BAB 4
PELAKSANAAN

Seperti yang telah dijelaskan pada Bab 3 Metodologi Penelitian, secara umum ada enam langkah yang dilakukan dalam pelaksanaan penelitian. Tahapan-tahapan tersebut akan dijelaskan satu-per-satu dalam bab ini.

4.1. Tinjauan Pustaka

Penjelasan mengenai tinjauan pustaka sebagian sudah dipaparkan pada butir 2.3.2. Ontologi *Student Model* yang Pernah Dikembangkan. Di situ dijelaskan bahwa salah satu referensi adalah *user model ontology* LOCO. Ontologi yang didapat adalah dalam bentuk berkas .rdf. Sedangkan penjelasan lain berupa gambar dan tulisan yang representatif untuk memahami ontologi tersebut secara menyeluruh tidak penulis dapatkan. Oleh karena itu penulis berusaha memahami ontologi LOCO dengan meng-*import*-nya pada Protégé lalu mendata semua kelas dan properti yang dimiliki. Penulis mencoba menggunakan *tool* yang dapat meng-*convert source code* menjadi gambar, tetapi ternyata *tool* tersebut tidak cukup *powerfull*. Agar tidak membuang-buang waktu untuk mencari *tools*, penulis memutuskan mendata kelas dan properti sendiri dan membuat gambar ontologi *user model* LOCO secara manual.

Kelas yang terdapat pada *user model ontology* LOCO yang menjadi *sub-class* utama dari `owl:Thing` adalah sebagai berikut: `User`, `UserRole`, `PersonalInformation`, `Group`, `Organization`, `Feedback`, `LearningModule`, `LearningStyleCategory`, `AuthorPreference` dan `LearningStyleTheory`. Gambaran mengenai ontologi *user model* LOCO dapat dilihat pada Lampiran 4 sedangkan secara umum penjelasannya adalah sebagai berikut:

a. `User` merupakan *class* untuk pengguna, yang dibagi menjadi beberapa *subclass*: `Author`, `Learner`, dan `Teacher` di mana masing-masing memiliki `UserRole` (didefinisikan dengan *object property* `hasRole`) dan `PersonalInformation`.

b. User merupakan anggota dari Group dan Organization. Group terdiri dari ResearchGroup dan StudyGroup, sedangkan Organization terdiri dari ResearchCentre dan University.
c. Learner memiliki LearningStyle. Terdapat beberapa LearningStyleCategory: LS_Active-Reflective, LS_Inductive-Deductive, LS_Sensing-Intuitive, LS_Sequential-Global, dan LS_Visual-Verbal. LearningStyleCategory merupakan pengkategorian berdasarkan LearningStyleTheory tertentu.
d. Author memiliki AuthorPreference.
e. Teacher mengajarkan LearningModule.
f. User yang bukan Teacher dimintai Feedback.

User model ontology juga menggunakan ontologi untuk pemodelan kompetensi, ontologi untuk pemodelan *learner performance*, dan ontologi untuk pemodelan preferensi pengguna. Ontologi-ontologi tersebut untuk menggambarkan *performance* yang dimiliki pengguna serta preferensi mereka.

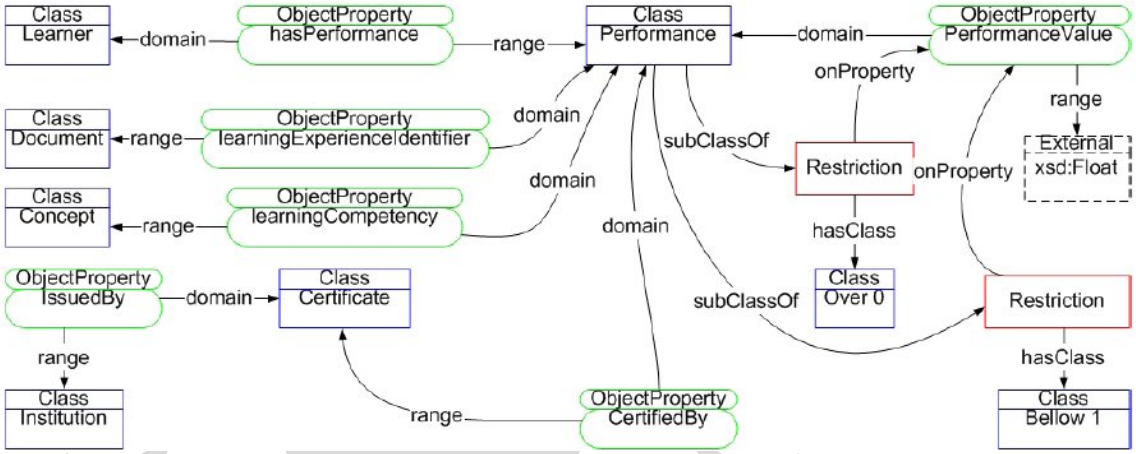
a. Performance dihubungkan dengan *class* Competency melalui *object property* learning_competency, sedangkan Performance dihubungkan dengan ContentUnit melalui *object property* learning_resource. learning_resource adalah *resource* yang digunakan untuk pembelajaran. *Student performance* sebagian dipengaruhi oleh kualitas kecocokan dari *resource* pembelajaran untuk tugas yang diberikan.
b. Preference terdiri dari ConceptPreference dan Language Preference di mana masing-masing memiliki conceptRef terhadap Concept dan languageRef terhadap Language. Preference yang satu dengan yang lain masing-masing memiliki kepentingan yang lebih atau kurang dari yang lainnya. Setiap User memiliki preference.

Dua referensi lain adalah tulisan ilmiah dengan judul "Reasoning and Ontologies for Personalized E-Learning in The Semantic Web"[5] serta "Use of Ontology-

[5]http://www.ifets.info/journals/7_4/10.pdf

Based Student Model in Semantic-Oriented Access to The Knowledge in Digital Libraries"[6].

Pada tulisan pertama penulis meninjau *ontology for learner performance*. Ontologi tersebut serupa dengan yang digunakan LOCO, karena ontologi tersebut berdasarkan kategori *performance* dari PAPI (IEEE Personal and Private Information) yang juga digunakan oleh LOCO. Gambar mengenai ontologi ini dapat dilihat pada Gambar 4.1.

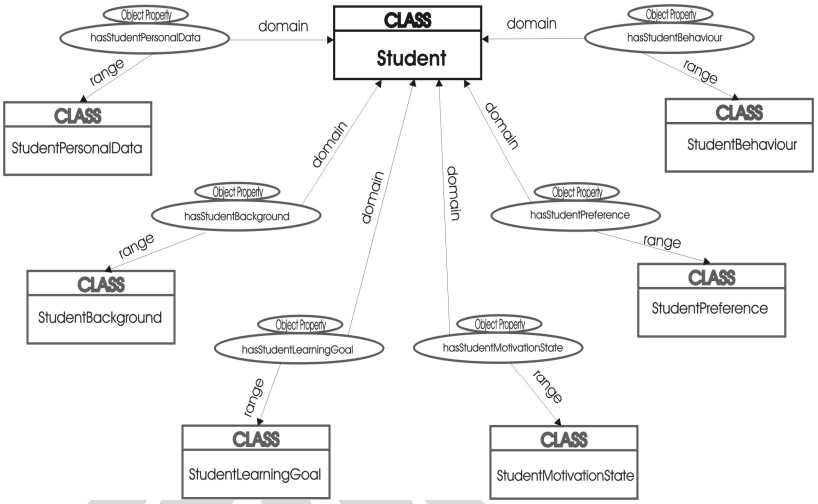


Gambar 4.1. Contoh *Ontology for Learner Performance*

Reasoning and Ontologies for Personalized E-Learning in The Semantic Web

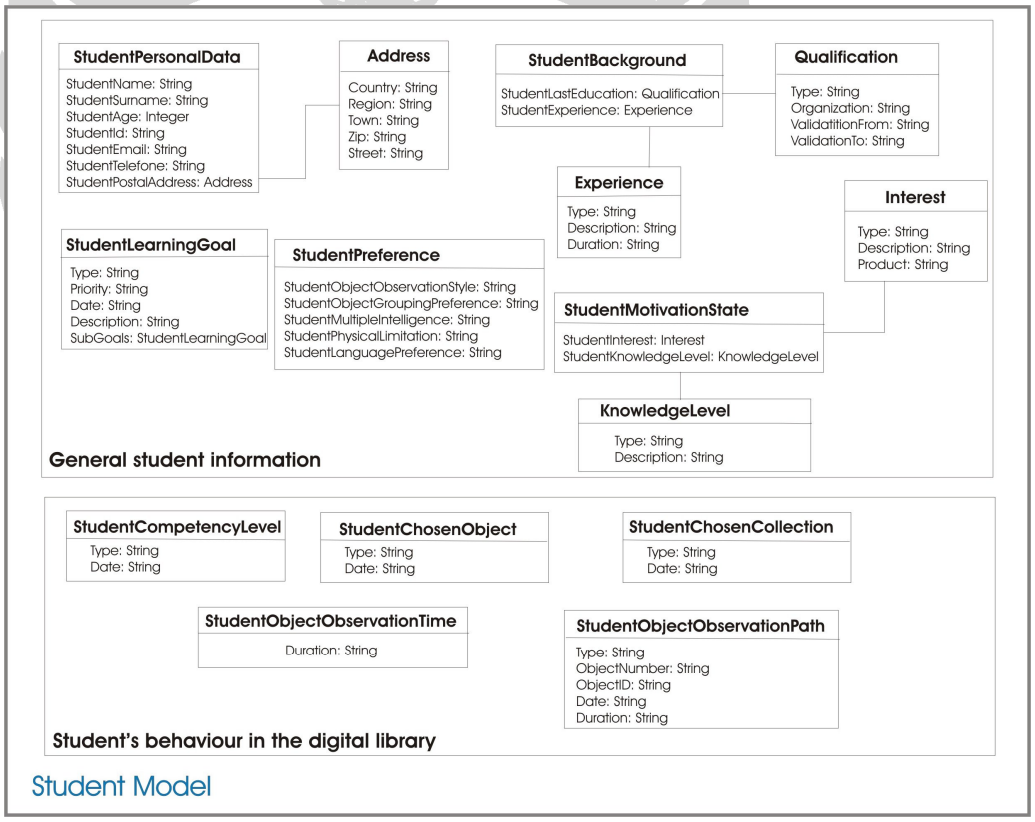
Pada tulisan kedua penulis meninjau *student ontology* yang memiliki 7 *class* utama: `Student`, `StudentBehaviour`, `StudentPreference`, `StudentMotivationState`, `StudentLearningGoal`, `StudentBackground`, dan `StudentPersonalData`. Ontologi ini dikhususkan untuk *digital libraries* pada pembelajaran online. Penulis terutama tertarik mengenai rancangan yang dibuat berkaitan dengan *student data* pada ontologi ini. Gambar model ontologi ini dapat dilihat pada gambar 4.2 dan 4.3.

[6]http://mdl.cc.bas.bg/dessi/Desislava%20Paneva_files/Ontology_based_Student_Modelling_Desislava_Paneva_092006_HUBUSKA_workshop.doc



Gambar 4.2. Contoh *Student Ontology*

Use of Ontology-Based Student Model in Semantic-Oriented Access to The Knowledge in Digital Libraries



Gambar 4.3. Contoh *Student Ontology* – Struktur *Class*

Use of Ontology-Based Student Model in Semantic-Oriented Access to The Knowledge in Digital Libraries

4.2. Pengumpulan Kebutuhan

Kebutuhan yang dikumpulkan adalah hasil pembelajaran ontologi maupun pendefinisian sejak awal. Sehingga tahapan ini dapat dianggap dilaksanakan sebelum atau sesudah tinjauan pustaka.

Berdasarkan pendefinisian kebutuhan dari dosen pembimbing, *student model ontology* yang ingin dikembangkan meliputi *learning style* dan *performance student.* Selain itu juga dibutuhkan *student data* sebagai referensi. *Performance student* untuk menggambarkan secara umum *prior knowledge* maupun kecerdasan siswa sebagai aspek personal pembelajar yang telah disebutkan dalam Bab 2 Tinjauan Pustaka.

Student performance dapat dilihat dari IPK, IPS, nilai-nilai mata kuliah pra syarat, maupun tes-tes yang dijalankan. Penulis menyimpulkan bahwa semua nilai yang didapat oleh mahasiswa mencerminkan *performance* dari masing-masing mahasiswa.

Ditekankan, dalam *student model ontology* yang ingin dikembangkan hanya diperlukan objek *student* dan tidak diperlukan objek *user* lainnya.

4.3. Analisis Ontologi

Secara umum perbandingan model-model ontologi yang menjadi bahan pertimbangan serta kebutuhan yang telah didefinisikan dapat dilihat pada Tabel 4.1. Namun, pada model-model ontologi yang menjadi bahan pertimbangan dengan kebutuhan yang didefinisikan, ada beberapa hal yang sesuai maupun tidak sesuai, walaupun mendeskripsikan hal yang sama misalnya sama-sama mendeskripsikan *student data* atau *student information*. Rancangan yang sesuai selanjutnya dapat diambil, sedangkan yang tidak sesuai tidak diambil.

Tabel 4.1. Garis Besar Perbandingan Ontologi dari Segi Deskripsi Tentang *Student/Learner*

Deskripsi Ontologi	a	b	c	Kebutuhan
Student Learning Style	V			V
Student Performance	V	V		V
Student Preference	V		V	
Student Personal Data	V		V	V
Student Background			V	
Student Learning Goal			V	
Student Motivation State			V	
Student Behaviour			V	
Observation		V		
Student Data/Information	V			V
(a)user model LOCO				
(b)ontologi pada "Reasoning and Ontologies for Personalized E-Learning in The Semantic Web"				
(c)ontologi pada "Use of Ontology-Based Student Model in Semantic-Oriented Access to The Knowledge in Digital Libraries"				

Rancangan model ontologi yang sesuai adalah LOCO *ontology* merancang *learner* memiliki *learning style*. LOCO *ontology* juga merancang *user* memiliki *performance*, namun penggambaran *performance* yang dimiliki oleh *user* tidak seperti pendefinisian *performance student* yang diinginkan. *Performance* yang digambarkan LOCO diukur dengan `performance_coding`, `performance_metric`, dan `performance_velue`. Seperti yang telah disebutkan, rancangan yang dibutuhkan mengukur *performance* dengan indeks prestasi maupun nilai mata kuliah dan tes. *Performance* yang digambarkan LOCO serupa dengan *ontology for learner performance* pada referensi kedua yang sama-sama berdasarkan PAPI. Namun demikian, model tersebut tidak sesuai dengan kebutuhan.

`PersonalInformation` pada *user model ontology* LOCO dapat diambil sebagai *student data*, serupa dengan `StudentPersonalData` pada *student ontology* untuk

digital libraries. Hubungan *learner* dengan *group* maupun *organization* pada LOCO tidak dibutuhkan untuk personalisasi pembelajaran online di Fakultas Ilmu Komputer Universitas Indonesia. Hubungan yang cukup relevan hanyalah *learner* dengan *university.* Di Fakultas Ilmu Komputer Universitas Indonesia relasi mahasiswa dengan suatu grup atau organisasi belum terlihat banyak berpengaruh terhadap pembelajaran kecuali sebagian kecil mahasiswa yang tergabung dengan riset grup tertentu juga terhadap mata kuliah tertentu.

4.4.Pengembangan Ontologi

Dalam pengembangan ontologi *student model*, penulis melakukan rancangan sub-sub ontologi atau pecahan-pecahan dari ontologi. Sub-sub ontologi tersebut adalah:

a. Sub-ontologi *student learning style*
b. Sub-ontologi *student performance*
c. Sub-ontologi *student data*

Berikut akan dijelaskan satu-per-satu mengenai rancangan sub-ontologi serta penyatuan ketiganya.

4.4.1. Rancangan Sub-Ontologi *Student Learning Style*

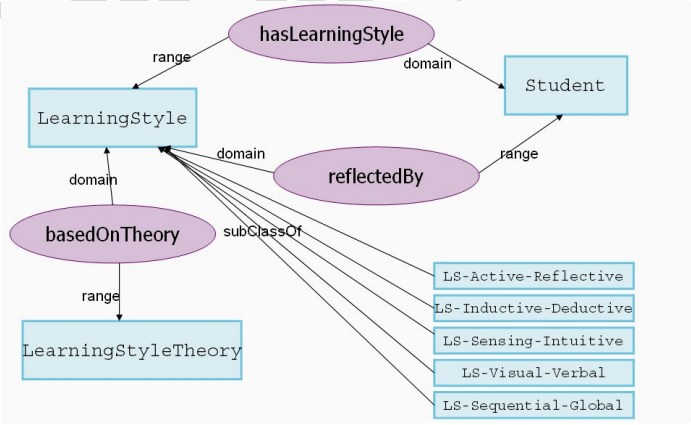
Pada butir 4.3 Analisis Ontologi telah dijelaskan bahwa *user model ontology* LOCO telah merancang *class* `Learner` memiliki `LearningStyle`. `LearningStyle` yang digambarkan juga merupakan gaya belajar yang umum dimiliki oleh semua pembelajar. Oleh karena itu, untuk mengembangkan sub-ontologi ini penulis dapat menggunakan ulang (*reuse*) rancangan yang dibuat oleh LOCO.

Namun, *relationship* yang dirancang pada *student model ontology* LOCO dirasakan perlu adanya revisi. Rancangan yang dibuat LOCO adalah setiap `Learner` memiliki `LearningStyle`, sedangkan `LearningStyle` memiliki `LearningStyleCategory`. Di mana `LS_Active-Reflective`, `LS_Inductive-Deductive`, `LS_Sensing-Intuitive`, `LS_Sequential-Global`, dan `LS_Visual-`

`Verbal` merupakan *subclass* dari `LearningStyleCategory`. Penulis menyederhanakannya menjadi `Student` memiliki `LearningStyle` di mana `LearningStyle` dapat berupa `LS_Active-Reflective`, `LS_Inductive-Deductive`, `LS_Sensing-Intuitive`, `LS_Sequential-Global`, dan `LS_Visual-Verbal`. Nantinya `LS_Active-Reflective` dapat dibuat *instances of learning style* yang bertipe *active,reflective*, atau di antaranya; `LS_Inductive-Deductive` dapat dibuat *instances of learning style* yang bertipe *inductive,deductive*, atau di antaranya; begitu juga yang lainnya dapat dibuat *instances of learning style* yang memiliki tipe sesuai dengan kategoriya. Pengkategorian ini berdasarkan teori ilmuwan tertentu sehingga *relationship* antara `LearningStyle` dan `LearningStyleTheory` pada LOCO juga diambil.

Learning style yang terdapat pada LOCO berdasarkan Felder-Silverman Learning Style Model. Penulis juga mengambil yang sama dengan LOCO karena dimensi yang dilihat pada model ini paling luas di antara model lainnya yang disebutkan pada butir 2.1.1.1 Gaya Belajar dalam Pembelajaran.

Relationship antara *class* didefinisikan dengan *object property*. Penulis juga menambahkan *object property* yang merupakan *inverse* dari `hasLearningStyle` yang dimiliki `Student`, yaitu `reflectedBy(Student)` yang dimiliki oleh `LearningStyle`. Berikut adalah tabel dan gambar rancangan sub-ontologi *student learning style.*



Gambar 4.4. Rancangan Sub-Ontologi *Student Learning Style*

Tabel 4.2. Rancangan Sub-Ontologi *Student Learning Style*

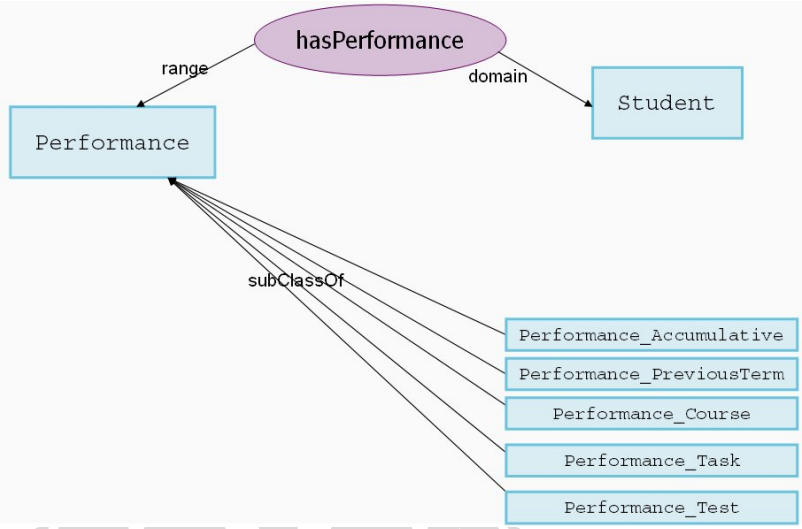
Class	***Object Property***	**Keterangan**
`Student` (Mahasiswa)	`hasLearningStyle(LearningStyle)`	Mahasiswa memiliki gaya belajar.
`LearningStyle` (Gaya Belajar) o `LS_Active-Reflective` o `LS_Inductive-Deductive` o `LS_Visual-Verbal` o `LS_Sequential-Global` o `LS_Sensing-Intuitive`	`basedOnTheory(LearningStyleTheory)`	Gaya belajar berdasarkan teori gaya belajar.
	`reflectedBy(Student)`	Gaya belajar direfleksikan oleh mahasiswa.
`LearningStyleTheory` (Teori Gaya Belajar)		

4.4.2. Rancangan Sub-Ontologi *Student Peformance*

Pada butir 4.3 (Analisis Ontologi) disebutkan bahwa rancangan *performance* berdasarkan PAPI tidak sesuai dengan kebutuhan. Penulis mengembangkan sendiri sub-ontologi *student performance* yang digambarkan dari nilai-nilai yang didapat oleh mahasiswa. Berdasarkan hal tersebut penulis mengkategorikan nilai-nilai yang didapat mahasiswa sebagai berikut:

a. Nilai akumulatif yaitu berupa IPK
b. Nilai per semester yaitu berupa IP
c. Nilai mata kuliah tertentu
d. Nilai-nilai tes
e. Nilai-nilai tugas

Selanjutnya ketegori nilai di atas menjadi *subclass* dari *class* `Performance` yang secara berurutan diberi nama : `Performance_Accumulative`, `Performance_PreviousTerm`, `Performance_Course`, `Performance_Test`, dan `Performance_Task`.



Gambar 4.5. Rancangan Sub-Ontologi *Student Performance*

4.4.3. Rancangan Sub-Ontologi *Student Data*

Data dari mahasiswa berfungsi melengkapi personalisasi serta sebagai identitas mahasiswa. Sering kali mata kuliah tertentu dapat diambil oleh mahasiswa dengan jumlah SKS tertentu atau semester tertentu. Pembelajaran di Fakultas Ilmu Komputer juga tidak hanya melibatkan mahasiswa Fakultas Ilmu Komputer serta tidak hanya mahasiswa Universitas Indonesia. Padahal dari praktek yang pernah penulis rasakan, kemampuan mahasiswa Fakultas Ilmu Komputer Universitas Indonesia berbeda dengan kemampuan mahasiswa Fakultas Ilmu Komputer universitas swasta lainnya, misalkan. Kemampuan berhubungan dengan komputer yang dimiliki mahasiswa Fakultas Ilmu Komputer juga berbeda dengan mahasiswa fakultas lain yang sama-sama di UI. Kemampuan mahasiswa S1 secara umum juga berbeda dengan mahasiswa S2.

Oleh karena itu diperlukan informasi akademik lainnya untuk ontologi. Penulis membagi informasi tersebut menjadi tiga, yaitu:

a. informasi yang menjadi atribut mahasiswa itu sendiri, seperti tahun masuk, semester, nama lengkap, *username*.

b. informasi yang sifatnya pribadi, seperti jumlah SKS, *e-mail*, nomor telepon; selanjutnya menjadi atribut atau *datatype property* dari *class* `PersonalInformation`.

c. informasi yang membutuhkan pembentukan *class* baru, seperti asal universitas, fakultas, jurusan, program studi, jenjang, dan jalur masuk. Penulis memutuskan untuk membuat *class-class* baru, karena objek-objek tersebut memiliki *relationship* satu dengan yang lain.

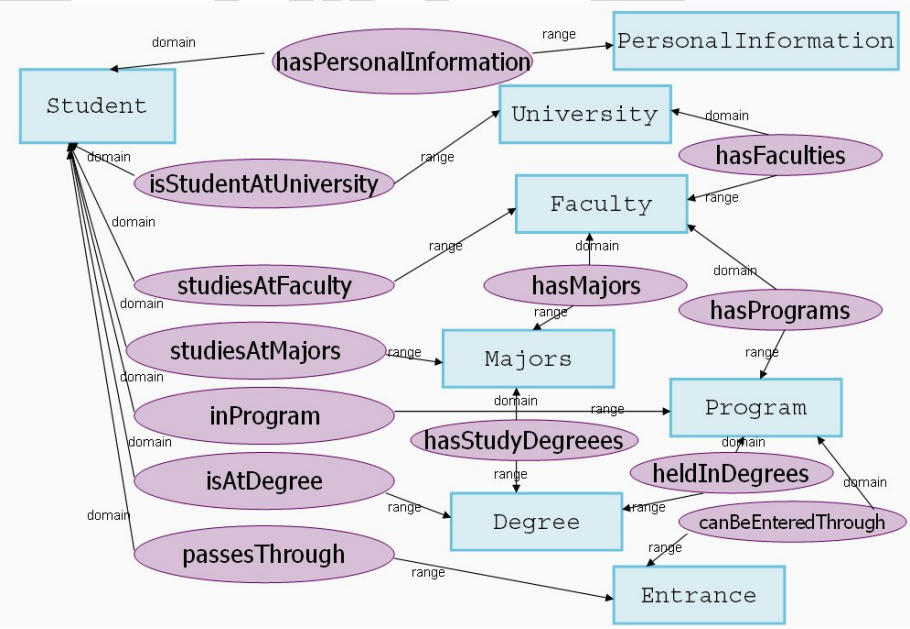
Relationship yang didapat penulis pada butir c di atas adalah:

a. Universitas memiliki beberapa fakultas, sedangkan fakultas memiliki beberapa jurusan dan atau program studi.

b. Program studi diadakan untuk jenjang tertentu dan jurusan dibuka untuk jenjang tertentu pula.

c. Mahasiswa masuk melalui jalur masuk tertentu dan program studi dapat dimasuki dengan jalur masuk tertentu.

d. Mahasiswa berasal dari universitas, fakultas, jurusan, dan program studi tertentu pada jenjang tertentu.



Gambar 4.6. Rancangan Sub-Ontologi *Student Data*

Tabel 4.3. Rancangan Sub-Ontologi *Student Data*

Class	***Object Property***	**Keterangan**
`Student` (Mahasiswa)	`isAStudentAtUniversity (University)`	Mahasiswa berasal dari universitas.
	`studiesAtFaculty(Faculty)`	Mahasiswa belajar di fakultas.
	`studiesAtMajors(Majors)`	Mahasiswa belajar di jurusan
	`inProgram(Program)`	Mahasiswa mengambil program studi.
	`isAtDegree(Degree)`	Mahasiswa berada pada jenjang.
	`passesThrough(Entrance)`	Mahasiswa melalui jalur masuk.
	`hasPersonalInformation (PersonalInformation)`	Mahasiswa memiliki informasi pribadi.
`University` (Universitas)	`hasFaculties(Faculty)`	Universitas memiliki beberapa fakultas
`Faculty` (Fakultas)	`hasMajors(Majors)`	Fakultas memiliki beberapa jurusan.
	`hasPrograms(Program)`	Fakultas memiliki beberapa program studi.
`Majors` (Jurusan)	`hasStudyDegrees(Degree)`	Jurusan memiliki jenjang.
`Program` (Program studi)	`heldInDegrees(Degree)`	Program studi diadakan pada jenjang.
	`canBeEnteredThrough(Entrance)`	Program studi dapat dimasuki melalui jalur masuk.
`PersonalInformation` (Informasi Pribadi)		

4.4.4. Penyatuan dan Pengembangan Ontologi

Setelah perancangan sub-sub ontologi, rancangan tersebut disatukan. Penyatuan tidak sulit karena setiap sub-ontologi memiliki *class* `Student` dan *class-class* lainnya pada sub-ontologi yang berbeda tidak saling memiliki *relationship.* Contohnya `LearningStyle`, `Performance`, dan `University`; satu dengan yang lainnya tidak memiliki hubungan. Oleh karena itu, *student model ontology* memiliki sebelas *class* yang menjadi *subclass* dari `owl:Thing`. Sebelas *class* tersebut adalah:

1. `Student` (Mahasiswa)
2. `LearningStyle` (Gaya Belajar)

3. `LearningStyleTheory` (Teori Gaya Belajar)
4. `Performance` (Performa)
5. `PersonalInformation` (Informasi Pribadi)
6. `University` (Universitas)
7. `Faculty` (Fakultas)
8. `Majors` (Jurusan)
9. `Program` (Program studi)
10. `Degree` (Jenjang)
11. `Entrance` (Jalur masuk)

Relationship yang terdapat pada *student model ontology* adalah seperti yang didefinisikan pada sub-sub ontologi. Selanjutnya ditentukan *datatype properti*, *restriction*, dan pengembangan dengan OWL.

4.4.4.1. Penentuan *Datatype Property*

Datatype property adalah properti yang dimiliki oleh *class* bertipe literal (bukan *instance of class).* Pada pengembangan awal, *datatype property* baru didefinisikan sebagian. *Datatype property* yang dimiliki oleh suatu *class* disesuaikan dengan kebutuhan. Namun ternyata penentuan *datatype property* bukan merupakan proses yang mudah. Penulis sering kali mencoba membuat *instances* lalu memikirkan *datatype property* apa saja yang dibutuhkan. Pengembangan portal juga sangat membantu penentuan *datatype property* karena pada portal penulis berusaha menampilkan *instances.*

Contoh berikut adalah penentuan *datatype property* dengan pembentukan *instances* terlebih dahulu.

a. Penulis membuat *instances* dari *performance_test,* misalnya Nilai UTS. Maka, *datatype property* yang dibutuhkan adalah `title` yang akan diisi dengan string:"UTS".
b. Untuk dapat mengetahui nilai dari "UTS" diperlukan *datatype property* `score` berupa *float*, namun dapat juga `value` yang berupa string (A, A+, B, B+, C, C+, D, D+, E).

c. *Range* nilai dapat bermacam-macam, sehingga untuk dapat mengetahui *range* nilai dibuat *datatype property* `lowestScore`, `lowestValue`, `highestScore`, dan `highestValue`.

d. Ternyata kebutuhan pada butir b dan c juga dibutuhkan semua *subclass* dari *performance* lainnya sehingga *datatype property* ini didefinisikan pada *class* `Performance`. Sedangkan `title` tidak dapat dimiliki semua *subclass* dari `Performance` karena `Performance_Accumulative` (IPK) dan `Performance_PreviousTerm` (IP) tidak memiliki judul yang berubah-ubah seperti `Performance_Test` (tes) dan `Performance_Task` (tugas). Namun, *datatype property* `title` dapat dihapus jika kedua *class* tersebut masing-masing dihubungkan dengan *class* `Test` dan `Task`. Tetapi karena kedua *class* tersebut berada pada area ontologi lain, kedua *class* tersebut tidak didefinisikan pada *student model ontology.*

e. Suatu nilai relatif terhadap nilai lain. Nilai 60 dapat dikatakan bagus, sedangkan nilai 80 dapat dikatakan biasa saja. Maka diperlukan *datatype property* `lowestScoreOfStudents` dan `highestScoreOfStudents`.

Kemudian, didapat *datatype property* setiap *class* adalah sebagai berikut:

1. `Student: name, enterYear, term`
2. `LearningStyle`
   a. `LS_Active-Reflective: typeOfActiveReflective`
   b. `LS_Sensing-Intuitive: typeOfSensingIntuitive`
   c. `LS_Visual-Verbal: typeOfVisualVerbal`
   d. `LS_Sequential-Global: typeOfSequentialGlobal`
   e. `LS_Inductive-Deductive: typeOfInductiveDeductive`
3. `LearningStyleTheory: title`
4. `Performance: score, value, lowestScore, highestScore, lowestValue, highestValue.`
   a. `Performance_Accumulative`
   b. `Performance_PreviousTerm: SKSTerm`
   c. `Performance_Course: course`
   d. `Performance_Test: title, highestScoreOfStudents, lowestScoreOfStudents.`
   e. `Performance_Task:title, highestScoreOfStudents, lowestScoreOfStudents.`

```
5. PersonalInformation:email, phone, sksTotal.
6. University: nameOfUniv
7. Faculty: nameOfFaculty
8. Majors : nameOfMajors
9. Program: nameOfProgram
10.   Degree: nameOfDegree
11.   Entrance: nameOfEntrance
```

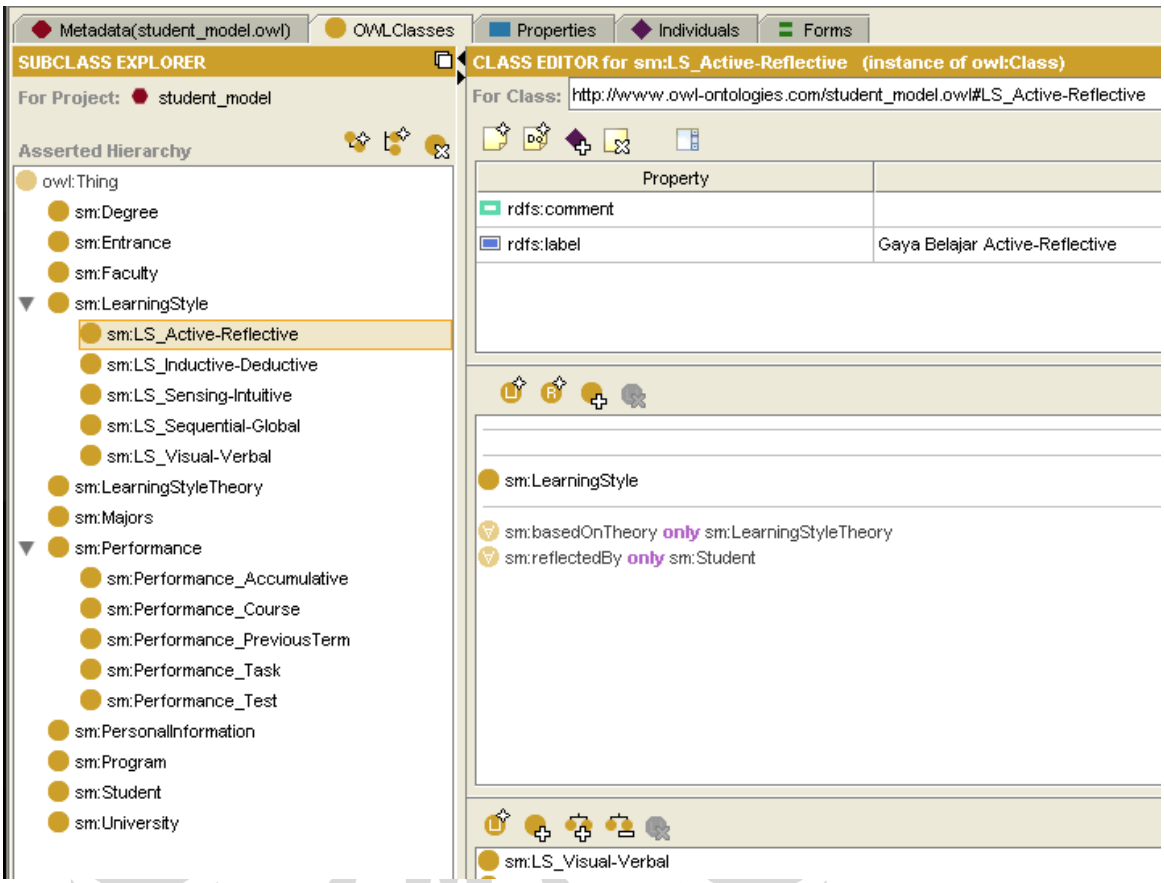
4.4.4.2. Penentuan *Restriction*

Restriction yang dibuat pada ontologi ini adalah *restriction properties*, di mana semua *object properties* yang dimiliki *class* memiliki *value* yang hanya berasal dari suatu *class* tertentu. Contohnya adalah `hasPerformance` pada `Student`, memiliki *values* yang berasal hanya dari *class* `Performance`, sehingga *restriction*-nya menggunakan *quantifier universal* (`allValuesFrom`).

4.4.4.3. Pengembangan dengan OWL

Pengembangan dengan OWL awalnya direncanakan menggunakan *tool* Protégé. Ini dilakukan untuk mempermudah pembuatan *source code* karena *source code* di-*generate* oleh Protégé. Namun, setelah mencobanya ternyata Protégé tidak hanya menghasilkan berkas .owl tetapi juga berkas .pprj yang tidak dibutuhkan dalam penelitan ini. Selain itu, dalam berkas .owl tidak didefinisikan bahwa *class*-*class* yang menjadi *class* utama adalah *subclass* dari `owl:Thing`. Alasan terakhir adalah *source code* yang dihasilkan oleh Protégé tidak rapi, padahal kerapihan penulisan sangat penting untuk kemudahan peng-*edit*-an serta untuk dokumentasi.

Protégé tetap digunakan dalam pengembangan untuk melihat gambaran *source code* yang dihasilkan. Seperti pada Gambar 4.7; bagian kiri untuk melihat hirarki *class* dan bagian kanan untuk melihat deskripsi dari *class* yang ditunjuk.



Gambar 4.7. Tampilan *Class* pada Protégé

Langkah pertama yang dilakukan dalam pembuatan *source code* diperlihatkan pada gambar 4.8. Sintaks yang digunakan adalah format RDF/XML atau *serialization. Prefix* dan *namespace* didefinisikan pada langkah pertama ini. Ontologi yang dikembangkan baru menggunakan *prefix* sm yang didefinisikan penulis untuk *namespace* URI: `http://www.owl-ontologies.com/student_model.owl` `serta` *prefix* rdf, rdfs, dan owl.

```
<?xml version="1.0"?>
<rdf:RDF
      xmlns:xsp="http://www.owl-
      ontologies.com/2005/08/07/xsp.owl#"
      xmlns:sm="http://www.owl-
      ontologies.com/student_model.owl#"
      xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
      xmlns:swrl="http://www.w3.org/2003/11/swrl#"
      xmlns:protege="http://protege.stanford.edu/plugins/owl/pro
      tege#"
      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#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xml:base="http://www.owl-
      ontologies.com/student_model.owl">

<owl:Ontology rdf:about="">
      <owl:versionInfo
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1.0
      </owl:versionInfo>
</owl:Ontology>

<!—Pendefinisian class, subclass, dan property ditulis di sini--
>

</rdf:RDF>
```

Gambar 4.8. Penulisan Awal *Source Code*

Selanjutnya setiap *class* dari *subclass* `owl:Thing` dibuat. *Class* dinyatakan dengan `<owl:Class></owl:Class>`, sedangkan *subclassOf* sesuatu dinyatakan dengan `<rdfs:subcClassOf></rdfs:subClassOf>`. Label dari *class* dinyatakan dengan `<rdfs:label></rdfs:label>`. Gambar 4.9. adalah contoh *source code* untuk membuat *class* `Entrance` sebagai *subclass* dari `owl:Thing` dengan label bertipe string “Jalur Masuk”.

```
<owl:Class rdf:about="http://www.owl-
ontologies.com/student_model.owl#Entrance">
      <rdfs:subClassOf
      rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
      <rdfs:label
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Jalur Masuk</rdfs:label>
</owl:Class>
```

Gambar 4.9. Contoh Penulisan SubClassOf dan Label

Kemudian dibuat *subclass* dari `LearningStyle` dan `Performance`. Setiap *subclass* dari `LearningStyle` dan `Performance` adalah *subclass* terbawah, maksudnya tidak memiliki *subclass* lagi. Jika diperlukan, didefinisikan *disjoint* antara *subclass* yang satu dengan yang lainnya yang menunjukkan suatu *instances* tidak dapat masuk ke dalam *class* yang dinyatakan *disjoint. Disjoint* harus dinyatakan satu per satu dengan `<owl:disjointWith></owl:disjointWith>`. Gambar 4.10. adalah contoh *source code* yang menunjukkan *class* `LS_Active-Reflective` adalah *subclass* dari `LearningStyle` dan *disjoint* dengan `LS_Visual-Verbal` serta diberi label "Gaya Belajar Active-Reflective".

```
<owl:Class rdf:about="http://www.owl-
ontologies.com/student_model.owl#LS_Active-Reflective">
      <rdfs:subClassOf>
            <owl:Class rdf:about="http://www.owl-
            ontologies.com/student_model.owl#LearningStyle
            "/>
      </rdfs:subClassOf>
      <owl:disjointWith>
            <owl:Class rdf:about="http://www.owl-
            ontologies.com/student_model.owl#LS_Visual-
            Verbal"/>
      </owl:disjointWith>
      <rdfs:label
      rdf:datatype="http://www.w3.org/2001/XMLSchema#strin
      g">Gaya Belajar Active-Reflective</rdfs:label>
</owl:Class>
```

Gambar 4.10. Contoh Penulisan SubClassOf dan DisjointWith

Properti terdiri dari *datatype property* dan *object property.* Jika hanya satu *value* yang bisa dimiliki oleh suatu *instance of class* maka disebut *functional property.*

Gambar 4.11. adalah *source code* untuk *functional property* `name` yang dimiliki `Student` dengan tipe *datatype property* berupa string.

```
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/student_model.owl#name">
        <rdfs:domain rdf:resource="http://www.owl-
      ontologies.com/student_model.owl#Student"/>
        <rdf:type
      rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
        <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
```

Gambar 4.11. *Code* untuk *Functional Datatype Property* 'name'

```
<owl:ObjectProperty rdf:about="http://www.owl-
ontologies.com/student_model.owl#hasPerformance">
        <rdfs:label
        rdf:datatype="http://www.w3.org/2001/XMLSchema#strin
        g">has Performance</rdfs:label>
        <rdf:type
        rdf:resource="http://www.w3.org/1999/02/22-rdf-
        syntax-ns#Property"/>
</owl:ObjectProperty>
```

Gambar 4.12. *Code* untuk *Object Property* 'hasPerformance'

`<owl:ObjectProperty></owl:ObjectProperty>` adalah untuk mendefinisikan *object property* seperti pada Gambar 4.12. Sedangkan `<owl:Restriction></owl:Restriction>` digunakan untuk mendefinisikan *restriction.* Pada gambar 4.13. adalah contoh penulisan *restriction property* `hasPerformance` pada *class* `Student`.

Sering kali suatu *datatype property* hanya mengijinkan *value* tertentu. Pada ontologi *student model* semua *datatype property 'type of subbclass'* dari `LearningStyle` dapat memiliki *value* sesuai dengan kategorinya. Contoh pada Gambar 4.14., `typeOfVisualVerbal` dapat memiliki *value*: "visual", "verbal", "cenderung visual", "cenderung verbal", atau "visual-verbal seimbang".

```
<owl:Class rdf:about="http://www.owl-
ontologies.com/student_model.owl#Student">
      <rdfs:subClassOf><owl:Restriction>
        <owl:onProperty>
            <owl:ObjectProperty rdf:about="http://www.owl-
      ontologies.com/student_model.owl#hasPerformance"/>
       </owl:onProperty>
       <owl:allValuesFrom rdf:resource="http://www.owl-
ontologies.com/student_model.owl#Performance"/>
      </owl:Restriction></rdfs:subClassOf>
</owl:Class>
```

Gambar 4.13. *Code* untuk *Restriction Property* hasPerformance allvaluesfrom Performance

```
  <owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/student_model.owl#typeOfVisualVerbal">
    <rdfs:domain rdf:resource="http://www.owl-
ontologies.com/student_model.owl#LS_Visual-Verbal"/>
    <rdfs:range><owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">visual</
rdf:first>
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">verbal</
rdf:first>
                    <rdf:rest rdf:parseType="Resource">
                      <rdf:rest rdf:parseType="Resource">
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">visual-
verbal seimbang</rdf:first>
                        <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                      </rdf:rest>
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">cenderun
g visual</rdf:first>
                    </rdf:rest>
                  </rdf:rest>
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">cenderun
g verbal</rdf:first>
                </rdf:rest>
        </owl:oneOf>
      </owl:DataRange></rdfs:range>
  </owl:FunctionalProperty>
```

Gambar 4.14. Contoh Penulisan *Enumerated Datatype*

Penentuan *range of data values* dinamakan *enumerated datatypes.* Format *datatype* ini menggunakan `owl:oneOf` yang juga digunakan untuk menggambarkan *enumerated class.* Dalam kasus *enumerated datatype*, subjek dari `owl:oneOf` adalah sebuah *node* kosong dari *class* `owl:DataRange` dan objeknya adalah *list of literals.* Kita harus menetapkan *list of data values* dengan *list constructs* dasar `rdf:first`, `rdf:rest` dan `rdf:nil`.

4.5.Persiapan Data

Pada tahap ini dilakukan pembuatan *instances* dan pendefinisian *rules* berdasarkan ontologi *student model.*

4.5.1. Pembuatan *Instances*

Karena *semantic portal* menggunakan data dalam format RDF khususnya sintaks N3, pembuatan data *instances* menggunakan format N3. Data yang digunakan bukan data asli, karena data asli sifatnya rahasia, sehingga penulis membuat data sendiri. Setiap *class* utama dipisah berkas .N3, sehingga dihasilkan 11 berkas .N3 untuk *instances.*

```
sm:LS_1
      a       sm:LS_Active-Reflective;
      sm:basedOnTheory sm:LST_1;
      sm:description "Pembelajar dengan gaya belajar aktif
cenderung untuk memelihara dan memahami informasi yang terbaik
dengan melakukan keaktifan dengannya, dapat dengan membahas,
menerapkannya, atau menjelaskannya kepada orang lain. Sebuah
ungkapan dari pembelajar aktif adalah, Mari mencobanya dan
lihat bagaimana ia bekerja. Pembelajar aktif lebih menyukai
belajar kelompok, mereka sangat sulit mendengarkan ceramah
tanpa melakukan kegiatan fisik dengan hanya menulis catatan.";
      sm:typeOfActiveReflective "active".

sm:LS_2
      a       sm:LS_Active-Reflective;
      sm:typeOfActiveReflective "reflective".
```

Gambar 4.15. Contoh *Code* untuk *Instances of* LearningStyle

Contoh penulisan *code* untuk *instances* dalam format .N3 terdapat pada Gambar 4.15. sm adalah *prefix* untuk *namespace* URI: `http://www.owl-ontologies.com/student_model.owl`. `LS_1` adalah *id* dari *instance.* a menunjukkan `rdf :type.` `basedOnTheory,` `description,` dan `typeOfActiveReflective` adalah properti yang dimiliki oleh *class* `LS_Active-Reflective,` diberikan *value* di sebelah kanan berupa literal (diapit tanda kutip) atau *instance* (contoh : `sm:basedOnTheory` **`sm:LST_1`**;).

Dalam tugas akhir pengembangan *semantic portal* berbasis ontologi komunitas riset, disebutkan untuk data yang bersumber dari *file spreadsheet* (.xls) terdapat tiga alternatif proses untuk menghasilkan data RDF, yaitu mengetik secara manual seperti yang telah dijelaskan sebelumnya, menggunakan *tool* Protégé, atau menggunakan *converter tool.* Namun karena tidak ada sumber berkas .xls, cara termudah yang dilakukan adalah pengetikan langsung secara manual. Adapun jika berkas .xls tersedia, cara termudah adalah menggunakan *converter tool.*

4.5.2. Pendefinisian *Rules*

Untuk mendapatkan data baru dari data yang sudah ada, dibuat *rules* untuk melakukan proses *inference.* Dalam tugas akhir pengembangan *semantic portal* berbasis ontologi komunitas riset, disebutkan proses *inference* pada portalCore menggunakan GenericRuleReasoner Jena sehingga struktur dan sintaks *rules* mengikuti *rule engine* tersebut.

Berikut ini adalah RDFS *closure rules* yang didefinisikan oleh portalCore.

a. `[rdfs2: (?x ?p ?y), (?p rdfs:domain ?c) -> (?x rdf:type ?c)]`

   menyatakan x adalah subjek, p predikat, y objek ; apabila *property* p memiliki *domain* dari *class* c, maka x bertipe c.

b. `[rdfs3: (?x ?p ?y), (?p rdfs:range ?c) -> (?y rdf:type ?c)]`

   menyatakan x adalah subjek, p predikat, y objek ; apabila *property* p memiliki *range* dari *class* c, maka y bertipe c.

c. `[rdfs5a: (?a rdfs:subPropertyOf ?b), (?b rdfs:subPropertyOf ?c) -> (?a rdfs:subPropertyOf ?c)]`
   menyatakan jika a *subproperty* dari b dan b *subproperty* dari c, maka a *subproperty* dari c.
d. `[rdfs5b: (?a rdf:type rdf:Property) -> (?a rdfs:subPropertyOf ?a)]`
   menyatakan suatu *property* adalah *sub property* dari dirinya sendiri.
e. `[rdfs6: (?a ?p ?b), (?p rdfs:subPropertyOf ?q) -> (?a ?q ?b)]`
   menyatakan a adalah subjek, p predikat, b objek ; apabila *property* p *subproperty* dari q, maka a memiliki *property* q dengan *value* b.
f. `[rdfs7: (?a rdf:type rdfs:Class) -> (?a rdfs:subClassOf ?a)]`
   menyatakan suatu *class* adalah *subclass* dari dirinya sendiri.
g. `[rdfs8: (?a rdfs:subClassOf ?b), (?b rdfs:subClassOf ?c) -> (?a rdfs:subClassOf ?c)]`
   menyatakan jika a *subclass* dari b dan b *sbuclass* dari c, maka a *subclass* dari c.
h. `[rdfs9: (?x rdfs:subClassOf ?y), (?a rdf:type ?x) -> (?a rdf:type ?y)]`
   menyatakan jika x *subclass* dari y dan a *instance* dari x, maka a *instance* dari y juga.

Selanjutnya didefinisikan beberapa *rules* oleh penulis.

a. `(?A rdf:type sm:Student),(?A sm:hasLearningStyle ?B) -> (?B sm:reflectedBy ?A) .`
   menyatakan jika seorang mahasiswa A memiliki *learning style* B, maka *learning style* B direfleksikan oleh mahasiswa A.
b. `(?A sm:isAStudentAtUniversity ?B), (?A sm:studiesAtFaculty ?C) -> (?B sm:hasFaculties ?C) .`
   menyatakan jika A kuliah di Universitas B dan juga fakultas C, maka B memiliki fakultas C.
c. `(?A sm:studiesAtFaculty ?B), (?A sm:studiesAtMajors ?C) -> (?B sm:hasMajors ?C) .`
   menyatakan jika A kuliah di fakultas B dan juga jurusan C, maka fakultas B memiliki jurusan C.

d. `(?A sm:studiesAtFaculty ?B), (?A sm:inProgram ?C) -> (?B sm:hasPrograms ?C) .`

   menyatakan jika A kuliah di fakultas B dan juga program studi C, maka fakultas B memiliki program studi C.

e. `(?A sm:studiesAtMajors ?B), (?A sm:isAtDegree ?C) -> (?B sm:hasStudyDegrees ?C) .`

   menyatakan jika A kuliah di jurusan B dan ia berada pada jenjang C, maka jurusan B memiliki jenjang C.

f. `(?A sm:inProgram ?B), (?A sm:isAtDegree ?C) -> (?B sm:heldInDegrees ?C) .`

   menyatakan jika A mengambil program studi B dan ia berada pada jenjang C, maka program studi B diadakan pada jenjang C.

g. `(?A sm:inProgram ?B), (?A sm:passesThrough ?C) -> (?B sm:canBeEnteredThrough ?C) .`

   menyatakan jika A mengambil program studi B dan ia melalui jalur masuk C maka program studi B dapat dimasuki melalui jalur masuk C.

Ada juga beberapa *rules* yang didefinisikan untuk visualisasi, ditunjukkan pada Gambar 4.16. di mana *rules* tersebut untuk mendefinisikan label dari setiap *instances of class.*

```
(?A rdf:type sm:Student), (?A sm:name ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:University), (?A sm:nameOfUniv ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:Faculty), (?A sm:nameOfFaculty ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:Majors), (?A sm:nameOfMajors ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:Program), (?A sm:nameOfProgram ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:Degree), (?A sm:nameOfDegree ?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:Entrance), (?A sm:nameOfEntrance ?B) -> (?A rdfs:label ?B)
.
(?A rdf:type sm:PersonalInformation)->(?A rdfs:label 'klik untuk lihat').
(?A rdf:type sm:LS_Active-Reflective), (?A sm:typeOfActiveReflective ?B) -
> (?A rdfs:label ?B) .
(?A rdf:type sm:LS_Sensing-Intuitive), (?A sm:typeOfSensingIntuitive ?B) -
> (?A rdfs:label ?B) .
(?A rdf:type sm:LS_Inductive-Deductive), (?A sm:typeOfInductiveDeductive
?B) -> (?A rdfs:label ?B) .
(?A rdf:type sm:LS_Sequential-Global), (?A sm:typeOfSequentialGlobal ?B) -
> (?A rdfs:label ?B) .
(?A rdf:type sm:LS_Visual-Verbal), (?A sm:typeOfVisualVerbal ?B) -> (?A
rdfs:label ?B) .
(?A rdf:type sm:LearningStyleTheory), (?A sm:title ?B) -> (?A rdfs:label
?B) .
(?A rdf:type sm:Performance), (?A sm:title ?B) -> (?A rdfs:label ?B) .
```

Gambar 4.16. Pendefinisian *Rules* untuk Visualisasi Portal

4.6.Pengembangan Portal

Pengembangan portal ditujukan supaya dapat melihat pemodelan apakah sudah sesuai dengan pendefinisian kebutuhan, dengan pembuatan contoh *instances* yang disesuaikan dengan kenyataan (walaupun bukan merupakan data asli), juga dengan pendefinisan *rules.* Karena bagian ini hanya sebagai pelengkap, pembahasan juga dijelaskan dengan singkat.

Pengembangan portal mengikuti tahapan yang dilakukan dalam pengembangan *semantic* portal berbasis komunitas riset. Data yang perlu disiapkan adalah ontologi, *instances,* dan *rules.* Semua data disimpan pada direktori portal[7]/web/data. Setelah data disiapkan, tahap selanjutnya untuk pengembangan portal adalah konfigurasi portal dan tampilan portal.

Konfigurasi dilakukan pada berkas sources.n3 yang berada pada direktori portal/web/WEB-INF/config. Pada tahap ini dilakukan pendefinisian *datasource, facets,* dan *templates.*

```
[]     rdf:type pcv:DataSource ;
       rdfs:label "Mahasiswa" ;
       pcv:encoding "p1";
       pcv:order "10"^^xsd:integer ;
       dc:description "Prototype Student Model Ontology" ;

       pcv:sourceURL <portal://data/student.n3> ;
       pcv:sourceURL <portal://data/LS.n3> ;
       pcv:sourceURL <portal://data/LST.n3> ;
       ...

       pcv:ontologySourceURL <portal://data/student_model.owl> ;
       pcv:closureRulesURL   <portal://data/portal.rules> ;
       pcv:filterOnType sm:Student;
       pcv:styleSheet "site.css" ;
```

Gambar 4.17. Pendefinisian *Datasource*

Gambar 4.17 adalah contoh pendefinisian *datasource* ”Mahasiswa” dengan membuat *instance* `pcv:DataSource`. Ada beberapa properti yang mengatur *input* portal:

a. `pcv:ontologySourceURL` : *input* ontologi

b. `pcv:closureRulesURL` : *input rules*

[7]http://www.swed.org.uk/swed/portal.zip

c. `pcv:sourceURL` : *input instances*

Pendefinisian *facet* adalah melalui properti `pcv:facet`. `pcv:linkProp` mengatur properti mana yang akan *filter*. Contohnya pada Gambar 4.18 adalah *facet* dengan `pcv:linkProp sm:inProgram` yang berarti program studi yang diambil mahasiswa. Tipe *facet* ada 3, yaitu *flat, alpharange,* dan *hierarchical.*

```
pcv:facet sm:p1nameFacet ;
pcv:facet sm:p1LSFacet ;
pcv:facet sm:p1yearFacet ;
...
 sm:p1nameFacet a pcv:AlphaRangeFacet;
   rdfs:label "Nama" ;
   pcv:linkProp sm:name;
   pcv:order "1"^^xsd:integer;
   .
```

Gambar 4.18. Pendefinisian *facet*

Gambar 4.19. *Facet* Program Studi untuk Mahasiswa

Pendefinisian *template* menggunakan `pcv:template`. Template yang didefinisikan adalah yang dihubungkan dengan *datasource. Template* diatur pada direktori portal/web/WEB-INF/templates. Bahasa yang digunakan adalah HTML, PHP, dan JSP. Contoh *source code template* terdapat pada Gambar 4.20.

```
pcv:template [a pcv:Template;
        pcv:templateContext "page" ;
        pcv:templatePath
   <portal://templates/pageDefault.vm> ; ];
pcv:template [a pcv:Template;
        pcv:templateContext "default" ;
        pcv:templatePath      <portal://templates/leaf.vm> ;];
pcv:template [a pcv:Template;
        pcv:templateContext "page" ;
        pcv:templatePath
<portal://templates/pageStudent.vm> ;
   pcv:templateClass     sm:Student;];
```

Gambar 4.20. Contoh Pendefinisian *Template*

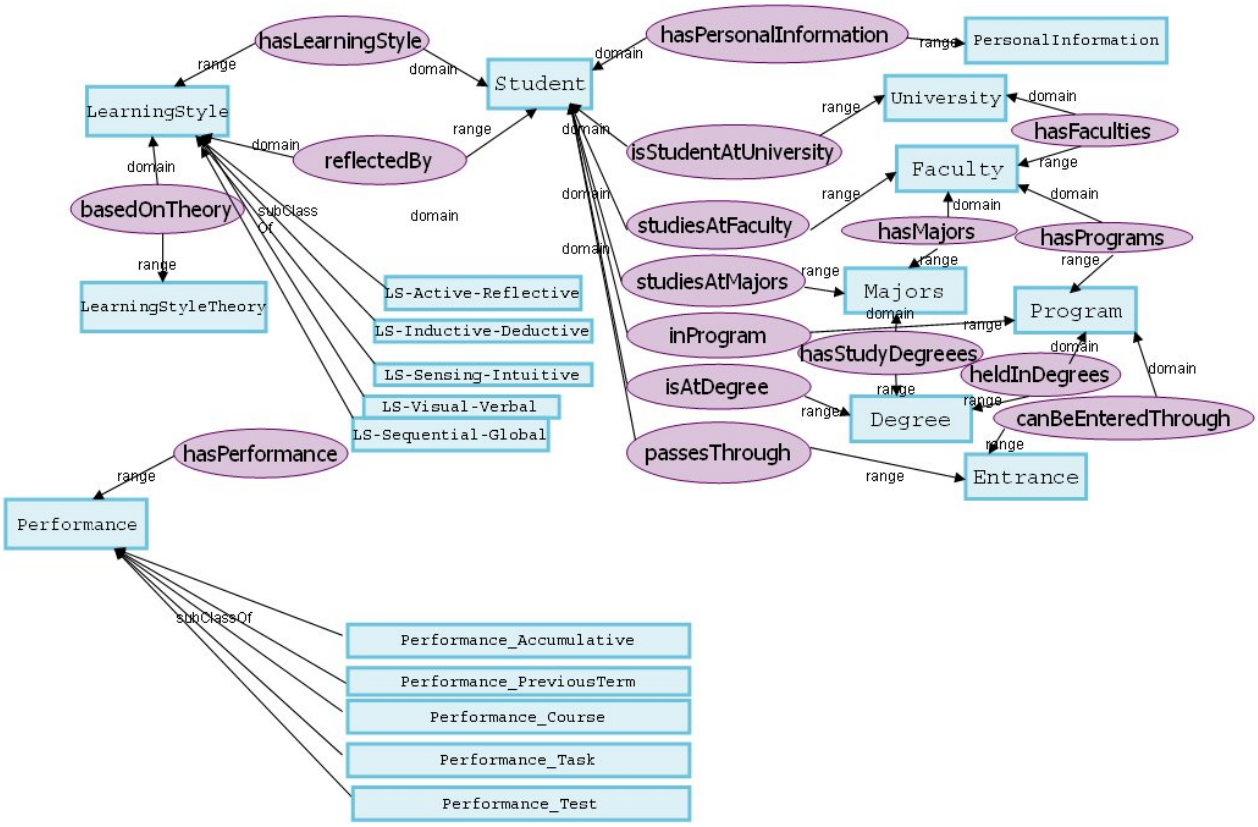
```
## Velocity template for render objects of type Student in page
format

<p><table width="100%">
<tr >
            #set ($link = $resource.getPropertyValue("sm:photo"))
            <td width="15%" align="center" >
             <img border="1" height="150"
src="${request.contextPath}/images/photo/$link">
            </td>

       <td width="85%">
              <h1>$!resource.getProperty("sm:name").value.name</h1>

                     #if($resource.hasType("sm:Student"))
                            Mahasiswa
                     #end

              <br><br>  <b>Tahun Masuk:</b>
$resource.getPropertyValue("sm:enterYear")
              <br>
<b>Semester:</b>$resource.getPropertyValue("sm:term")
              <br>
                        #if
($resource.hasProperty("sm:hasPersonalInformation"))
            <b>Personal Information</b>
               #set($count=0)
               #foreach ($p in
$resource.findProperties("sm:hasPersonalInformation"))
                     #foreach ($v in $p.values)
                            #if ($count > 0 ) <br>
                            #end
                            $v.render("leaf", $request)
                     #set($count = $count+1)
                     #end
              #end
```

Gambar 4.21. *Source Code Template* untuk Halaman Mahasiswa (PageStudent.vm)

BAB 5
HASIL DAN PEMBAHASAN

5.1. Hasil

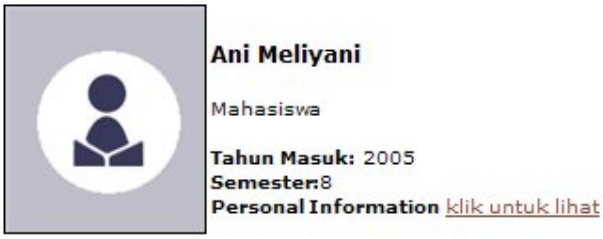
Pengembangan ontologi *student model* menghasilkan model yang digambarkan pada Gambar 5.1. Gambaran lebih jelas dapat dilihat pada Lampiran 1, sedangkan OWL *source code* dapat dilihat pada Lampiran 2.



Gambar 5.1. *Student Model Ontology*

Contoh visualisasi *instances of class* melalui *semantic portal* terdapat pada Gambar 5.2. Objek Ani Meliyani adalah *instance* dari `Student`. “Ani Meliyani” adalah *value* dari `name`. “2005” adalah *value* dari `enterYear`, sedangkan “8” adalah *value* dari `term`. Data yang memiliki *link* menandakan data tersebut adalah

instance of class. Tampilan pada *semantic portal* lainnya dapat dilihat pada lampiran.



Ani Meliyani

Mahasiswa

Tahun Masuk: 2005
Semester:8
Personal Information klik untuk lihat

Info Akademis

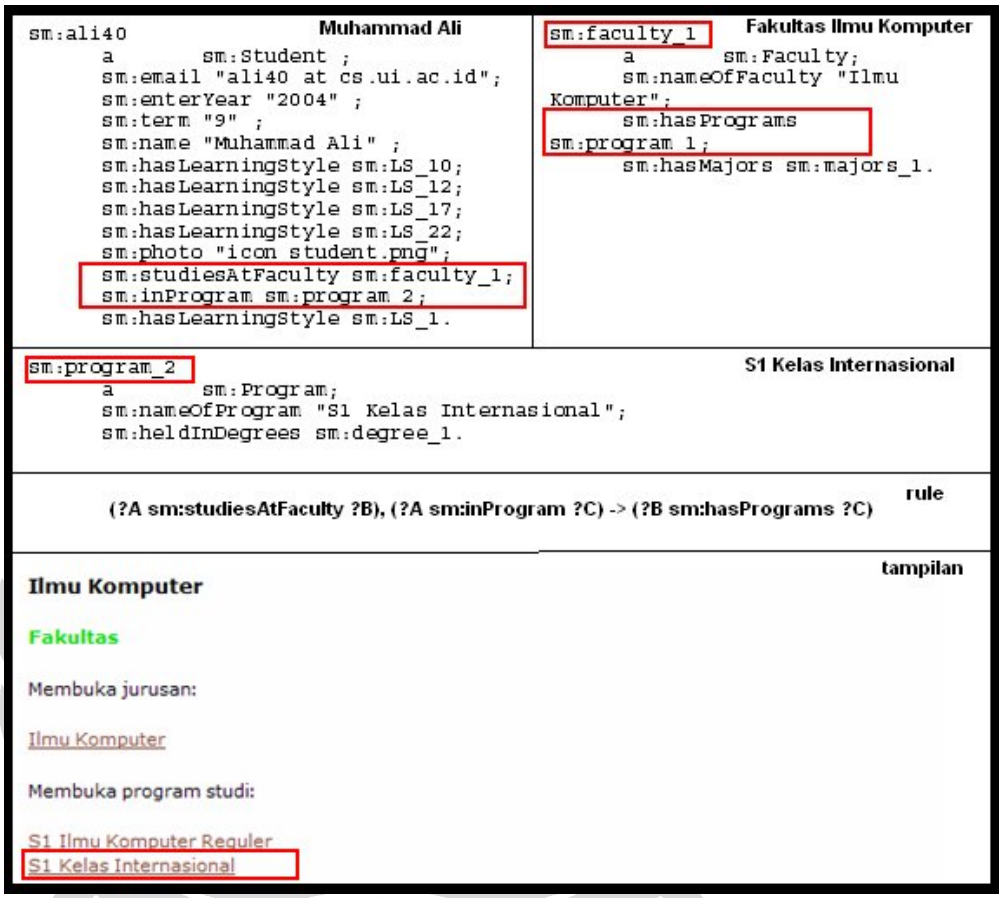
Universitas	Universitas Indonesia
Fakultas	Ilmu Komputer
Jurusan	Ilmu Komputer
Program Studi	S1 Ilmu Komputer Reguler
Jenjang	Sarjana
Jalur Masuk	SPMB
Gaya Belajar	active cenderung sensing cenderung verbal inductive sequential
Performa	Indeks Prestasi Kumulatif Quiz 1 Tugas 1

Gambar 5.2. Tampilan *Instance of Student* pada *Semantic Portal*

5.2.Pembahasan

Pada *semantic portal*, selain telah menampilkan *instance* sesuai dengan data yang dimasukkan dan ontologi yang dirancang, *rules* yang didefinisikan juga berhasil berjalan. Contohnya pada *instance* fakultas “Ilmu Komputer” tidak didefinisikan memiliki program studi ”S1 Kelas Internasional”. Namun pada data Student didefinisikan ”Muhammad Ali” belajar di fakultas “Ilmu Komputer” dan ”Muhammad Ali” berada pada program ”S1 Kelas Internasional”. Karena pada

portal dibuat *rule* `(?A sm:studiesAtFaculty ?B), (?A sm:inProgram ?C) -> (?B sm:hasPrograms ?C)` maka terdapat data baru bahwa “Ilmu Komputer” memiliki program studi ”S1 Kelas Internasional”.



Gambar 5.3. Contoh Data yang Didapat dari *Inference Rule*

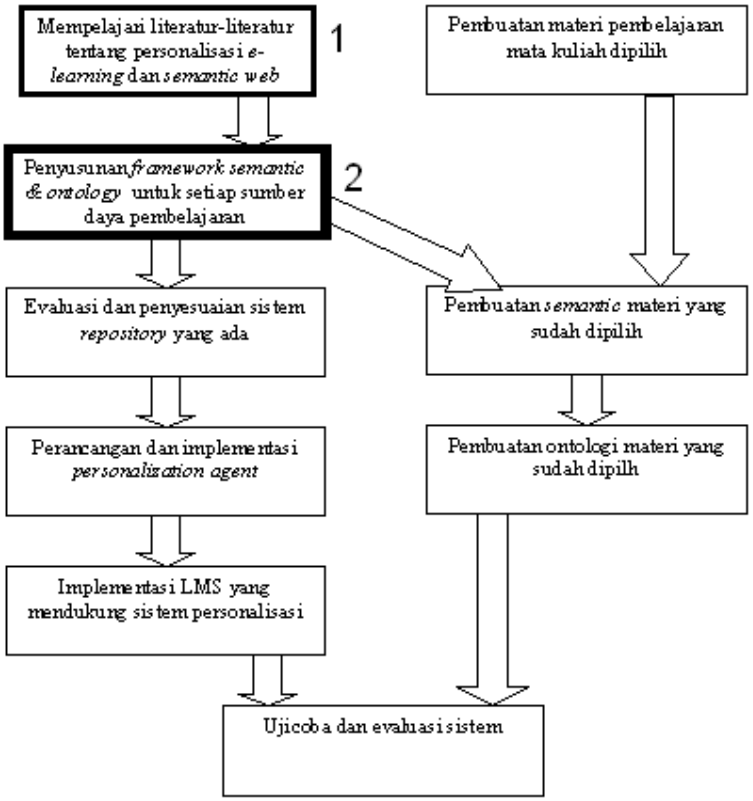
Tampilan pada *semantic portal* yang sudah menampilkan data sesuai dengan *instance*, ontologi, dan *rule* memperlihatkan bahwa ontologi dan *rules* sudah merepresentasikan pemodelan data yang dibutuhkan untuk personalisasi sesuai dengan pendefinisian kebutuhan. Namun, untuk menyimpulkan bahwa ontologi ini cocok dengan personalisasi pembelajaran online khususnya di Fasilkom UI masih terlalu dini, perlu dilakukan pengembangan sistem dan pengujian sistem.

Selain kesimpulan di atas, ada beberapa evaluasi penulis setelah melihat hasil pengembangan, di antaranya sebagai berikut:

a. Gaya belajar yang digunakan pada *student model ontology* terlihat sudah dapat merepresentasikan *style* dalam cakupan dimensi yang berbeda-beda. Namun, gaya belajar dapat saja ditambah dari berbagai macam model karena ontologi yang dibuat dapat merepresentasikan hal tersebut. Dapat juga dilakukan penelitian maupun studi literatur lebih lanjut mengenai model yang paling cocok, di mana belum dilakukan pada penelitian ini.
b. Pada penelitian ini, penulis menggunakan komponen-komponen ontologi yang sederhana. Kemungkinan dengan perkembangan *semantic web modeling* maupun pembelajaran lebih mendalam, ontologi dapat disempurnakan. Salah satunya pada pengembangan ontologi *student model* ini penulis tidak berhasil membuat *range* untuk data bertipe *float* atau *int*[8]. Diharapkan masalah tersebut dapat terpecahkan seiring dengan penyempurnaan OWL 2.0 beserta pendalaman penguasaan *semantic web modeling language.*
c. *Student model ontology* dapat diperluas dengan mempertimbangkan preferensi, dan lain-lain. Dalam penelitian ini ontologi hanya mempertimbangkan kebutuhan yang telah didefinisikan. Namun jauh ke depan, apabila diimplementasikan ontologi dapat meluas atau berkurang lingkupnya jika disesuaikan dengan kebutuhan sebenarnya. Hal tersebut tentunya membutuhkan pengujian.
d. *Student model ontology* butuh diintegrasikan dengan ontologi lainnya terutama berkaitan dengan aspek pembelajaran untuk mengembangkan personalisasi pembelajaran online. Integrasi dengan ontologi lainnya sangat memungkinkan terjadi penambahan properti-properti baru maupun pengurangan-pengurangan. Namun seharusnya hal tersebut tidak mengubah rancangan dasar dari ontologi.
e. Ada beberapa tahap lanjutan untuk pengembangan personalisasi pembelajaran online berbasis *semantic web* seperti terlihat pada Gambar 5.4. yang merupakan gambar rancangan pengembangan di Fakultas Ilmu

[8] Penulis menemukan pada forum bahwa masalah tersebut dapat diatasi dengan OWL 2.0. Namun seperti yang dialami oleh pengembang ontologi lainnya yang menemukan masalah ini dan menggunakan Protégé 4 (Protégé 4 mendukung OWL 2.0), penulis belum dapat memecahkan masalah ini.

Komputer UI [38]. Pembuatan materi pembelajaran dilakukan secara paralel dengan pengembangan sistem personalisasi. Pemberian semantik untuk setiap materi pembelajaran akan mengacu pada kerangka yang dihasilkan dalam tahap pengembangan kerangka semantik untuk setiap sumber daya pembelajaran (tahap nomor 2).



Gambar 5.4 Tahapan Pengembangan Sistem Personalisasi Pembelajaran Online Berbasis *Semantic Web*

Peningkatan Kualitas Pendidikan Melalui Sistem Personalisasi Berbasis *Semantic Web* dalam Student Centered e-Learning Environment (telah diolah kembali)

Penelitian ini adalah bagian dari bagan 2 yang ditunjukkan pada Gambar 5.4. Penelitian SHECAR dan pengembangan ontologi *learning object* yang paralel dengan penelitian ini juga merupakan bagian penelitian nomor 2. Tahap ini mencakup identifikasi terhadap sumber daya pembelajaran yang

relevan dengan teknik personalisasi yang dipilih. Setiap kelompok sumber daya yang relevan disusun kerangka semantiknya agar sumber daya pembelajaran tersebut nantinya memiliki makna yang bisa dievaluasi dan dikenali oleh sistem secara otomatis. Kerangka semantik tersebut meliputi format metadata untuk setiap sumber pembelajaran, penyusunan nilai-nilai yang mungkin setiap makna yang diberikan pada sumber daya pembelajaran tersebut, serta ontologi untuk mengatur keterkaitan antar sumber daya pembelajaran. Tahap ini masih memerlukan penyempurnaan, antara lain format metadata untuk sumber pembelajaran yang belum diidentifikasi serta integrasi ontologi seperti disebutkan pada butir d.

f. Salah satu tujuan *semantic web* adalah integrasi. Seharusnya aplikasi lain yang membutuhkan *student model ontology* di Fasilkom UI menggunakan ontologi yang sama. Jadi, sebaiknya suatu ontologi yang dimodelkan tidak hanya memperhatikan satu jenis kebutuhan saja, atau minimal tidak bertentangan dengan kebutuhan lainnya.

g. Sebaiknya ontologi disesuaikan dengan standar yang ditetapkan; seperti halnya *performance* sebaiknya dapat mengikuti standar PAPI. Namun sering kali, standar tidak sesuai dengan kebutuhan seperti dalam pengembangan *student model ontology* ini.