2. LANDASAN TEORI

Bab ini menjelaskan landasan teori dari pengembangan sistem dengan beberapa tinjauan pustaka, yaitu definisi *knowledge management system,* ontologi, RDF, dan OWL. Selain itu juga terdapat penjelasan *previous works* dari beberapa lembaga riset dunia mengenai pengembangan *knowledge management system*.

2.1 Definisi *Knowledge Management*

Sebelum membahas lebih detail mengenai apa itu *knowledge management*, terlebih dahulu mengetahui perbedaan mendasar data, informasi, dan pengetahuan (*knowledge*). Perbedaan tersebut dapat diketahui melalui definisi dan contoh. Berikut penjelasannya.

2.1.1 Data

Data merupakan unit terkecil yang bersifat statis dan merupakan representasi dari fakta, observasi, dan persepsi (bisa benar ataupun salah) yang ditemukan dalam aktivitas sehari-hari. Contohnya adalah suatu toko menawarkan potongan harga terhadap dua celana dan dua baju.[6]

2.1.2 Informasi

Informasi adalah hasil pengolahan dari data yang dapat memberikan gambaran lebih jelas terhadap suatu *trend* atau pola dari data tersebut. Informasi bersifat dinamis. Bagi pengelola toko, banyaknya penjualan (dalam rupiah, kuantitas, dan persentase penjualan harian) suatu celana dan baju dan lain-lain merupakan contoh dari informasi. Pengelola toko dapat menggunakan informasi tersebut untuk membuat keputusan dalam menentukan harga pembelian.[6]

2.1.3 Pengetahuan

Pengetahuan memiliki definisi, yaitu data dan informasi yang digabung dengan kemampuan, intuisi, pengalaman, gagasan, motivasi dari sumber yang

kompeten. Pengetahuan merupakan level tertinggi, informasi pada level menengah, sedangkan data level terendah. Pengetahuan dapat merujuk kepada suatu informasi yang memiliki arah, aksi, dan membuat keputusan. Pengetahuan sangat bernilai.

Contohnya adalah penjualan harian terhadap celana dan baju termasuk informasi lainnya (informasi banyaknya bahan kain yang ada di gudang), untuk menghitung berapa banyak kain yang akan dibeli. Hubungan antara banyaknya kain yang dipesan, banyaknya kain yang ada di gudang dan penjualan sehari-hari celana dan baju adalah contoh dari pengetahuan. Dengan memahami hubungan ketiganya dapat membantu dalam menggunakan informasi untuk menghitung kuantitas kain yang dibeli. Akan tetapi, kuantitas kain yang dipesan dipertimbangkan sebagai informasi bukan suatu pengetahuan.

Sedangkan *management* berarti merencanakan, mengumpulkan dan mengorganisir, mengkoordinasikan *resource* untuk suatu tujuan. Sehingga, *knowledge management* adalah proses yang mengkoordinasikan penggunaan informasi, pengetahuan dan pengalaman.

Knowledge management juga berkaitan dengan konsep *intellectual capital*. Terdiri atas *human capital, relational capital* dan *structural capital*, *intellectual capital*. *Intellectual capital* dilihat sebagai objek dari *knowledge*, menyediakan dasar dalam penilaian *knowledge management*. *Human capital* merujuk kepada *body of knowledge* milik perusahaan berasal dari pikiran dan ide karyawan pada organisasi. *Relational capital* menjelaskan bagaimana suatu organisasi berhubungan dengan *customer* dan *vendor.* Sedangkan *structural capital* merujuk kepada *databases*, *file customer*, *software*, *manuals, trademarks,* dan struktur organisasi, kebijakan dan budaya organisasi. [6]

Knowledge management mengalami progres dan perkembangan yang meningkat, terbukti dengan merambahnya bidang tersebut dengan bidang teknologi informasi. Teknologi informasi merupakan suatu media atau sarana untuk *sharing* atau mentransfer secepat pertumbuhan *knowledge* itu sendiri terjadi, memberikan pergerakan dari informasi dengan kecepatan tinggi, efektif, dan efisien. Sebagai contoh, komputer dapat mengambil data dari penghitungan fenomena alam, dan

dengan cepat memanipulasi data sehingga dapat dimengerti atas fenomena yang ditampilkan. Aplikasi yang menghasilkan seperti sinergi tersebut yang didasarkan oleh *IT-based* dan mekanisme sosial atau struktural dapat disebut dengan *knowledge management system*. Definisi lain dari *knowledge management system* adalah suatu sistem *IT-based* yang digunakan untuk mengelola, menciptakan, mengambil, menyimpan, menyebarkan data, informasi serta pengetahuan.

2.2 Ontologi

Pada mulanya ontologi diambil dari istilah filsafat, dimana ontologi dipandang sebagai penjelasan sistematis dari suatu yang *exist*. Lalu, pengertian tersebut diganti menjadi suatu objek, konsep, dan entitas lain yang diasumsikan "*exist*" pada suatu area dan terdapat relasi diantaranya. Ontologi menjelaskan secara eksplisit suatu konsep. Konsep ini mengacu kepada kumpulan konsep yang digunakan untuk merepresentasikan suatu objek nyata atau suatu *knowledge.* Eksplisit artinya mendefinisikan tipe konsep dan batasan yang digunakan dalam memberikan arti formal yang dipahami oleh bahasa mesin.

Dalam ilmu kecerdasan buatan (*artificial intelegence)*, suatu yang "*exist*" merupakan sesuatu yang dapat direpresentasikan. Ketika pengetahuan adalah domain yang direpresentasikan pada deklarasi formal, himpunan dari objek yang dapat direpresentasikan disebut *universe of discource*. Himpunan objek tersebut dan penjelasan relasi antaranya direfleksikan pada representasi *vocabulary* dimana *knowledge-based program* menggambarkan pengetahuan tersebut. Sehingga, dalam konteks *artificial intelegence*, dapat dideskripsikan ontologi adalah suatu mekanisme pendefinisian sebuah terminologi representasi. Pada ontologi, bentuk representasi primitif berupa *classes* atau *set*, *attributes* atau *properties,* dan *relationship (*atau relasi antara *class* dan *member).* [7]

Pada beberapa tahun lalu, komunitas peneliti AI bersama-sama mencari suatu cara untuk membantu *sharing knowledge*. Mereka menginginkan suatu program komputer dapat berinteraksi dengan dan berdiri pada informasi dari program komputer lainnya. Akhirnya, pada tahun 2001 Tim Berners-Lee membuat misi

menjadi nyata dengan menciptakan *semantic web*. Ide dasar dari *semantic web* adalah *web* yang memiliki data yang terdefinisi dan saling terhubung sehingga data tersebut dapat digunakan untuk penemuan, otomasi, integrasi, dan *reuse* secara efektif melalui beberapa aplikasi. Ontologi inilah merupakan ide utama pemrosesan data oleh mesin (atau disebut juga *machine-readable)* pada *semantic web*. [8]

Salah satu jenis *knowledge management system* yang sesuai dengan penerapan ontologi adalah *knowledge sharing system*, menggunakan teknologi *semantic web. Semantic web* diekpresikan dalam spesifikasi formal, yaitu *Resource Description Framework (RDF),* beberapa format *data interchange* (misal *RDF/XML, N3, Turtle, N-Triples*), dan notasi seperti *RDF Schema (RDFS)* dan *the Web Ontology Language (OWL)*, semua menampilkan deskripsi formal suatu konsep, terminologi, dan relasi diantara domain pengetahuan yang diberikan. Pembahasan berikutnya mengenai bahasa yang banyak dipakai pada pengembangan *semantic web* dan juga dipakai pada pengembangan *knowledge management system* di lembaga riset BPPT, yaitu RDF(S) dan OWL.

2.3 *RDF (Resource Description Framework)*

RDF (*Resource Description Framework*) adalah bahasa yang digunakan untuk merepresentasikan informasi tentang *resource* pada *World Wide Web*, merepresentasikan metadata tentang *web resource*, seperti judul, pengarang, dan modifikasi tanggal suatu halaman *web*, hak cipta dan lisensi informasi tentang dokumen *web*, dan lain-lain. Akan tetapi, pada umumnya konsep dari "*web resource*", RDF dapat juga digunakan untuk merepresentasikan informasi tentang "*things*" yang dapat diidentifikasikan pada *web*, bahkan ketika tidak bisa secara langsung didapatkan pada *web*. Contohnya termasuk informasi mengenai barang yang terdapat pada fasilitas *online shopping* (misalnya informasi mengenai spesifikasi, harga, dan ketersediaan barang), atau deskripsi mengenai preferensi pengguna *web* untuk informasi pengiriman barang. [11]

RDF adalah ide dasar untuk identifikasi "*things*" menggunakan *web identifier* (yang disebut dengan *Uniform Resource Identifiers* atau *URIs*), dan mendeskripsikan *resources* ke dalam *simple properties* dan *property value*. RDF mampu untuk merepresentasikan *statement* sederhana tentang *resources* dalam suatu grafik yang terdiri dari *nodes* dan *arcs* yang mewakili *resource* serta *property* dan *value*. Hal ini dapat dilihat dari suatu *statement* "seseorang diidentifikasi oleh http://www.w3.org/People/EM/contact#me, bernama Eric Miller, dengan *e-mail* em@w3.org, dan memiliki gelar Dr." dapat direpresentasikan dalam suatu grafik RDF berikut ini. [11]

Gambar 2.1 RDF Graph Menjelaskan Eric Miller [11]

- *individuals*, contoh: **Eric Miller**, diidentifikasikan dengan http://www.w3.org/People/EM/contact#me
- *kinds of things*, contoh: **Person**, diidentifikasikan dengan http://www.w3.org/2000/10/swap/pim/contact#Person
- *properties of those things*, contoh: **mailbox**, diidentifikasikan dengan http://www.w3.org/2000/10/swap/pim/contact#mailbox

- *values of those properties*, contoh: **mailto:em@w3.org** sebagai *value* dari *property* mailbox (RDF juga menggunakan karakter string seperti "Eric Miller", dan values dari tipe data lain seperti *integer* dan *date* , sebagai *value* dari *property*)

2.3.1 RDF Model

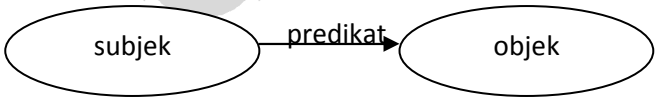
Telah diketahui sebelumnya RDF adalah ide dasar untuk mengekspresikan *statement* sederhana tentang *resources*, dimana tiap *statement* terdiri dari subjek, predikat, dan objek. Pada RDF, *statement* berikut ini

```
http://www.example.org/index.html has a creator whose value is
John Smith
```

dapat direpresentasikan dengan sebuah RDF *statement* yang memiliki:

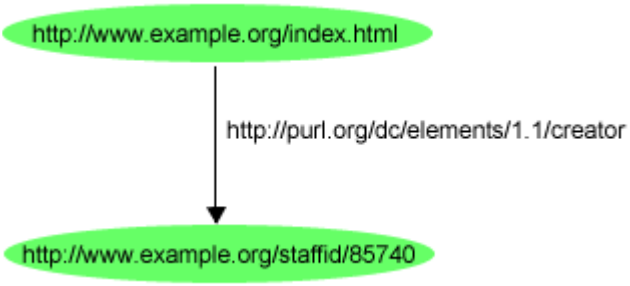
- **subjek** http://www.example.org/index.html
- **predikat** http://purl.org/dc/elements/1.1/creator
- **objek** http://www.example.org/staffid/85740

Dapat dilihat di atas URIref digunakan tidak hanya untuk identifikasi subjek , tapi juga predikat dan objek, sebagai pengganti dalam penggunaan kata “creator” dan “John Smith”. RDF *model statement* ditampilkan dalam bentuk grafik (kumpulan dari *triple*), terdiri dari *nodes* untuk subjek dan objek serta *arcs* untuk predikat (atau disebut dengan *property* berfungsi sebagai relasi. [10]



Gambar 2.2 *Graph Data Model* [10]

dapat dillihat grafik RDF untuk *statement* di atas adalah

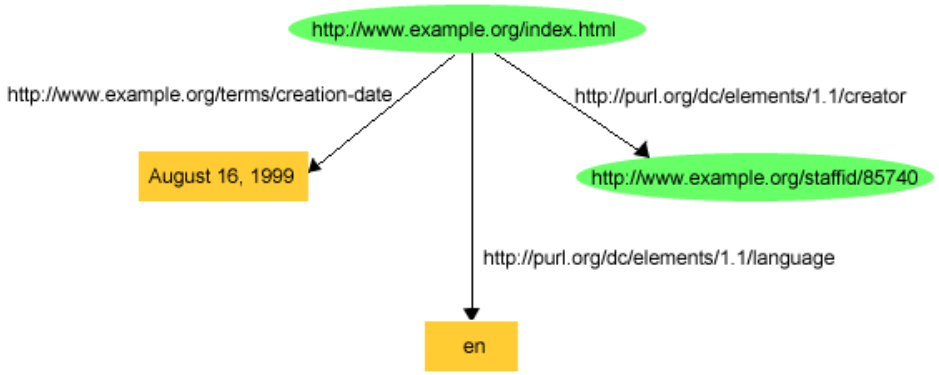


Gambar 2.3 RDF Graph Menjelaskan John Smith [11]

Jika ditambahkan *statement* :

```
http://www.example.org/index.html has a creation-date whose
value is August 16, 1999
http://www.example.org/index.html has a language whose value is
English
```

Pada grafik RDF, akan ditampilkan dengan menggunakan URIrefs yang sesuai untuk nama *property* “creation-date” dan”language”.



Gambar 2.4 RDF Graph *Add Statement* John Smith [11]

Dari grafik di atas, jika ditulis kedalam notasi *triple* akan dihasilkan seperti berikut.

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740> .

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date> "August 16, 1999" .

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language> "en".
```

Penulisan notasi *triples* di atas akan memakan baris yang panjang, oleh karena itu terdapat cara menyingkat dengan mensubstitusi XML *qualified name (Qname)* tanpa tanda kurung siku "<>". *Qname* mengandung sebuah *prefix* yang telah di-*assign* untuk *namespace* URI, diikuti dengan tanda ":" serta sebuah *local name*. Misalnya jika *prefix Qname* `foo` di-*assign* terhadap *namespace* URI `http://example.org/somewhere/`, maka *Qname* `foo:bar` adalah penyingkatan untuk URIref `http://example.org/somewhere/bar`. [11]

Contoh primer yang digunakan untuk beberapa *Qname* yang terkenal adalah sebagai berikut:

- *prefix* `rdf:`, *namespace* URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- *prefix* `rdfs:`, *namespace* URI: `http://www.w3.org/2000/01/rdf-schema#`
- *prefix* `dc:`, *namespace* URI: `http://purl.org/dc/elements/1.1/`
- *prefix* `owl:`, *namespace* URI: `http://www.w3.org/2002/07/owl#`
- *prefix* `ex:`, *namespace* URI: `http://www.example.org/` (or `http://www.example.com/`)
- *prefix* `xsd:`, *namespace* URI: `http://www.w3.org/2001/XMLSchema#`

Variasi lainnya untuk "example" *prefix* `ex:` juga digunakan untuk kebutuhan lain, yaitu:

- *prefix* `exterms:`, *namespace* URI: `http://www.example.org/terms/` (*example organization*),
- *prefix* `exstaff:`, *namespace* URI: `http://www.example.org/staffid/` (*example organization staff's identificator*),
- *prefix* `ex2:`, *namespace* URI: `http://www.domain2.example.org/` (*example second organization*), dan lain-lain.

Dengan penyingkatan seperti ini, kumpulan notasi *triples* sebelumnya bisa diganti dengan [11]

```
ex:index.html  dc:creator             exstaff:85740 .
ex:index.html  exterms:creation-date  "August 16, 1999" .
ex:index.html  dc:language            "en" .
```

2.3.2 XML *syntax* untuk RDF: RDF/XML

RDF memiliki XML *syntax* untuk menulis atau mengubah grafik RDF, disebut dengan RDF/XML. Sebagai contoh dari suatu *statements* di bawah ini:

```
http://www.example.org/index.html has a creation-date whose
value is August 16, 1999
```

Grafik RDF untuk hal di atas adalah seperti berikut

Gambar 2.5 RDF Graph [11]

Dengan sebuah representasi *triple* sebagai berikut

```
ex:index.html   exterms:creation-date   "August 16, 1999" .
```

Sedangkan untuk RDF/XML *syntax* dapat dilihat seperti ini

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
3. xmlns:exterms="http://www.example.org/terms/">
4. <rdf:Description
rdf:about="http://www.example.org/index.html">
5. <exterms:creation-date>August 16, 1999</exterms:creation-
date>
6.</rdf:Description>
7. </rdf:RDF>
```

Baris 1, `<?xml version="1.0"?>`, adalah *XML declaration*, yang mengidentifikasikan bahwa seluruh isi adalah XML, dan versi berapakah XML yang digunakan.

Baris 2 dimulai dengan `rdf:RDF` element, menjelaskan bahwa konten XML (dimulai dari sini dan berakhir dengan `</rdf:RDF>` pada baris 7) digunakan untuk representasi RDF. Berikutnya setelah `rdf:RDF` pada baris yang sama terdapat sebuah deklarasi XML *namespace*, direpresentasikan sebagai sebuah atribut `xmlns` dari `rdf:RDF` start-tag. Deklarasi ini mengkhususkan pada semua *tags* di konten ini berawalan dengan `rdf:` adalah bagian dari *namespace* diidentifikasikan oleh URIref `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. URIrefs mulai dengan *string* `http://www.w3.org/1999/02/22-rdf-syntax-ns#` digunakan pada suatu dari RDF *vocabulary*. [11]

Baris 3 mengkhususkan kepada deklarasi lain XML *namespace*, yaitu *prefix* `exterms:`. Hal ini dijelaskan sebagai atribut lain `xmlns` dari elemen `rdf:RDF`, dan memerincikan suatu *namespace* URIref `http://www.example.org/terms/` diasosiasikan dengan *prefix* `exterms:`. URIrefs dimulai dengan *string* `http://www.example.org/terms/` digunakan untuk sesuatu dari *vocabulary* yang

didefinisikan oleh *example organization*, example.org. Tanda ">" di akhir baris 3 menunjukan akhir dari `rdf:RDF` *start-tag*. Baris 1-3 adalah kebutuhan umum untuk menyatakan hal ini adalah konten RDF/XML , dan untuk identifikasi *namespaces* telah digunakan diantara konten RDF/XML.

Baris 4-6 menampilkan RDF/XML untuk spesifik tertentu, yaitu sebuah *description*, dan merupakan *about* subjek dari suatu statement (pada kasus ini adalah `http://www.example.org/index.html)`. `rdf:Description` *start-tag* pada baris 4 menunjukan mulainya suatu *description* dari *resource*, dan kemudian identifikasi *resource*, suatu *statement* adalah *about* (subjek dari *statement*) menggunakan atribut `rdf:about` untuk merincikan URIref dari suatu *resource* subjek. Baris 5 menyediakan sebuah *property element*, dengan QName `exterms:creation-date` sebagai *tag*-nya, untuk merepresentasi predikat dan objek dari suatu *statement*. QName `exterms:creation-date` dipilih sehingga menambah *local name* `creation-date` untuk URIref dari *prefix* `exterms:` (`http://www.example.org/terms/`) memberi predikat *statement* URIref `http://www.example.org/terms/creation-date`. Isi dari elemen *property* adalah objek dari *statement*, suatu *plain literal* `August 19, 1999` (*value* suatu *property* creation-date dari *resource* subjek). Elemen ini disimpan di dalam suatu elemen `rdf:Description`, menunjukkan bahwa *property* ini menerapkan kepada spesifik *resource* di dalam atribut `rdf:about` dari elemen `rdf:Description`. Baris 6 menunjukan akhir dari elemen `rdf:Description`. Terakhir adalah baris 7 menunjukkan akhir dari elemen `rdf:RDF`. [11]

2.3.3 RDF Schema

RDF Schema digunakan untuk mendekripsikan *vocabulary* pada RDF, dimanfaatkan untuk menyatakan *class, property, value* dan kita dapat membuat *class hierarchies* untuk mengklasifikasikan dan mendeskripsikan objek. Berikut di bawah ini merupakan daftar *vocabulary* inti dari RDF Schema, yaitu untuk *class* dan *property types*. [5]

Tabel 2.1 *Vocabulary* Untuk *Class* RDF Schema [5]

Class Name	*Comment*
rdfs:Resource	Class *resource,* everything
rdfs:Class	Class of classes
rdfs:Literal	Class of literal value (strings, integers)
rdfs:Datatype	Class of RDF Datatypes
rdf:Property	Class of RDF Properties
rdf:XMLLiteral	Class of XML literals values
rdf:Statement	Class of RDF statements.
rdf:Bag	Class of unordered containers.
rdf:Seq	Class of ordered containers.
rdf:Alt	Class of containers of alternatives.
rdfs:Container	Class of RDF containers.
rdfs:ContainerMembershipProperty	Class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	Class of RDF Lists.

Tabel 2.2 *Vocabulary* Untuk *Properties* RDF Schema [5]

Property	*Comment*	*Domain*	*Range*
rdf:type	Subject is an instance of a class	rdfs:Resource	rdfs:Class
rdfs:subClassOf	Subject is a subclass of a class	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	Subject is a subproperty of a property	rdf:Property	rdf:Property
rdfs:domain	Domain of subject property	rdf:Property	rdfs:Class
rdfs:range	Range of subject property	rdf:Property	rdfs:Class
rdfs:label	Name for the subject	rdfs:Resource	rdfs:Literal
rdfs:comment	Description of subject resource	rdfs:Resource	rdfs:Literal

Tabel 2.3 *Vocabulary* Untuk *Properties* RDF Schema [5] (lanjutan)

rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource

2.3.4 Keunggulan RDF

Keunggulan yang paling utama dari penggunaan RDF adalah *data interoperability*. Melalui berbagai macam *processor* atau *extractor*, RDF dapat mengambil dan menyampaikan metadata atau informasi dari sumber yang bersifat *unstructured (text), semi-structured* (dokumen HTML), atau *structured* (*standard database)*. Hal ini menjadikan RDF disebut sebagai "*universal solvent*" untuk representasi struktur data. Terdapat kemudahan untuk menggabungkan *dataset* atau atribut baru serta mengumpulkan sumber data yang berlainan ke dalam satu sumber.[1]

Kegagalan utama dari *data integration* terjadi ketika *data relationship*s sudah terbentuk, data tersebut akan tetap seperti itu dan tidak bisa dengan mudah diganti. Kegagalan terjadi pada *data management system* yang masih konvensional.

Relational Database Management (RDBM) *system* sama sekali tidak mampu membantu masalah tersebut. Ketika digunakan untuk transaksi dan keperluan penambahan *record* data (*instance,* atau baris dalam *relational table schema*), RDBM tidak dapat beradaptasi dan fleksibel. Mengapa? Hal ini berkaitan dengan *structural view* dimana semuanya direpresentasikan sebagai tabel yang terdiri dari baris dan kolom, dengan *keys* terhadap struktur datar lainnya, secepat perubahan representasi yang terjadi, koneksi dapat terputus.

Sebagai *framework* untuk *data interoperability*, RDF dan turunannya bersifat fleksibel. RDF dapat memberikan *framework* untuk representasi keseluruhan *instance data* dan struktur atau skema yang menjelaskan dari *record* data yang mendasar hingga keseluruhan domain. Dilihat dari perbedaan pada RDBM *system* dan RDF, RDBM *system* menyimpan dan menyediakan *instance records* (Abox) sementara RDF dan turunannya menyediakan skema fleksibel (Tbox). [1]

2.4 *OWL (Web Ontology Language)*

Dengan adanya OWL, visi *semantic web* yaitu mengenai pengembangan *web* yang akan datang, dimana suatu informasi dijelaskan maknanya secara eksplisit serta *machine-readable* lebih mudah memproses dan mengintegrasikan informasi dalam *web* dapat tercapai. Mengapa? OWL dapat digunakan untuk merepresentasikan secara eksplisit suatu makna dalam *vocabularies* dan relasi diantara keduanya. Representasi suatu terminologi dan interrelasi ini disebut ontologi. OWL memiliki fasilitas lebih untuk mengekspresikan makna dan semantik daripada XML, RDF, dan RDF-S, sehingga OWL melalui bahasanya memiliki kemampuan untuk merepresentasikan mesin yang dapat menginterpretasi konten pada *web*. OWL merupakan pembaharuan dari *DAML+OIL web ontology language*.

Pada *semantic web*, XML digunakan sebagai *surface syntax* untuk dokumen terstruktur, tetapi tidak ada batasan makna semantik pada dokumen tersebut. XML Schema adalah bahasa untuk membatasi struktur dokumen XML dan juga merupakan perluasan XML dengan tipe data. Lalu, RDF merupakan model data

untuk objek (*resource*) dan relasi diantara keduanya, menyediakan semantik sederhana untuk model data ini, dan model data tersebut dapat direpresentasikan dalam sintaks XML. RDF Schema merupakan *vocabulary* untuk menjelaskan *property* dan *classes* dari RDF *resource*, dengan semantik untuk generalisasi hirarki dari *property* dan *class* tersebut. Sedangkan OWL, menambahkan lebih *vocabulary* untuk menjelaskan *property* dan *classes* antara keduanya, relasi antara *class* (misal *disjointness*), kardinalitas (misal "*exactly one"*), persamaan, penjelasan detail mengenai *property*, karakteristik suatu *property*, dan *enumerated class*. [12]

OWL memiliki tiga *sublanguage* yang didisain agar dapat digunakan oleh komunitas tertentu (*implementer* dan *user*), yaitu

1. *OWL Lite* digunakan untuk kebutuhan primer pengguna dalam membuat klasifikasi hirarki dengan pembatasan yang sederhana. Pada OWL *Lite* pembatasan kardinalitas diperbolehkan hanya 0 atau 1, memiliki kompleksitas yang lebih rendah dibanding OWL DL.
2. *OWL DL* termasuk semua konstruksi bahasa OWL, tetapi hanya bisa digunakan pada batasan tertentu (misalnya ketika suatu *class* bisa menjadi *subclass* dari beberapa *classes*, *class* tersebut tidak dapat menjadi sebuah *instance* dari *class* lain). *OWL DL* berhubungan dengan *description logics*, sebuah bidang yang mempelajari logika yang membentuk pondasi formal dari OWL.
3. *OWL Full* merupakan *sublanguage* yang paling maksimal dalam ekspresi dan kebebasan sintaks dari RDF tanpa jaminan komputasi. Misalnya, dalam *OWL Full* sebuah *class* dapat diperlakukan sebagai kumpulan individu dan sebagai suatu individu itu sendiri.

Pemilihan antara *OWL Lite* dengan *OWL DL* tergantung kepada tingkat dimana pengguna membutuhkan kontruksi yang lebih ekpresif dari *OWL DL,*sedangkan pemilihan *OWL DL* dengan *OWL Full* tergantung kepada tingkat dimana pengguna membutuhkan fasilitas *meta-modeling* dari RDF Schema (misal

mendefinisikan *classes* dari *classes*, atau menghubungkan *property* kepada *classes*).[12]

Kebanyakan elemen dari OWL ontologi memperhatikan *classes, properties, instance of classes,* dan relasi antara *instance*. Berikut di bawah ini penjelasan dari elemen-elemen utama OWL. [16]

1. *Simple Named Classed*

Tiap individu pada *OWL world* adalah anggota dari *class* `owl:Thing`. Sehingga tiap class yang didefinisikan secara implisit adalah *subclass* dari `owl:Thing`. OWL juga menjelaskan *empty class*, `owl:Nothing`. Misal domain *wines*, dengan *root classes* : `Winery`, `Region`, dan `ConsumableThing`.

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

2. *Individuals*

Merupakan anggota dari *classes* atau dapat dikenal dengan sebutan *instance*.

```
<Region rdf:ID="CentralCoastRegion" />
```

Atau dengan makna yang identik, dapat ditulis seperti ini

```
<owl:Thing rdf:ID="CentralCoastRegion" />

<owl:Thing
rdf:about="#CentralCoastRegion">
   <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

3. *Properties*

Properties merupakan relasi *binary*. Terdapat dua jenis *property* dalam OWL, yaitu:

- *datatype properties*, relasi antara *instance* dari classes dengan RDF *literals* dan XML Schema *datatypes.*

- *object properties*, relasi antara *instance* dari dua *classes* .

Pada saat mendefinisikan *property*, terdapat beberapa aturan yang dapat ditambahkan. Domain dan *range* dapat dispesifikasikan, *property* dapat dikhususkan lagi menjadi *subproperty* dari suatu *property* yang ada. Berikut contoh penggunaan *property*.

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</owl:ObjectProperty>
```

Dari sintaks tersebut, *property* `madeFromGrape` memiliki domain `Wine` dan *range* `WineGrape`. Sehingga, hal ini menghubungkan *instance* dari *class* `Wine` dengan *instance* dari *class* `WineGrape`. Sama seperti *class, property* dapat berhirarki.

```
<owl:Class rdf:ID="WineDescriptor" />

<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
  ...
</owl:Class>

<owl:ObjectProperty rdf:ID="hasWineDescriptor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range  rdf:resource="#WineDescriptor" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineColor" />
  ...
</owl:ObjectProperty>
```

`WineDescriptor` *properties* berelasi dengan wines (meliputi warna dan rasa). `hasColor` adalah *subproperty* dari *property* `hasWineDescriptor` , dengan *range* yang dibatasi kepada `WineColor`. Relasi `rdfs:subPropertyOf` dalam kasus ini menjelaskan

bahwa apapun yang berhubungan dengan *property* `hasColor` dengan *value* X juga memiliki *property* `hasWineDescriptor` dengan *value* X.

Terdapat mekanisme yang dapat digunakan untuk spesifikasi *property*, yaitu *property characteristic* (yang memberikan mekanisme maksimal untuk meningkatkan *reasoning* mengenai *property*), dan *property restrictions* (digunakan untuk membatasi *range* suatu *property* dalam konteks spesifik). Pada *property characteristic* caranya dengan menambahkan keterangan `inverseOf`, `TransitiveProperty`, `SymmetricProperty`, `FunctionalProperty`, dan `InverseFunctionalProperty`. Sedangkan pada *property restriction* terdapat tiga macam, yaitu *quantifier, cardinality,* dan *hasValue.* Pada *quantifier* digunakan `allValuesFrom` dan `someValuesFrom`. [16]

2.5 *Previous Works*

Berikut ini merupakan beberapa penelitian atau proyek menggunakan teknologi *semantic web* dalam pengembangan *knowledge management system* yang telah dilakukan sebelumnya. Penelitian tersebut diantaranya, yaitu

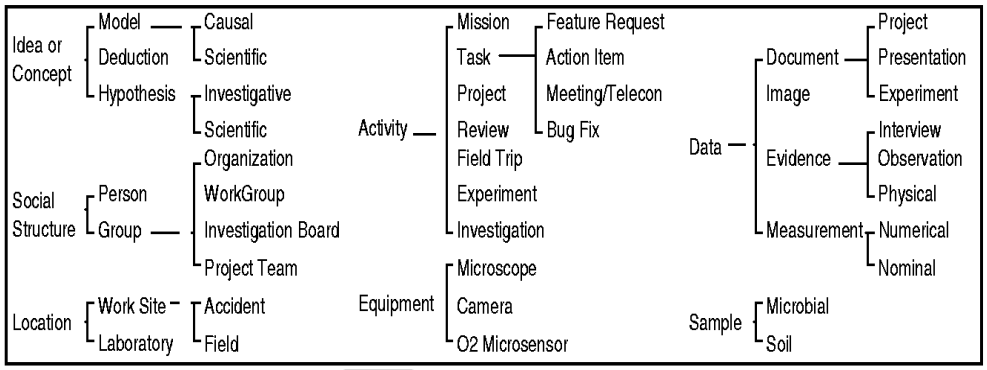
2.5.1 SemanticOrganizer

SemanticOrganizer merupakan *collaborative knowledge management system* yang digunakan untuk mendukung proyek-proyek yang dikerjakan oleh NASA. Sistem ini merupakan *repository* informasi terstruktur secara semantik, contohnya akses terhadap semua produk-produk yang berhubungan dengan proyek yang sedang berjalan. Pengguna dapat mengunggah dokumen, data, *images,* dan informasi relevan lainnya. *Repository* tidak hanya menyimpan *file-file*, tetapi juga metadata untuk menjelaskan konsep domain yang memberikan konteks untuk suatu produk.[9]

Pada awalnya, tim NASA memiliki beberapa *requirement* spesifik untuk mendukung kebutuhan mendasar *information-sharing*, yaitu

- *Sharing of heterogeneous technical information* : semua tim harus saling menukarkan beberapa jenis ilmu spesialis dan informasi teknik ke dalam format yang berbeda.
- *Detailed descriptive metadata* : semua tim harus menggunakan terminologi teknis yang pasti untuk mendokumentasikan aspek yang berupa asal-usul informasi, kualitas, dan metodologi pengumpulan.
- *Multi-dimensional correlation and dependency tracking*: semua tim butuh untuk saling berhubungan dan mengeksplorasi informasi teknis secara simultan, serta membuat koneksi terhadap informasi baru secara cepat.
- *Evidential reasoning*: semua tim harus bisa menyimpan hipotesis bersamaan dengan mendukung dan membuktikan fakta yang salah, serta analisis hubungan sebab penyebab (*causal relationship).*
- *Experimentation*: semua tim harus menguji hipotesis dengan mengumpulkan pengukuran secara sistematis yang dihasilkan dari penggunaan *tools* dan sensor ilmu pengetahuan khusus.
- *Security and access control*: informasi dikumpulkan dan dianalisis dapat menjadi kepemilikan yang tinggi, sensitif, dan/atau pembatasan secara legal.
- *Historical record*: semua tim proyek harus mendokumentasikan proses kerja dan produk yang dihasilkannya, termasuk didalamnya apa yang sukses dan gagal.

SemanticOrganizer terdiri dari *repository hypermedia* dengan jaringan terstruktur secara semantik yang berisikan *information items*. Tiap *repository item* mewakili sesuatu yang relevan dengan tim proyek (misalnya orang tertentu, tempat, hipotesis, dokumen, sampel fisik, subsistem, *meeting, event*, dan lain-lain.). Sebuah *item* terdiri atas satu himpunan metadata, sebuah *attached file* yang berisi *image, dataset*, dokumen, atau produk elektronik relevan lainnya. *Item-item* tersebut dihubungkan secara luas *via semantically labeled relations* untuk memudahkan akses terhadap bagian informasi yang terkait. [9]

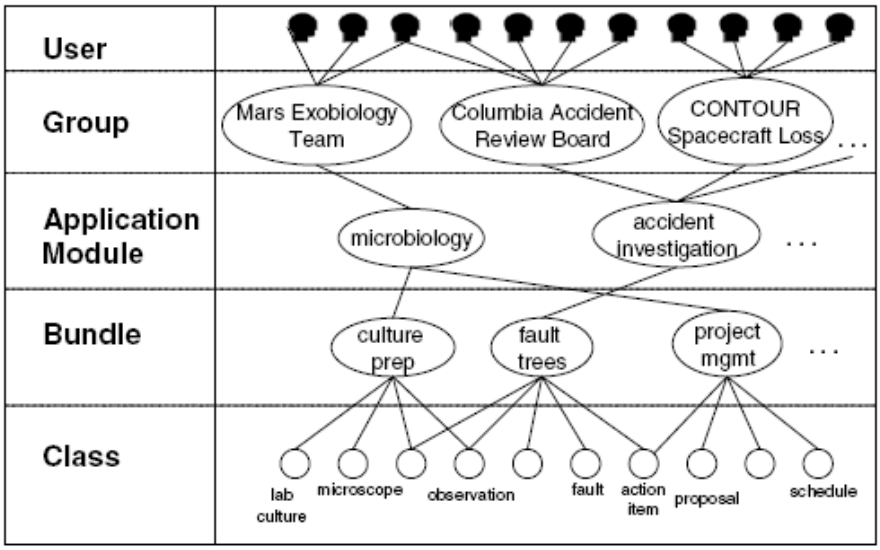


Gambar 2.6 Representasi *Classes* dari Master Ontologi SemanticOrganizer [9]

Master ontologi di atas menjelaskan *classes* dari *item-item* untuk aplikasi SemanticOrganizer dan mendefinisikan bagian dari *item* tersebut berupa *link-link* yang bisa digunakan untuk mengekspresikan hubungan antara *item-item*. Sistem ini dibangun dengan menggunakan Java dan ontologinya serta hubungan *instances* disimpan di basis data MySQL, komponen *inference* dibangun di atas Jess (*rule engine* untuk Java *platform*).

SemanticOrganizer mencakup distribusi *e-mail* dan fasilitas pengarsipan yang memungkinkan tim-tim untuk membuat daftar *e-mail* secara *ad-hoc*. Pengguna sistem ini dapat membuat dan menghubungkan *item-item* menggunakan *servlet-driven Web interface* yang mampu melakukan navigasi pada *repository*. Dalam membuat *interface*, sistem ini menggunakan HTML dan JavaScript. [9]

Pada sistem SemanticOrganizer terdapat mekanisme yang disebut dengan mekanisme personalisasi aplikasi. Intinya adalah terdapat beberapa orang yang berfungsi sebagai *knowledge modeler* yang bekerja di tiap-tiap kelompok baru (pengguna) untuk memahami kebutuhan mereka. *Modeler* inilah yang dapat menambah atau menggunakan kembali *class* ontologi untuk membentuk aplikasi yang sesuai dan cocok untuk tim tersebut. Penetapan *classes* ontologi terhadap pengguna melalui suatu proses yang diilustrasikan sebagai berikut.



Gambar 2.7Pemetaan *Classes* Ontologi Terhadap Pengguna *via Bundles, Application Modules*, dan *Group* [9]

Pada level paling bawah, terdapat *class* yang dikelompokkan menjadi *bundle*, dimana masing-masing *bundle* merupakan kumpulan dari *class* relevan dengan fungsi *task* yang spesifik. Disamping mengelompokkan *class* yang berelasi, *bundles* juga menyediakan suatu mekanisme untuk *aliasing classes* untuk mengontrol tampilan *interface* kepada pengguna. Kemudian, kumpulan-kumpulan dari *bundle* dikelompokkan bersama sebagai *application module*. Modul ini berisikan semua *bundle* yang berkoresponden terhadap *task* yang relevan terhadap aplikasi yang diberikan. Pada level tertinggi, modul-modul ini di-*assign* menjadi *group* dari para pengguna, yang akhirnya melalui *group* ini pengguna secara individu dapat mengakses *classes* yang sesuai dengan aplikasinya.

Terdapat dua komunitas pengguna primer, yaitu *the NASA scientific community* (dimana sistem dikenal dengan ScienceOrganizer) dan *the NASA safety and accident investigation community* (dikenal dengan InvestigationOrganizer atau IO). [9]

2.5.2 Organik *Project*

Organik *Project* merupakan proyek yang dikembangkan oleh Uni Eropa. Tujuan dari Organik *Project* ini adalah untuk mengembangkan dan meneliti suatu inovasi baru *knowledge management system* yang memiliki fitur semantik pada aplikasi *enterprise social software*. Sistem ini mengumpulkan informasi yang dapat disebarkan antara satu atau beberapa perusahaan yang berkolaborasi. Organik *Project* dapat disesuaikan dengan *requirements* pada perusahaan kecil dan bersifat *knowledge-intensive* (seperti lembaga riset). Oleh karena itu, sistem ini memiliki dua macam fokus yang akan diperoleh melalui aktivitas-aktivitas riset, yaitu:

1. Organik *Project* bertujuan untuk mengembangkan sebuah kerangka teknologi dan arsitektur teknis dari *knowledge management* yang inovatif dan berfokus kepada pekerjaan terkini suatu perusahaan kecil bersifat *knowledge-intensive* di Eropa, memadukan elemen-elemen yang berbeda dari *enterprise social software* dan teknologi *semantic web*.
2. Organik *Project* bertujuan untuk mengembangkan dasar teoritis dari *knowledge management* yang inovatif dan berfokus kepada pekerjaan terkini suatu perusahaan kecil bersifat *knowledge-intensive* di Eropa, memadukan elemen-elemen yang berbeda dari sktruktur organisasi. [14]

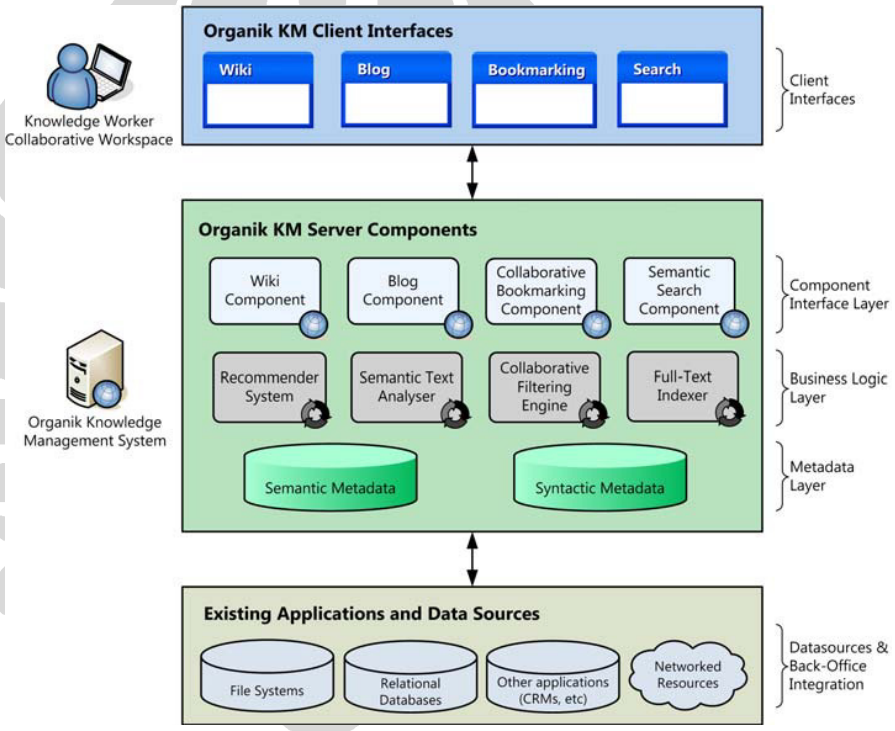
Berikut ini merupakan karakteristik khusus dari perusahaan yang berbasis *knowledge-intensive* yang berperan sebagai penghambat terhadap implementasi *knowledge management system* yang formal dan menyeluruh, yaitu:

- Kurangnya sumber-sumber untuk memperoleh, mengimplementasi, dan memelihara *knowledge management system*.
- Komunikasi informal antara individu dan *ad hoc people-centric operations.*
- Kurangnya biaya untuk pelatihan dan penyebarluasan program-program edukasi. [14]

Ketiga hal di atas menyebabkan adanya suatu kebutuhan untuk mengembangkan lingkungan digital yang baru untuk memperbaiki pengetahuan organisasi. *Enterprise social software* cocok untuk memfasilitasi pengelolaan pengetahuan pada lingkungan

yang bersifat *idiosyncratic* suatu firma kecil berbasis *knowledge* di Eropa. Penggunaan *enterprise social software* dengan teknologi *semantic web* menjadikan suatu sistem memiliki keunggulan dalam memproses informasi lebih canggih.

Kunci utama dari Organik *Project* adalah membangun suatu sistem yang memiliki arsitektur teknologi Web 2.0 dengan *interface* dan metode *data integration* yang bersifat *easy-to-use* dengan penggabungan *semantic web*. Berikut ini merupakan ilustrasi konsep arsitektur dari Organik KM.



Gambar 2.8 Rencana Konseptual Arsitektur untuk *Semantically-enriched Enterprise Social Software* [14]

Kumpulan dari *Organik KM Client Interfaces* meliputi Wiki, *Blog*, *Social Bookmarking and Search Component* yang bersama-sama membangun suatu lingkungan kerja yang kolaboratif untuk *small and medium-sized enterprises* (SME) *knowledge worker*. Masing-masing dari *client interface* berkoresponden kepada komponen bagian *server* pada layer selanjutnya, yaitu *Component Interface Layer*. Selanjutnya adalah komponen-komponen yang digunakan untuk membangun

Business Logic Layer dari Organik KM *server* terdiri dari:

- *Recommender System*, penyedia utama untuk layanan ontologi, yaitu penyaranan *tag* dan klasifikasi serta penyaranan terhadap informasi yang berhubungan.
- *Semantic Text Analyser*, pemrosesan teks dari *item* informasi di dalam sistem, menggunakan algoritma linguistik dan statistikal. Hasil ini dari analisis ini adalah suatu identifikasi dari adanya entitas seperti penggunaan tag atau entitas dari basis data yang sudah ada (misalnya *customer*) yang mana dapat digunakan oleh *Recommender System* untuk menyarankan tag dan klasifikasi mana yang cocok.
- *Collaborative Filtering Engine*, kemampuan individu untuk memanfaatkan pengalaman dalam melakukan pencarian dengan grupnya. Pencarian secara individu apakah dengan pencarian kata kunci secara biasa atau pencarian menggunakan kemampuan canggih dari komponen *semantic search* akan menghasilkan daftar dokumen-dokumen yang diterima. Hal ini meninggalkan *task* pengguna untuk memilih dokumen yang paling relevan dari daftar yang tersedia.
- *Full-text Indexer* , kemampuan mendapat dokumen secara efisien yang mengandung *query text snippets/keywords.*

Metadata Layer merujuk kepada tempat penyimpanan yang digunakan untuk *syntactic* dan *semantic metadata* dalam mendukung fungsionalitas seluruh komponen bagian *server*, sedangkan *Datasources and Back-Office Interagration Layer* merujuk kepada sistem informasi bisnis dan segala macam bentuk penyimpanan sumber dimana suatu perusahaan bergantung untuk kegiatan sehari-harinya. [4, 13]

Organik KM ini memiliki kerangka SLATES untuk arsitektur sistemnya yang merupakan akronim dari:

1. *Search,* menyediakan mekanisme pencarian informasi.
2. *Links,* bantuan untuk *knowledge worker* dalam mencari kebutuhan akan

pengetahuan serta menjamin munculnya struktur untuk konten *online*.

3. *Authoring*, memungkinkan *knowledge workers* untuk *sharing* opininya dengan pihak eksternal.
4. *Tags,* menampilkan navigasi alternatif dalam memanfaatkan kategori tanpa hirarki dari konten intranet.
5. *Signals,* secara otomatis memberikan sinyal kepada *knowledge worker* adanya konten relevan dan baru.

Jika dilihat asosiasi antara komponen pada SLATES dan arsitektur sistem yang dimiliki Organik KM, maka akan dapat disimpulkan dengan tabel berikut ini. [4]

SLATES Framework	Proposed Architecture
Search	Semantic Search
Links	Collaborative Bookmarking
Authoring	Wiki and Blog spaces
Tags	Collaborative Bookmarking, Wiki and Blog spaces
Extensions	Recommender System
Signals	Really Simple Syndication (RSS)

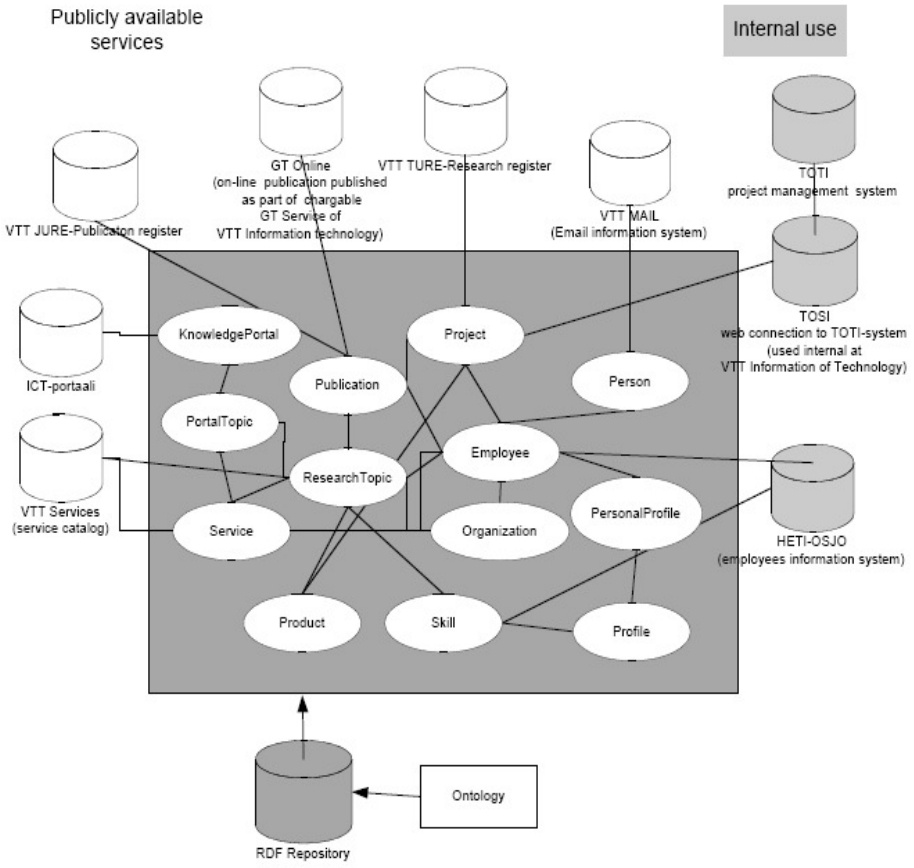
Gambar 2.9 Asosiasi Antara Komponen di SLATES dengan Rencana Arsitektur [4]

2.5.3 VTT's ICT Knowledge Portal

Portal ini dibuat oleh VTT Information Technology yang berasal dari Finlandia. Ide berupa suatu ontologi dapat mengelola informasi tentang *project*, *people, document,* dan *product.* Keseluruhan informasi ini dapat di-*query* dari basis data yang telah ada, dan hanya informasi baru yang disimpan di RDF *repository* yang dibangun pada proyek ini.

Ontologi dari portal ini terdiri dari sebelas *classes*, yaitu *KnowledgePortal, PortalTopic, ResearchTopic, Service, Skill, Profile, Project, Publication, Product, Organization,* dan *Person*. VTT ICT memiliki empat area pengetahuan, yaitu *Enabling technologies, Telecommunication system, Intelligent system and service,* dan *Tools for information society.*

Sumber pengetahuan penting yang tersedia berasal dari sistem manajemen proyek, registrasi proyek penelitian, registrasi publikasi, publikasi *online*, sistem informasi kepegawaian, katalog pelayanan, dan sistem informasi *e-mail.* Berikut merupakan gambaran hubungan antara informasi yang tersedia pada sumber pengetahuan dan *classes* ontologi. [3]



Gambar 2.10 Hubungan Antara *Existing Knowledge Source* dengan *Ontology Resource* [3]

Pengembangan ontologi menggunakan bahasa RDF. Skema RDF dibuat dengan Protégé-2000 *ontology editor*. Semua *instances* disimpan dalam file RDF. Kedua skema RDF dan file data RDF diimpor ke *Sesame RDF repository*. File-file diimpor melewati *Sesame web interface*. *Sesame repository* di-*install* pada MySQL *database*. Seluruh data disimpan ke dalam basis data melalui *Sesame*. *Queries* terhadap *knowledge base* dievaluasi dengan menggunakan *Sesame's RQL editor.*

People dan *project* dapat dicari berdasarkan topik penelitian atau kemampuan. *Publication* dan *product* dapat di-*query* berdasarkan topik penelitian. *Querying* dapat dibatasi berdasarkan *knowledge portal topic* atau bagian dari organisasi. Sedangkan pencarian keahlian berdasarkan keahlian dengan dua opsi, yaitu pencarian langsung pada keahlian itu sendiri dan keahlian yang berhubungan dengan topik penelitian.[3]

2.5.4 SWED (Semantic Web Environmental Directory)

SWED (Semantic Web Environmental Directory) adalah suatu proyek yang dibiayai oleh Uni Eropa sebagai bagian dari proyek SWAD-E (Semantic Web Advanced Development for Europe) yang bertujuan untuk mendukung W3C Semantic Web di Eropa, menyediakan penelitian, demonstrasi dan memastikan teknologi *semantic web* mencapai target yang lebih baik ke dalam *networked computing*. SWED berupa prototipe yang menggambarkan lingkungan organisasi dan proyek. [21] Akan tetapi fokus utama SWED dimotivasi oleh beberapa faktor, yaitu

- Terdapatnya kebutuhan pada saat survei informasi *wildlife and biodiversity* di Inggris
- Pada saat itu tidak ada direktori umum yang berlaku pada lingkungan Inggris, yang menyediakan informasi mendetail pada proyek baik yang mencakup subjek dan sektor serta termasuk didalamnya proyek-proyek dan gagasan-gagasan yang ditemui dalam suatu organisasi.
- Direktori yang dibuat akan berguna pada berbagai kelompok yang berbeda dari orang-orang dan organisasi dari bermacam-macam sektor pendidikan, lingkungan, konservasi, pemerintahan lokal dan nasional, bisnis, penelitian pengetahuan, media, dan lain-lain.

- Proyek memiliki dukungan dari beberapa lingkungan organisasi termasuk *The Environment Council*, yang memproduksi publikasi *Who's Who in the Environment* sekita tahun 90-an.

- Terdapat bermacam-macam proyek bekerja untuk menyatukan berbagai jenis tipe data yang berkaitan

Tujuan atau sasaran utama yang ingin dicapai dari pengembangan proyek SWED sesuai dengan faktor-faktor yang memotivasinya, diantaranya:

- Membuat demonstrasi yang efektif dari teknologi *semantic web* serta pendekatan-pendekatan sebagai bagian dari proyek SWAD-E

- Membuat sistem yang menyediakan *tools* prototipe untuk pembuatan dan pemeliharaan direktori suatu organisasi dan informasi lengkap mengenai organisasi tersebut.

- Sistem sebaiknya memberikan solusi terhadap berbagai jenis batasan dan problem dari pendekatan tradisional dan yang sudah ada, sebagaimana yang telah diidentifikasi sebagai bagian dari aktifitas survei di Inggris. [21]

Berikut ini merupakan *architecture layer* yang merepresentasikan prototipe SWED, terdiri dari [15]:

- *Domain specific representation* merupakan gambaran umum tentang spesifik domain dan *thesauri* portal yang ingin dikembangkan. Dalam proyek SWED domain berupa *directory of wildlife and environmental organizations.*

- *Cross domain ontologies,* beberapa domain dapat diperluas dengan pendekatan *semantic portal* dalam melakukan *sharing* suatu kebutuhan dengan merepresentasikan konsep umum seperti *person, organization*, dan *events*. Prototipe SWED sendiri terdiri dari dua *class*, yaitu *organization* dan *project*.

- *Portal infrastuctures* dibagi menjadi tiga komponen, yakni

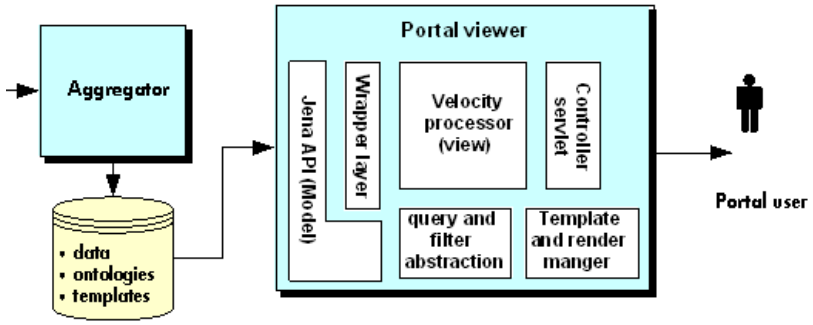
    - Kumpulan konvensi yang dibutuhkan untuk mengkombinasikan RDF *instance data* dan *OWL ontologies* yang digunakan untuk men-*generate human accessible views, edit screens, and search tools.* Hal ini berfungsi dalam bagaimana data sebaiknya dikelompokkan ke dalam *viewable pages* dan bagaimana data sebaiknya diurut dan ditampilkan di halaman *web*. SWED memiliki dua *class* yang ditampilkan pada halaman *web* dengan enam *facet*, yaitu *topic of interest, organization type, activity, project type, operational area,* dan *name*. Terdapat lima jenis *search tools*, yaitu *faceted browse, text search, refine search, tree search*, dan *visualize link*.

    - Kebutuhan akan *adequate controls* perlu didukung pada level ini dengan representasi eksplisit suatu kebijakan akses atau validasi level (keakuratan data), moderasi (eliminasi konten yang tidak sesuai), dan privasi (larangan agregasi data sensitif seperti alamat *e-mail*) untuk komunitas tersebut dan akses protokol untuk mendukung pelaksaaan kebijakan ini.

    - Selain bahasa ontologi OWL dapat merepresentasikan model informasi untuk portal, konseptual domain yang relevan juga butuh untuk menjelaskan klasifikasi dalam bentuk *thesauri*.

- *Semantic web*, seluruh informasi di portal direpresentasikan oleh RDF, sedangkan data dan spesifik domain menggunakan OWL *language*. Akses jaringan mengadopsi RDF NetAPI, profil HTTP. [15]

SWED memiliki struktur portal seperti gambar di bawah ini



Gambar 2.11 Struktur Portal SWED [20]

Komponen utama adalah tampilan portal. Tampilan portal ini berjalan sebagai aplikasi *web* pada *Java servlet container* dan menyediakan *web interface* pada *semantic web data* yang ada di portal tersebut. Lalu, komponen yang kedua adalah *aggregator* yang secara periodik *scan* daftar dari sumber situs yang diketahui dan *upload* tiap perubahan RDF data ke portal *database* sehingga dapat di-*display* oleh *viewer*.

Meskipun dalam diagram di atas memperlihatkan bahwa terdapat satu *database*, akan tetapi kenyataannya adalah portal mengambil data dari *multiple files* yang dimasukkan ke dalam memori, seperti layaknya dari sebuah *database* tambahan. Tampilan *templates* berupa *file* sederhana (*local* atau *retrieval via* htpp URLs) dan dimasukkan serta dikelola oleh *template engine*. Ontologi merupakan *local* RDF (RDFS atau OWL) *files* dan dimasukkan ke dalam memori. Biasanya data disimpan di dalam sebuah *database*, tetapi dalam portal sederhana ini disimpan ke dalam memori dari *static file*.

Tampilan portal mengadopsi disain MVC (*Model-View-Controller*). *Model* ditampilkan oleh sebuah *linked* kumpulan dari Java *classes*. *Instance* portal dispesifikasikan oleh *DataSource* yang memberikan akses untuk representasi *filter*

state, kumpulan *facets*, *datasource* dan RDF API yang disediakan oleh Jena *library*. Lalu, komponen *View* menggunakan *Jakarta Velocity template engine* untuk membuat *displayed resource* berdasarkan kepada kumpulan tampilan *templates*. *Controller* adalah Java servlet untuk membangun dan memanggil *velocity template*. Aplikasi *web* portal berjalan pada JSP *container* seperti tomcat. [20]