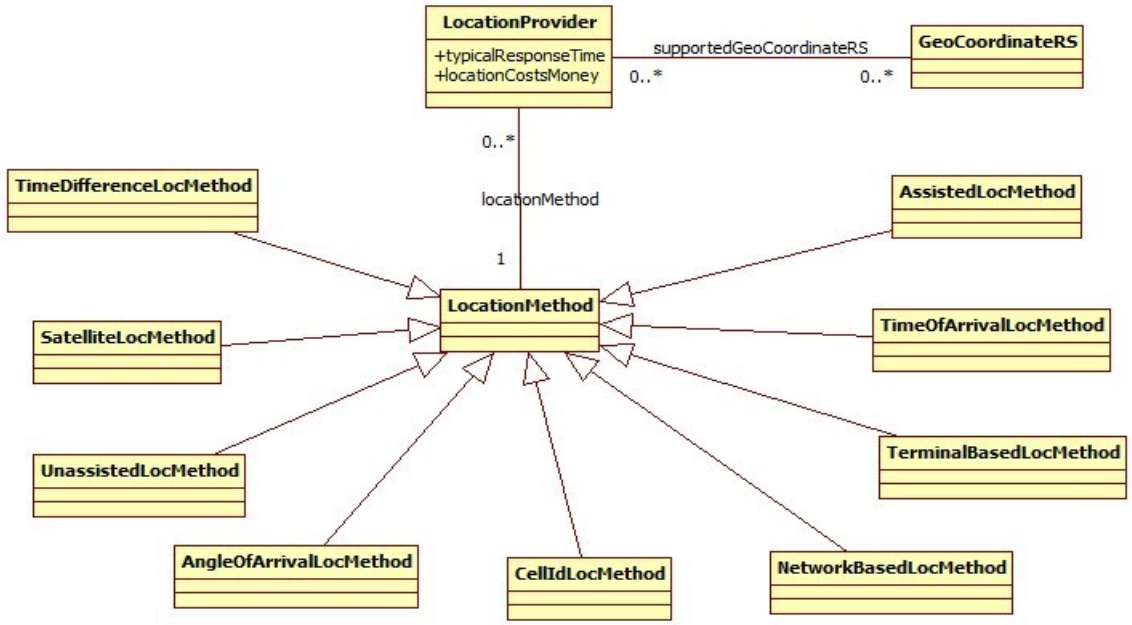
Usarem les classes que permeten representar els tipus de xarxes i modes d'operació, i els serveis de comunicacions.



Il·lustració 17: Diagrama de classes: xarxa

Localització

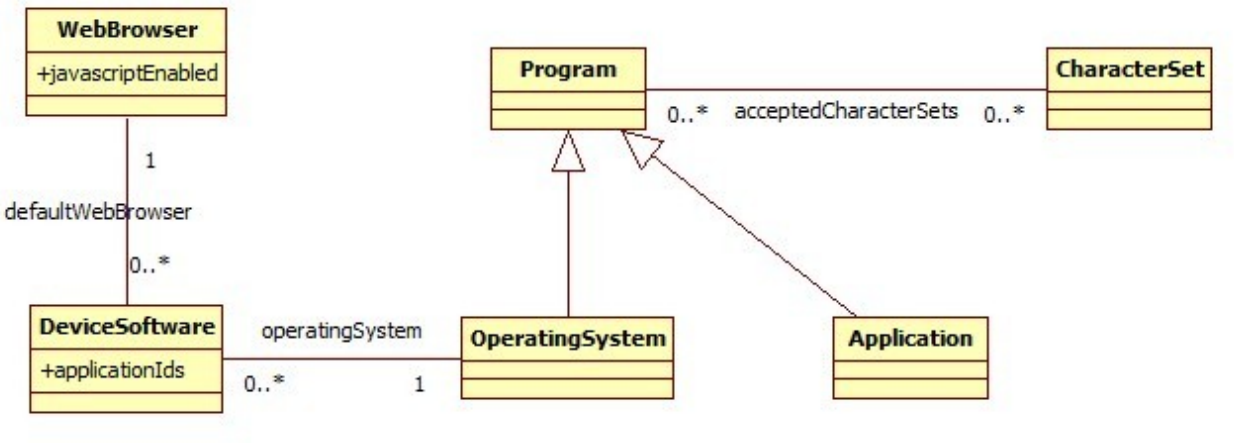
En aquest apartat només utilitzarem les classes que representen les capacitats de localització del dispositiu, i obviarem les classes per a representar la posició del dispositiu.



Il·lustració 18: Diagrama de classes: localització

Programari

En aquest apartat només tindrem en compte el sistema operatiu i el navegador que el dispositiu duu instal·lat, i no representarem la resta d'aplicacions que puguin haver-hi.



Il·lustració 19: Diagrama de classes: programari

Formats

Aquesta és la part que hem modificat i ampliat, i que hem descrit en l'apartat anterior.

Instàncies

Els dispositius mòbils instanciats en l'ontologia són els següents, tots cinc ben actuals i amb sistemes operatius diferents:

- iPhone 4S 64 GB amb sistema operatiu iOS
- Sony Tablet S 3G amb sistema operatiu Android
- Nokia Lumia 800 amb sistema operatiu Windows Phone
- Nokia 7230
- Samsung Galaxy ACE

Per a representar tot el maquinari hem hagut d'instanciar també tots els dispositius de maquinari, tipus de xarxes i sistemes de localització. La majoria d'objectes que representen tecnologies de xarxa i de serveis de comunicacions ja estaven a l'ontologia, i hem completat els que faltaven.

Hem instanciat també els sistemes operatius següents:

- iOS 5
- Android
- Android 3.1
- Windows Phone 7.5
- Nokia RM-604

Per a representar tot el suport de formats també hem instanciat contenidors i codecs.

Consultes

Per al desenvolupament del prototip caldrà que l'ontologia sigui capaç de respondre una sèrie de preguntes. A continuació llistem les preguntes en llenguatge natural amb la corresponent expressió de classe per a obtenir els resultats amb la biblioteca OWL-API i el raonador Pellet.

- Quins dispositius hi ha?

  ```
  Device
  ```

- Quins dispositius suporten HTML5?

  ```
  Device and (deviceSoftware some ( DeviceSoftware and defaultWebBrowser some (WebBrowser and supportedFormats some (Format and name value "HTML"^^string and version value "5"^^string))))
  ```

- Quins dispositius tenen acceleròmetre?

  ```
  Device and (deviceHardware some ( DeviceHardware and sensor some (Sensor and name value "accelerometer"^^string)))
  ```

- Quins dispositius tenen sensor de llum?

  ```
  Device and (deviceHardware some ( DeviceHardware and sensor some (Sensor and name value "light"^^string)))
  ```

- Quins dispositius tenen algun sistema de localització per satèl·lit?

  ```
  Device and (supportedLocationProviders some (LocationProvider and (locationMethod some SatelliteLocMethod)))
  ```

- Quins dispositius tenen Wifi?

  ```
  Device and networkSupport some (NetworkSupport and supportedNetworkModes some (NetworkMode and networkTechnology some (NetworkTechnology and (name value "IEEE 802.11a"^^string or name value "IEEE 802.11b"^^string or name value "IEEE 802.11g"^^string or name value "IEEE 802.11n"^^string ) )))
  ```

- Quins dispositius tenen càmera amb flash?

  ```
  Device and (deviceHardware some ( DeviceHardware and primaryCamera some (Camera and cameraFlash value true)))
  ```

- Quins dispositius tenen sistema operatiu Android?

  ```
  Device and deviceSoftware some (DeviceSoftware and operatingSystem some (OperatingSystem and name value "Android"^^string))
  ```

- Quins dispositius tenen una pantalla de més de 3 polzades?

  ```
  Device and deviceHardware some (DeviceHardware and display
  ```

```
some (Display and displaySize some decimal[>"3"^^decimal]) )
```

- Quins fitxers suporta el dispositiu X?

  ```
  File and inverse supports some (Device and model value "X"^^string)
  ```

Aquesta només és una petita selecció de consultes que podem fer a la nostra ontologia, i que posarem a prova en el prototip que desenvoluparem en el següent capítol.

4 Desenvolupament d'un prototip

4.1 Requeriments

El prototip que desenvoluparem ha de permetre mostrar com podem utilitzar l'ontologia de dispositius mòbils des d'una aplicació. L'aplicació ha de permetre que l'usuari pugui seleccionar un dispositiu mòbil a partir d'una llista obtinguda d'una cerca utilitzant diferents criteris de cerca, i del dispositiu mòbil que seleccioni, l'aplicació ha de mostrar algunes de les característiques del dispositiu i un llistat de fitxers que el dispositiu pugui visualitzar.

Cercador

L'usuari podrà seleccionar un dispositiu mòbil en una llista, que en principi contindrà els noms de tots els dispositius disponibles. Si ho desitja, podrà seleccionar un conjunt de característiques i aplicar un filtre que farà que la llista només mostri els dispositius que tinguin les característiques seleccionades. Hem escollit les següents característiques per a filtrar la llista de dispositius:

- Presència d'acceleròmetre
- Presència de sensor de llum
- Capacitat per a visualitzar documents HTML 5
- Presència de flash en la càmera
- Capacitat de localització per satèl·lit
- Capacitat d'accés a xarxes Wifi
- Mida de pantalla mínima
- Sistema operatiu

Característiques del dispositiu

Del dispositiu seleccionat es mostrarà una taula amb la següent informació:

- Sistema operatiu
- Model de CPU i la seva freqüència
- Quantitat de memòria RAM i memòria d'emmagatzematge
- Mida de la pantalla
- Resolució
- Enllaç a la informació en la web del fabricant

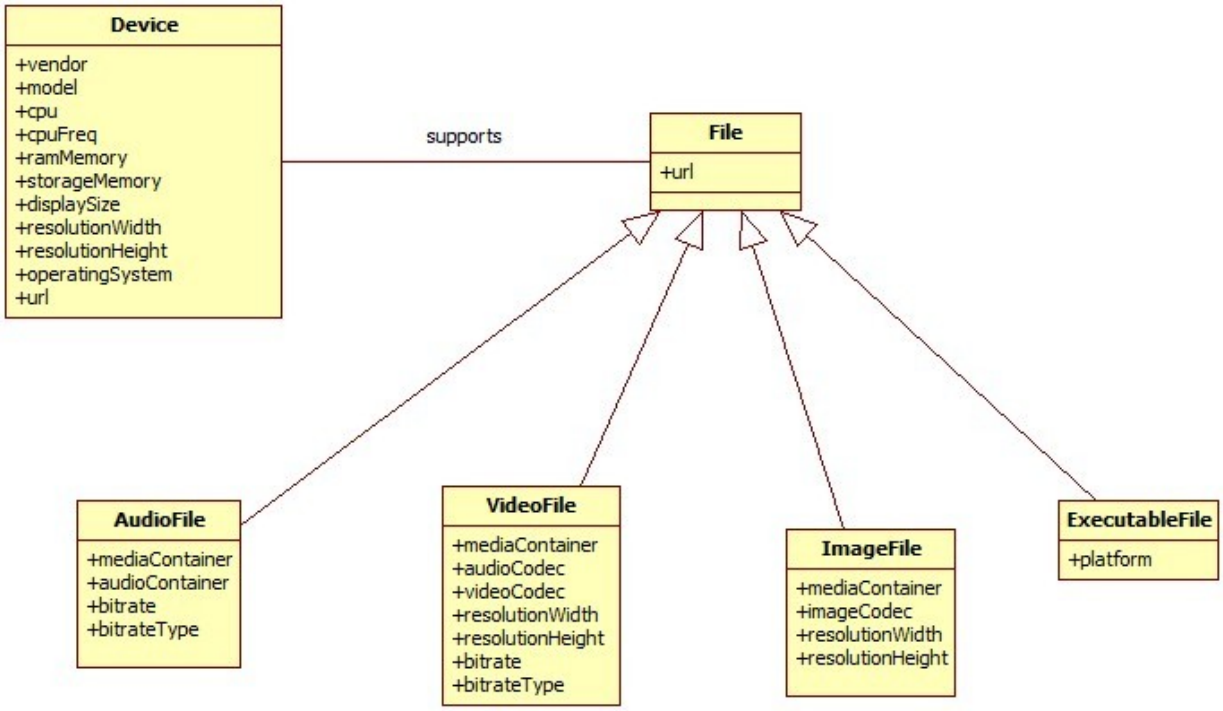
Fitxers suportats

L'ontologia emmagatzemarà també informació sobre els fitxers que hi ha en el servidor. Del dispositiu seleccionat es mostraran els fitxers que suporti. Es mostraran els següents tipus de fitxers i informacions:

13. Fitxers d'audio: URL, contenidor, codec, bitrate i tipus de bitrate
14. Fitxers de video: URL, contenidor, codec d'audio i video, resolució, bitrate i tipus de bitrate
15. Fitxers d'imatge: URL, contenidor, codec i resolució
16. Altres: URL i tipus

4.2 Model de dades

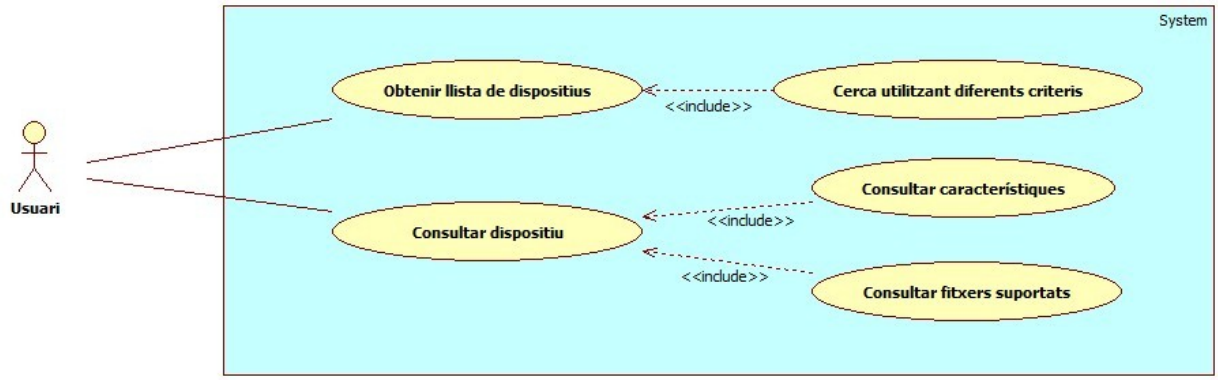
Dels requeriments identifiquem les següents classes per a modelar la informació amb què treballarà l'aplicació:



Il·lustració 20: Model de dades

4.3 Casos d'ús

Dels requeriments obtenim els següents casos d'ús:



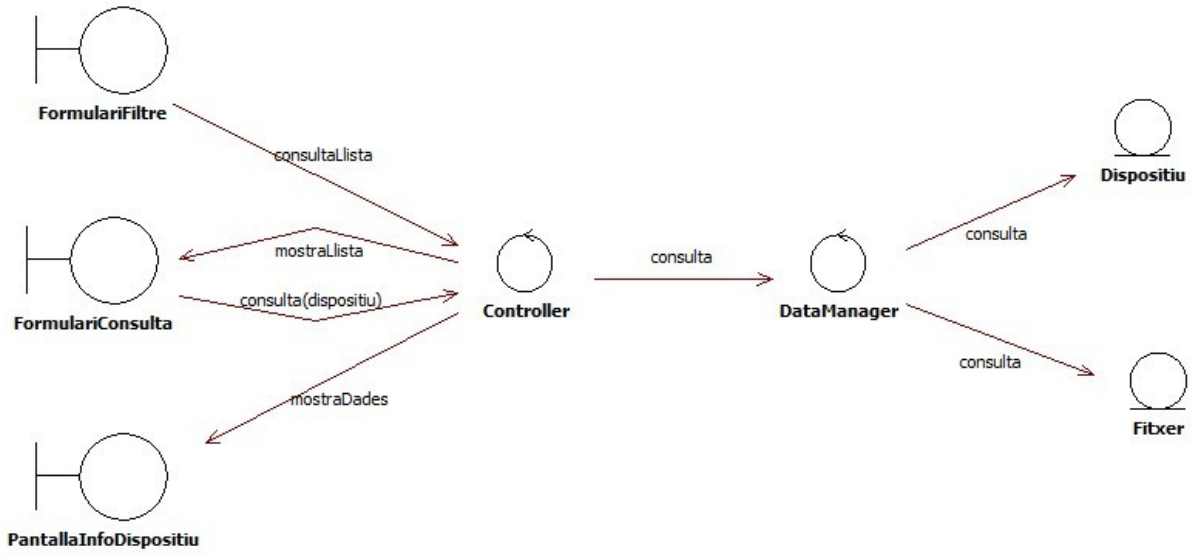
Il·lustració 21: Diagrama de casos d'ús

L'usuari té dues opcions, que correspondran a dos botons en la interfície d'usuari, i que són:

- Obtenir la llista de dispositius. Aquest inclou el cas d'ús de cercar segons diferents criteris a l'hora de generar la llista.
- Consultar un dispositiu concret. Aquest cas d'us realitza dues consultes: d'una banda algunes característiques del dispositiu, i d'altra banda els fitxers del servidor que suporta.

Posem aquests tres cas d'ús com a inclosos, ja que en una ampliació de l'aplicació es podrien reutilitzar.

A continuació mostrem en un únic diagrama de col·laboració les classes frontera, de control i d'entitat que intervindran en els casos d'ús.

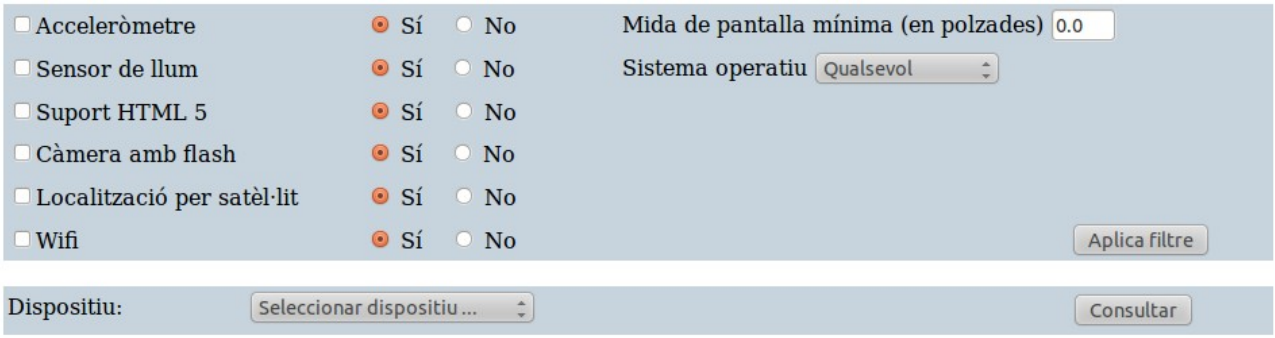


Il·lustració 22: Diagrama de col·laboració

A través de FormulariFiltre, l'usuari introdueix un filtre format per un conjunt de característiques de dispositius mòbils. A continuació el controlador realitza una consulta al DataManager, el qual obté de l'ontologia els dispositius que compleixen les condicions del filtre, i els mostra en el FormulariConsulta. Aleshores l'usuari selecciona un dispositiu de la llista i el controlador realitza les consultes necessàries per a obtenir la informació del dispositiu seleccionat i els fitxers que suporta.

4.4 Interfície d'usuari

El prototip serà una aplicació web senzilla. Contindrà una única plana amb dos formularis i una àrea on mostrar els resultats, que correspondrien a les classes frontera especificades en l'apartat anterior.



Il·lustració 23: Interfíce gràfica d'usuari

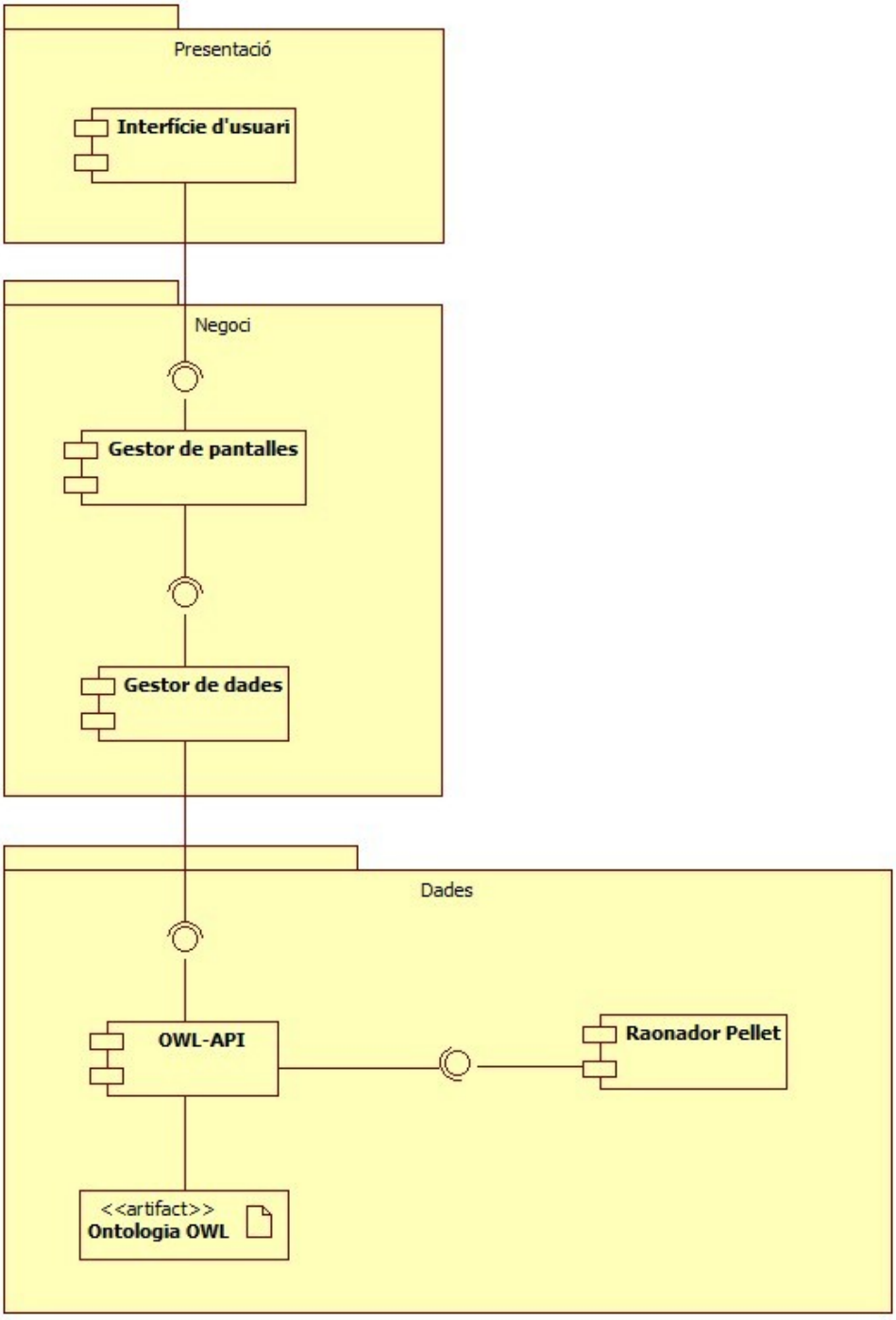
El raonador resol per inferència utilitzant regles quins fitxers suporta cada dispositiu. Es tracta d'una resolució bàsica. Per als fitxers de video, imatge i audio, es té en compte si es suporten els codecs i contenidors. Per als fitxers executables es té en compte quina és la seva plataforma d'execució. Per a la resta es té en compte el seu format.

4.5 Implementació

Com que es tracta d'una aplicació web, seguirà el model de tres capes: presentació, negoci i dades. Implementarem l'aplicació utilitzant les Java Server Faces (JSF), que és un marc de desenvolupament de la capa de presentació que segueix l'arquitectura Model-Vista-Controlador, que forma part de les especificacions de l'arquitectura JEE.

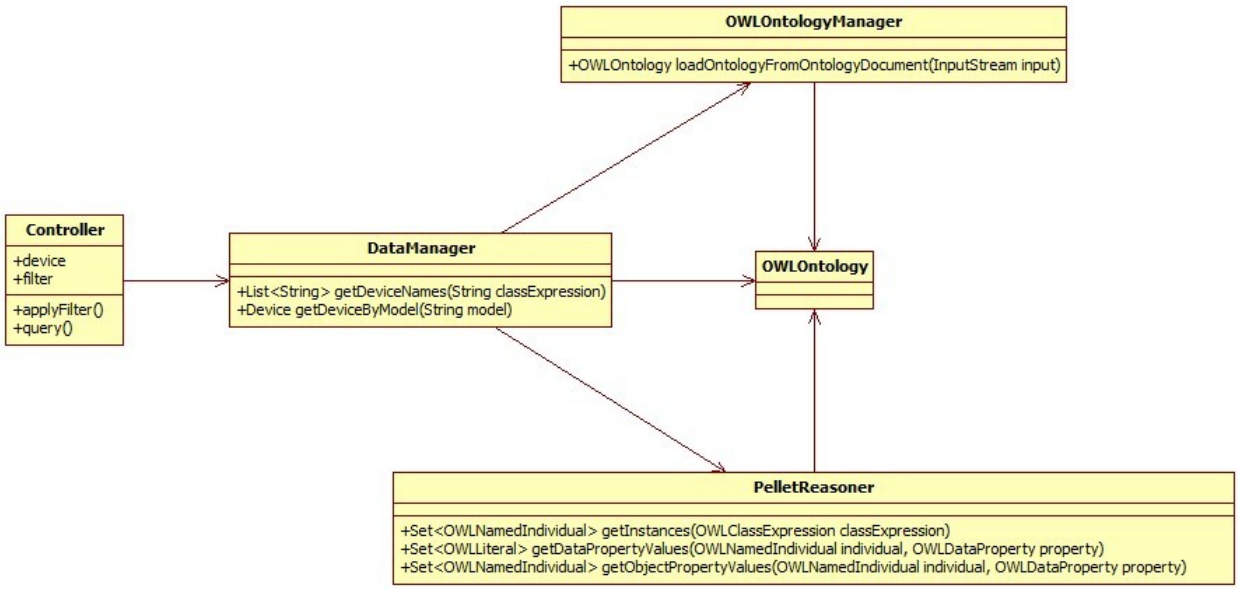
Com podem veure en el diagrama de components, en la capa de dades hi trobem les biblioteques de Java OWL-API i el raonador Pellet que fan ús de l'ontologia desada en un document OWL. En la capa de negoci implementarem les classes de Java necessàries per a accedir a les dades i per a controlar els elements de la interfície d'usuari. En la capa de presentació hi haurà les pàgines JSP (Java Server Pages) amb elements de JSF, les quals quan es despleguin seran Servlets que serviran documents HTML a l'usuari que es connecti amb el navegador. En aquest prototip senzill hi haurà

una única pàgina JSP.



Il·lustració 24: Diagrama de components

En la figura següent mostrem les classes de procés implementades i les classes més importants utilitzades de les biblioteques OWL-API i Pellet.



Il·lustració 25: Classes de procés

Només hem mostrat les operacions més significatives, i també les hem simplificat per a que siguin més entenedores i no haver d'entrar en el detall de l'ús de les biblioteques de programació. Les expliquem a continuació:

- applyFilter: Mostra en la interfície d'usuari els dispositius que resulten d'aplicar el filtre. Utilitza la funció getDeviceNames de la classe DataManager
- query: Mostra en la interfície d'usuari la informació del dispositiu consultat. Utilitza la funció getDeviceByModel de la classe DataManager
- getDeviceNames: A partir d'una expressió de classe emprant la sintaxi Manchester, retorna un conjunt de noms de dispositius que corresponen a les instàncies obtingudes de l'ontologia. Utilitza les funcions getInstances i getDataPropertyValues de la classe PelletReasoner
- getDevicesByModel: Rep com a paràmetre el model de dispositiu i retorna el dispositiu especificat. Utilitza les funcions getInstances, getDataPropertyValues i getObjectPropertyValues de la classe PelletReasoner.
- getInstances: Rep una expressió de classe i retorna un conjunt d'instàncies pertanyents a la classe especificada.
- GetDataPropertyValues: Rep com a paràmetres un objecte i una propietat de tipus dada, i retorna els valors literals que la propietat té per a aquest objecte.
- GetObjectPropertyValues: Rep com a paràmetres un objecte i una propietat de tipus objecte, i retorna els objectes que conté la propietat per a aquest objecte.
- LoadOntologyFromOntologyDocument: Rep com a paràmetre un flux d'entrada que representa un document OWL i retorna una ontologia en forma d'objecte de tipus OWLOntology.

4.6 Manual d'instal·lació i utilització

Instal·lació

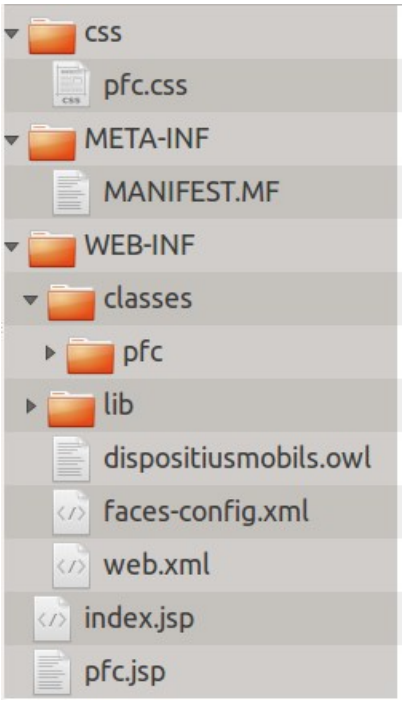
Aquesta aplicació s'ha de poder desplegar en un servidor web que compleixi l'especificació JSP 2.0, com és el cas de Tomcat 7.0.23 que hem utilitzat per al desenvolupament del prototip. La instal·lació del servidor és senzilla, només cal descomprimir el contingut del paquet que ens descarreguem en algun directori del sistema de fitxers.

Per a arrencar el servidor en un sistema Linux cal executar el guió de sistema `startup.sh` que es troba en el directori `bin` del directori del servidor, i per a aturar-lo només cal executar `shutdown.sh` que es troba en el mateix directori. Per a sistemes Windows és fa de manera equivalent executant els fitxers `startup.bat` i `shutdown.bat`.

El prototip es presenta empaquetat en un fitxer WAR. Per a desplegar-lo en el servidor només cal copiar-lo al directori `webapps` del directori de Tomcat. S'hi accedeix connectant amb un navegador a la URL http://localhost:8080/PFC si la connexió és des del mateix ordinador.

Contingut del fitxer WAR

El fitxer PFC.WAR conté la següent estructura de directoris:



Il·lustració 26: Fitxer WAR

- pfc.css: Full d'estil CSS
- MANIFEST.MF: Fitxer per defecte sense modificacions que es crea automàticament
- pfc: Hi trobem totes les classes implementades per al prototip. Són les següents:
  - Device, File, VideoFile, ImageFile, AudioFile, ExecutableFile que representen les

entitats del model de dades

- Controller, que controla la interfície d'usuari i realitza consultes a la classe DataManager
- DataManager, estableix la comunicació amb la biblioteca OWL-API i el raonador Pellet

- lib: Biblioteques Java per a JSF, OWL-API i raonador Pellet
- dispositiusmobils.owl: Ontologia de dispositius en format OWL-RDF
- faces-config.xml: Fitxer de configuració de les JavaServer Faces
- web.xml: Descriptor de desplegament
- index.jsp: Pàgina inicial, redirecciona a pfc.jsp
- pfc.jsp: Pàgina amb la interfície d'usuari del prototip, implementada en JSP i utilitzant components JSF

Utilització

L'ús del prototip és ben senzill i la serva interfície d'usuari es troba en una única pàgina web. En la part superior hi trobem un formulari on hi podem seleccionar diversos filtres. Just a sota hi trobem un segon formulari on es pot seleccionar un dispositiu. La resta de pantalla servirà per a presentar els resultats.

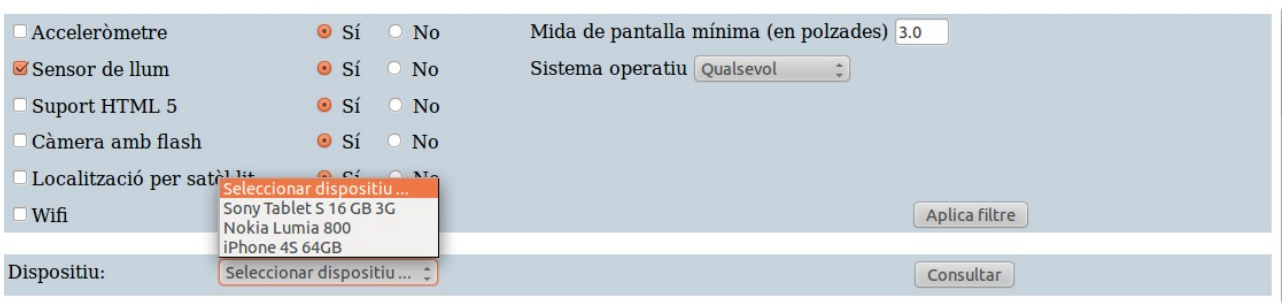
Hi ha sis filtres de tipus booleà que s'activen marcant la casella de verificació corresponent i establint el valor desitjat a Sí o No mitjançant botons d'opció.

Els altres dos filtres no porten casella de verificació. El de la mida de pantalla mínima desitjada l'especifiquem mitjançant un número decimal, i si no volem activar-lo doncs el deixem al seu valor per defecte que és 0. El filtre per sistema operatiu permet seleccionar d'un desplegable, i una de les opcions és “Qualsevol”, la qual cosa equival a no activar el filtre.

Un cop s'han activat els valors de filtre desitjat, el filtre s'aplica amb el botó d'aplicar filtre. L'aplicació cercarà en l'ontologia els dispositius que compleixen les condicions especificades i carregarà en el desplegable del segon formulari els noms dels dispositius trobats.

Per a consultar la informació d'un dispositiu només cal seleccionar-lo en el desplegable i prémer el botó Consultar. A continuació apareix la informació del dispositiu i els fitxers del servidor que suporta classificats en les categories especificades en els requeriments.

En aquesta imatge mostrem com utilitzaríem el cercador de dispositius per a obtenir tots els dispositius amb sensor de llum, una mida de pantalla mínima de tres polzades i qualsevol sistema operatiu.



Il·lustració 27: Cercador de dispositius

5 Conclusions

5.1 Resum del treball realitzat

A l'inici del projecte vam establir l'objectiu de crear una ontologia OWL per a descriure dispositius mòbils i utilitzar aquesta descripció en un prototip que a partir de la selecció d'un dispositiu per part de l'usuari, es generés una plana web amb els enllaços als fitxers suportats pel dispositiu emmagatzemats en un servidor.

En primer lloc hem fet una recerca per Internet per a localitzar treballs realitzats en aquest sentit. El projecte més actualitzat i que conté una gran base de dades de dispositius és WURFL. Com hem vist en el capítol 2, es tracta d'un gran fitxer XML amb informació sobre molts dispositius i una biblioteca de programació. També hem vist altres projectes molt relacionats amb aquest, com són CC/PP i UAProf, basats en RDF. Tots aquests projectes tenen per objectiu adaptar el contingut mostrat a les característiques dels dispositius, que es troben emmagatzemades en el servidor.

Pel que fa a solucions basades en OWL tenim l'ontologia Delivery Context i l'ontologia mIO!. La primera es va crear amb la intenció de permetre representar tota la informació de maquinari, programari, xarxes disponibles, estat, etc. dels dispositius mòbils amb la intenció d'utilitzar-la per adaptar el contingut servit a un dispositiu. L'ontologia mIO! va més enllà i fent ús també de l'ontologia Delivery Context, entre altres, pretén no solament representar la informació de dispositius mòbils, sinó representar també tota la informació relativa a l'usuari, com per exemple informació de l'entorn on es troba, contactes de l'agenda, gustos i preferències, etc. A diferència de WURFL i UAProf, tant Delivery Context com mIO! necessiten que els dispositius comuniquin informació al servidor.

Per l'abast del nostre treball hem considerat l'ontologia Delivery Context prou adequada. De fet, permet representar molta més informació de la que en principi necessitàvem. De totes maneres, hi hem fet algunes modificacions per a poder representar millor els formats de fitxers i el maquinari. WURFL i UAProf ens han anat bé per a obtenir informació addicional sobre els dispositius.

Cal dir que l'ontologia Delivery Context ha vist aturat el seu desenvolupament des de l'any 2009, en què el grup *Ubiquitous Web Applications Working Group* es va dissoldre. En el darrer document podem veure que el grup tenia per intenció publicar diagrames, tutorials, exemples i altres recursos per a aprendre a utilitzar-la. Malauradament no hem pogut servir-nos d'aquests materials.

Per finalitzar el projecte hem fet un prototip que fa ús de l'ontologia de dispositius, i que serveix de demostració del que es pot fer amb la tecnologia de la web semàntica.

5.2 Adaptació de continguts i web semàntica

Avui en dia tenim la gran diversitat de dispositius que es connecten a Internet fa que sigui necessària l'adaptació de continguts. Existeixen projectes per a aquest propòsit com WURFL o UAProf, però no utilitzen les eines de la web semàntica.

Amb aquest projecte hem mostrat que les ontologies de dispositius mòbils es poden utilitzar per a l'adaptació de continguts, i que la web semàntica permet representar models de dades amb molta expressivitat.

L'ús dels raonadors fa que es verifiqui la consistència dels models i a més permet la inferència, una eina molt potent per a obtenir informació nova a partir de la ja existent utilitzant el raonament lògic.

Els raonadors treballen amb el criteri de Open World Assumption, és a dir, que no es pot afirmar que un objecte no compleix una certa propietat a no ser que es pugui deduir a partir dels axiomes assertats explícitament en l'ontologia. En la nostra ontologia, per exemple, podem fer que el raonador Pellet ens retorni els dispositius que tenen acceleròmetre, però no podem fer que ens retorni els dispositius que no en tenen. El fet que l'acceleròmetre no estigui en les propietats dels dispositius no s'interpreta com a que no en tingui. Això fa que el cercador hagi de ser completat mitjançant l'aplicació, i considerar que tots els dispositius que no es poden classificar com a "dispositius amb acceleròmetre" es considerin com a "dispositius sense acceleròmetre".

5.3 Línies de futur

Hem vist que el projecte més actiu en la línia de l'adaptació de continguts per a dispositius mòbils és WURFL. Tot i que el model WURFL no té la riquesa semàntica que té una ontologia OWL, sí que és interessant tota la informació sobre dispositius que conté. Una possible línia de treball seria estudiar el procés de migració de tot aquest repositori a OWL.

Ja a les acaballes del projecte, hem vist que ha aparegut una nova versió, la 4, de l'ontologia mIO! Sembla que es tracta d'un projecte actiu, al contrari de l'ontologia DeliveryContext, i per tant probablement és més adient utilitzar mIO! en aplicacions de web semàntica. Donada la gran quantitat de classes i propietats que conté, i que no es té a l'abast documentació per a utilitzar-la, caldria invertir cert temps en fer-ne un estudi a fons.

Pel que fa al prototip, és clar que es tracta d'un cercador molt bàsic realitzat en un temps limitat, però ja es poden albirar les grans possibilitats d'ampliació que té. Si ens fixem en la potència de la sintaxi Manchester per a fer consultes en l'ontologia, veurem que podem efectuar qualsevol cerca que puguem imaginar. També és evident que la interfície gràfica és susceptible de millora, creant més pantalles i afegint altres elements com JavaScript, CSS3 o utilitzar HTML 5.

En una aplicació real seria convenient emmagatzemar les dades de l'ontologia en una base de dades XML, ja que d'aquesta manera l'emmagatzematge seria molt més eficient. A més, en la interfície d'usuari s'haurien d'incorporar formularis d'entrada de dades. També es podria estudiar si és possible el traspàs de dades des d'altres recursos no ontològics com WURLF.

Pel que fa al suport de fitxers, hem fet que sigui la inferència qui ho determini. Com ja hem comentat anteriorment, el fet que la web semàntica utilitzi Open World Assumption limita les possibilitats en aquest sentit. De totes maneres, el fet de poder utilitzar l'ontologia des d'una aplicació Java fa que l'ampliació de funcionalitats sigui il·limitada. En aquesta línia es podria determinar quins fitxers són compatibles amb un determinat dispositiu en funció del maquinari necessari per a executar-los si es tracta de programes, o tenir en compte el bitrate, les freqüències de mostreig i altres característiques en els fitxers de video i audio. L'ontologia està preparada per a emmagatzemar aquests tipus d'informació, però no s'ha implementat en l'aplicació per manca de temps.

Índex d'il·lustracions

Referències bibliogràfiques i d'Internet

Bergsten, Hans. (2004). *JavaServer faces*. O'Reilly.

Antoniou, Grigoris; van Harmelen, James. (2004). *A Semantic Web primer*. **The MIT Press.**

Allemang, Dean; Hendler, James. (2008). *Semantic Web for the working ontologist : effective modeling in RDFS and OWL*. Elsevier Science.

Hebeler, John. *Semantic Web programming*. (2009). Wiley.

http://wurfl.sourceforge.net/

http://www.obitko.com/tutorials/ontologies-semantic-web/

http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ

http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL

http://protegewiki.stanford.edu/wiki/Protege4Migration#Side_by_Side_Comparison

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*.(2003). http://www.daml.org/2003/11/swrl/

Christine Golbreich, Evan K. Wallace, Peter F. Patel-Schneider. (2009). *OWL 2 Web Ontology Language New Features and Rationale.* http://www.w3.org/TR/owl2-new-features/

http://dior.ics.muni.cz/~makub/owl/

Sean Bechhofer. *Programming to the OWL API: Introduction*. University of Manchester.

Ignazio Palmisano & the OWL API team. *The rough guide to the OWL API: a tutorial.*

Martin O'Connor, Amar Das. *SQWRL: a Query Language for OWL*. Standford Center for Biomedical Informatics Research.

Matthew Horridge. *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3.*

David Mérida, Ramón Fabregat, Silvia Baldiris. *Sistemas heterogéneos adaptativos basados en el contexto.* Universitat de Girona.