Appendix A: Chatbot Ontologies

SeqOnto.n3

```
@prefix :       <http://www.cs.rdg.ac.uk/seequel#> .

@prefix owl:    <http://www.w3.org/2002/07/owl#> .

@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .

@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .

@prefix foaf:   <http://xmlns.com/foaf/0.1/> .


:Quality                    a                       owl:Class ;

                            rdfs:subClassOf         owl:Thing ;

                            owl:oneOf               (

                                                    :lowQuality

                                                    :highQuality

                                                    ) ;

                            rdfs:label              "quality"@en ;

                            rdfs:comment            "type or level of

                                                    quality"@en .


:LearningExperienceArea     a                       owl:Class ;

                            rdfs:subClassOf         owl:Thing ;
```

rdfs:label "learning experience area"@en ;

rdfs:comment "an area of the learning experience, according to SEEQUEL there are three such areas in the e-Learning experience; context, sources and processes."@en .

:Role a owl:Class ;

rdfs:subClassOf owl:Thing ;

rdfs:label "role"@en ;

rdfs:comment "role of an individual in a specific sector."@en .

:Sector a owl:Class ;

rdfs:subClassOf owl:Thing ;

rdfs:label "sector"@en ;

rdfs:comment "the sector in which an individual works."@en .

:WorldView a owl:Class ;

rdfs:subClassOf owl:Thing ;

rdfs:label "world view"@en ;

rdfs:comment "world view is a representation of the values an individual has, according to Boltanski and Thevenot

there are six different world view; inspiration, domestic, opinion, civic, merchant and industrial."@en .

:Descriptor a owl:Class ;
rdfs:subClassOf owl:Thing ;
rdfs:comment "quality term used to evaluate quality."@en .

:ExperienceDescriptor a owl:Class ;
rdfs:subClassOf :Descriptor ;
rdfs:label "experience descriptor"@en ;
rdfs:comment "an experience descriptor is more specific than a general descriptor and can be more easily be measured or assessed"@en .

:GeneralDescriptor a owl:Class ;
rdfs:subClassOf :Descriptor ;
rdfs:label "general descriptor"@en ;
rdfs:comment "a quality term which cannot be directly measured and tends to be an abstract quality measure."@en .

:Criteria a owl:Class ;

```
                    rdfs:subClassOf     owl:Thing ;
                    rdfs:comment        "an area in which quality
can be measured in e-Learning"@en .

:SpecificCriteria   a                   owl:Class ;
                    rdfs:subClassOf     :Criteria ;
                    rdfs:label          "specific criteria"@en ;
                    rdfs:comment        "is a specific area of the
e-Learning domain that relates to quality"@en .

:GeneralCriteria    a                   owl:Class ;
                    rdfs:subClassOf     :Criteria ;
                    rdfs:label          "general criteria"@en ;
                    rdfs:comment        "a broad area of the e-
Learning domain with specific quality issues relevant to it"@en .


foaf:Group          a                   owl:Class ;
                    rdfs:subClassOf     owl:Thing ;
                    rdfs:label          "group"@en ;
                    rdfs:comment        "a group of people"@en .

foaf:Person         a                   owl:Class ;
```

	rdfs:subClassOf	owl:Thing ;
	rdfs:label	"person"@en ;
	rdfs:comment	"an individual"@en .
foaf:firstName	a	owl:DatatypeProperty ;
	rdfs:domain	foaf:Person ;
	rdfs:range	xsd:string ;
	rdfs:label	"first name"@en ;
	rdfs:comment	"the name of a person"@en .
foaf:surname	a	owl:DatatypeProperty ;
	rdfs:domain	foaf:Person ;
	rdfs:range	xsd:string ;
	rdfs:label	"surname"@en ;
	rdfs:comment	"the surname of a person"@en .
foaf:member	a	owl:ObjectProperty ;
	rdfs:domain	foaf:Group ;
	rdfs:range	foaf:Person ;
	rdfs:label	"member"@en ;

```
                    rdfs:comment        "a person who is a
member of a group."@en .

:MemberOf           a                   owl:ObjectProperty ;
                    rdfs:domain         foaf:Person ;
                    rdfs:range          foaf:Group ;
                    owl:inverseOf       foaf:member ;
                    rdfs:label          "member of"@en ;
                    rdfs:comment        "a group in which the
person is a member"@en .


:Relation           rdfs:subClassOf     rdf:statement ,
                            [
                            a                   owl:Restriction ;
                            owl:onProperty      :subject ;
                            owl:cardinality
                                "1"^^xsd:nonNegativeInteger
                            ] ,
                            [
                            a                   owl:Restriction ;
                            owl:onProperty      :predicate ;
```

```
                    owl:cardinality
                        "1"^^xsd:nonNegativeInteger
                    ] ,
                    [
                    a                   owl:Restriction ;
                    owl:onProperty      :object ;
                    owl:cardinality
                        "1"^^xsd:nonNegativeInteger
                    ] ;
                rdfs:label              "Relation"@en ;
                rdfs:comment            "internal class used to
represent relations between properties"@en .


:Claim          rdfs:subClassOf         :Relation ;
                owl:disjointWith        :Fact ;
                rdfs:label              "claim"@en ;
                rdfs:comment            "a statement or relation,
which has not been verified."@en .

:Fact           rdfs:subClassOf         :Relation ;
                owl:disjointWith        :Claim ;
                rdfs:label              "fact"@en ;
```

```
                    rdfs:comment        "a verified statement or
relation."@en .

:subject            a                   owl:ObjectProperty ;
                    rdfs:subPropertyOf  rdf:subject ;
                    rdfs:domain         :Relation .

:predicate          a                   owl:ObjectProperty ;
                    rdfs:subPropertyOf  rdf:predicate ;
                    rdfs:domain         :Relation .

:object             a                   owl:ObjectProperty ;
                    rdfs:subPropertyOf  rdf:object ;
                    rdfs:domain         :Relation .

:madeBy             a                   owl:ObjectProperty ;
                    rdfs:domain         :Relation ;
                    rdfs:range          [
                                        owl:unionOf
                                        ( foaf:Group foaf:Person )
                                        ] ;
                    rdfs:label          "made by"@en ;
```

```
                    rdfs:comment        "the person or group of
persons who has made a claim or stated a fact."@en .

:contributesTo      a                   owl:ObjectProperty ,
                                        owl:TransitiveProperty ;
                    rdfs:domain         :Descriptor ;
                    rdfs:range          [
                                        owl:unionOf
                                        ( :Descriptor :Quality )
                                        ] ;
                    rdfs:label          "contributes to"@en ;
                    rdfs:comment        "implication that one
descriptor contributes to the other"@en .

:describes          a                   owl:ObjectProperty ;
                    rdfs:domain         :Descriptor ;
                    rdfs:range          :LearningExperienceArea
;
                    rdfs:label          "describes"@en ;
                    rdfs:comment        "Indicates which area of
the learning experience a quality term describes"@en .
```

```
:implies        a               owl:ObjectProperty ,
                                owl:TransitiveProperty ;
                rdfs:domain     :Descriptor ;
                rdfs:range      [
                                owl:unionOf
                                ( :Descriptor :Quality )
                                ] ;
                rdfs:label      "implies"@en ;
                rdfs:comment    "Indicates that a
descriptor implies that another descriptor may also apply to the same entity being
evaluated"@en .


:impliesNot     a               owl:ObjectProperty ;
                rdfs:domain     [
                                owl:unionOf
                                ( :Descriptor :Quality )
                                 ] ;
                rdfs:range      [
                                owl:unionOf
                                 ( :Descriptor :Quality )
                                ] ;
                rdfs:label      "does not imply"@en ;
```

rdfs:comment "Indicates that a descriptor implies that another descriptor does not apply to the same entity"@en .

:partOf a owl:ObjectProperty , owl:TransitiveProperty ;
rdfs:domain :Sector ;
rdfs:range :Sector ;
rdfs:label "part of"@en ;
rdfs:comment "Indicates that a sector is a sub sector of another"@en .

:typeOf a owl:ObjectProperty ;
rdfs:domain :Role ;
rdfs:range :Role ;
rdfs:label "type of"@en ;
rdfs:comment "Indicates that a role is a type of another role (eg tutor is a type of teacher)"@en .

:workInSector a owl:ObjectProperty ;
rdfs:domain foaf:Person ;
rdfs:range :Sector ;
rdfs:label "works in sector" ;

rdfs:comment "the sector in which a person works."@en .

:hasRole a owl:ObjectProperty ;
rdfs:domain foaf:Person ;
rdfs:range :Role ;
rdfs:label "has role"@en ;
rdfs:comment "the role a person has in a specific sector."@en .

:hasWorldView a owl:ObjectProperty ;
rdfs:domain foaf:Person ;
rdfs:range :WorldView ;
rdfs:label "has world view"@en ;
rdfs:comment "the world view of a person"@en .

:areaOfInterest a owl:ObjectProperty ;
rdfs:domain foaf:Person ;
rdfs:range :LearningExperienceArea ;
rdfs:label "area of interest"@en ;

```
                    rdfs:comment        "areas in which a person
is particularly interested."@en .

:hasMade            a                   owl:ObjectProperty ;
                    rdfs:domain         [
                                        owl:unionOf
                                        ( foaf:Group foaf:Person )
                                        ] ;
                    rdfs:range          :Relation ;
                    owl:inverseOf       :madeBy ;
                    rdfs:label          "has made"@en ;
                    rdfs:comment        "claim and facts a person
has made."@en .


:boring             a                   :GeneralDescriptor ;
                    rdfs:label          "boring"@en .

:accessible         a                   :GeneralDescriptor ;
                    rdfs:label          "accessible"@en .

:easyToUse          a                   :GeneralDescriptor ;
                    rdfs:label          "easy to use"@en .
```

```
:engaging        a            :GeneralDescriptor ;
                 rdfs:label   "engaging"@en .

:flexible        a            :GeneralDescriptor ;
                 rdfs:label   "flexible"@en .

:inaccessible    a            :GeneralDescriptor ;
                 rdfs:label   "inaccessible"@en .

:interesting     a            :GeneralDescriptor ;
                 rdfs:label   "interesting"@en .

:organised       a            :GeneralDescriptor ;
                 rdfs:label   "organised"@en .

:respected       a            :GeneralDescriptor ;
                 rdfs:label   "respected"@en .

:challenging     a            :ExperienceDescriptor ;
                 rdfs:label   "challenging"@en .

:collaboration   a            :ExperienceDescriptor ;
```

```
                    rdfs:label          "collaborative"@en .

:concise            a                   :ExperienceDescriptor ;
                    rdfs:label          "concise"@en .

:easyNavigation     a                   :ExperienceDescriptor ;
                    rdfs:label          "easy to navigate"@en .

:effective          a                   :ExperienceDescriptor ;
                    rdfs:label          "effective"@en .

:efficient          a                   :ExperienceDescriptor ;
                    rdfs:label          "efficient"@en .

:fun                a                   :ExperienceDescriptor ;
                    rdfs:label          "fun"@en .

:incomplete         a                   :ExperienceDescriptor ;
                    rdfs:label          "incomplete"@en .

:interactive        a                   :ExperienceDescriptor ;
                    rdfs:label          "interactive"@en .
```

:monotonous a :ExperienceDescriptor ;
rdfs:label "monotonous"@en .

:motivating a :ExperienceDescriptor ;
rdfs:label "motivating"@en .

:nonIntuitiveLayout a :ExperienceDescriptor ;
rdfs:label "having a non-intuitive layout"@en .

:offTopic a :ExperienceDescriptor ;
rdfs:label "off topic"@en .

:openMinded a :ExperienceDescriptor ;
rdfs:label "open minded"@en .

:qualifiedTeachers a :ExperienceDescriptor ;
rdfs:label "having qualified teachers"@en .

:relevant a :ExperienceDescriptor ;
rdfs:label "relevant"@en .

```
:routine        a             :ExperienceDescriptor ;
                rdfs:label    "routine"@en .

:studentSupport a             :ExperienceDescriptor ;
                rdfs:label    "supported by
students"@en .

:tolerantOfMistakes a         :ExperienceDescriptor ;
                rdfs:label    "tolerant of mistakes"@en
.

:tooLong        a             :ExperienceDescriptor ;
                rdfs:label    "too long"@en .

:tooSimple      a             :ExperienceDescriptor ;
                rdfs:label    "too simple"@en .

:trackable      a             :ExperienceDescriptor ;
                rdfs:label    "trackable"@en .

:culture        a             :GeneralCriteria ;
                rdfs:label    "culture"@en .
```

:delivery a :GeneralCriteria ;
rdfs:label "delivery"@en .

:designGuidancePrinciples a :GeneralCriteria ;
rdfs:label "design guidance principles"@en .

:evaluation a :GeneralCriteria ;
rdfs:label "evaluation"@en .

:financialRules a :GeneralCriteria ;
rdfs:label "financial rules"@en .

:guidanceRecruitmentTNA a :GeneralCriteria ;
rdfs:label "guidance and recruitment in the training needs analysis"@en .

:institutionalSetting a :GeneralCriteria ;
rdfs:label "institutional setting"@en .

:learningEnvironment a :GeneralCriteria ;

```
                                rdfs:label                  "learning
environment"@en .

:learningMaterials              a                           :GeneralCriteria ;
                                rdfs:label                  "learning material"@en .

:legislation                    a                           :GeneralCriteria ;
                                rdfs:label                  "legislation"@en .

:staffCVandQualifications       a                           :GeneralCriteria ;
                                rdfs:label                  "supporting and teaching
staff CV and qualifications"@en .

:technicalLearningEnvironment       a                       :GeneralCriteria ;
                                rdfs:label                  "technical learning
environment"@en .

:valueSystem                    a                           :GeneralCriteria ;
                                rdfs:label                  "value system"@en .

:learningSources                a                           :LearningExperienceArea
;
                                rdfs:label                  "learning sources"@en ;
```

rdfs:comment "examples of learning sources are: Learning materials, reference materials, components, teachers, teachers qualifications etc. "@en .

:learningContext a :LearningExperienceArea ;

rdfs:label "learning context"@en ;

rdfs:comment "examples of learning contexts are: Institutional setting, environments, culture, legislation, values etc."@en .

:learningProcesses a :LearningExperienceArea ;

rdfs:label "learning processes"@en ;

rdfs:comment "examples of learning processes are: Planning, design, delivery, evaluation etc."@en .

:lowQuality a :Quality ;

rdfs:label "low quality"@en .

:highQuality a :Quality ;

owl:differentFrom :lowQuality ;

rdfs:label "high quality"@en .

```
:classroomAssistant     a                   :Role ;
                        rdfs:label          "classroom assistant"@en
.

:etManager              a                   :Role ;
                        rdfs:label          "E&T manager"@en .

:learner                a                   :Role ;
                        rdfs:label          "learner"@en .

:learnersRepresentative a                   :Role ;
                        rdfs:label          "learner's
representative"@en .

:policymaker            a                   :Role ;
                        rdfs:label          "policymaker"@en .

:teacher                a                   :Role ;
                        rdfs:label          "teacher"@en .

:tutor                  a                   :Role ;
                        :typeOf             :teacher ;
                        rdfs:label          "tutor"@en .
```

```
:administration        a              :Sector ;
                       rdfs:label     "administration"@en .

:corporate             a              :Sector ;
                       rdfs:label     "corporate"@en .

:higherEducation       a              :Sector ;
                       rdfs:label     "higher education"@en .

:ict                   a              :Sector ;
                       rdfs:label     "ICT"@en .

:informalLearning      a              :Sector ;
                       rdfs:label     "informal learning"@en .

:primaryEducation      a              :Sector ;
                       rdfs:label     "primary education"@en .

:professionalAssociation a            :Sector ;
                       rdfs:label     "professional
association"@en .
```

:publishers a :Sector ;

rdfs:label "publishers"@en .

:schoolK12 a :Sector ;

rdfs:label "school K-12"@en .

:trainingServices a :Sector ;

rdfs:label "training services"@en .

:usersOfCorporateTrainingServices a :Sector ;

rdfs:label "users of corporate training services"@en .

:vocationalTraining a :Sector ;

rdfs:label "vocational training"@en .

:civic a :WorldView ;

rdfs:label "civic"@en ;

rdfs:comment "following is a list of values for this world view: The general will, the common interest, generosity, selfabnegation, sacrifice, pride, the group, collective action, collective entities (ideas, values, symbols and institutions)."@en .

:domestic a :WorldView ;

rdfs:label "domestic"@en ;

rdfs:comment "following is a list of values for this world view: Confidence, responsibility, merit, respectability, convention, dignity, tradition, hierarchy, rank; parents, children, generation; rules and confidence, principles; harmony; the natural; the duty."@en .

:industrial a :WorldView ;

rdfs:label "industrial"@en ;

rdfs:comment "following is a list of values for this world view: Progress, future, functionality, efficiency, optimality, performance, productivity, professionality, reliability, farsightedness, system."@en .

:inspiration a :WorldView ;

rdfs:label "inspiration"@en ;

rdfs:comment "following is a list of values for this world view: Singularity, difference, innovation, originality, irrationality, imaginary, spirituality, unconscious, chance."@en .

:merchant a :WorldView ;

rdfs:label "merchant"@en ;

rdfs:comment "following is a list of values for this world view: Wealth, money, luxury; business, fair deals, good deals,

bargain; interest, attentions to others; contract; competition, rivalry, opportunism, freedom."@en .

:opinion a :WorldView ;
rdfs:label "opinion"@en ;
rdfs:comment "following is a list of values for this world view: Image, reputation, fame, success, honour, acknowledgement, visibility, audience, credibility, identification. "@en .

:seequel a foaf:Group ;
foaf:member :karsten ;
rdfs:comment "Seequel"@en ;
rdfs:label "SEEQUEL"@en .

:karsten a foaf:Person ;
foaf:firstName "Karsten" ;
foaf:surname "Lundqvist" ;
rdfs:comment "Karsten Oster Lundqvist"@en ;
rdfs:label "Karsten Oster Lundqvist"@en .

<http://www.cs.rdg.ac.uk/seequel.n3> a owl:Ontology ;

rdfs:comment "The SEEQUEL e-Learning Ontology" ;

rdfs:label "e-Learning Ontology" .

Tp-feedback.n3

```
@prefix :           <http://www.cs.rdf.sc.uk/tp-feedback#> .

@prefix seequel:    <http://www.cs.rdg.ac.uk/seequel#> .

@prefix owl:        <http://www.w3.org/2002/07/owl#> .

@prefix rdfs:       <http://www.w3.org/2000/01/rdf-schema#> .

@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix xsd:        <http://www.w3.org/2001/XMLSchema#> .

@prefix foaf:       <http://xmlns.com/foaf/0.1/> .



:TP                 a                   seequel:learningSources ;

                    rdfs:label          "teaching product"@en ;

                    rdfs:comment        "a product, which is used
in a teaching environment."@en .


:unitOfTP           a                   owl:Class ;

                    rdfs:subClassOf     owl:Thing ;

                    rdfs:label          "units of a teaching
product"@en ;

                    rdfs:comment        "a unit of a teaching
product, such as different courses."@en .


:Functionality      a                   owl:Class ;
```

```
                    rdfs:subClassOf     owl:Thing ;
                    rdfs:label          "functionality"@en ;
                    rdfs:comment        "functionality of a
teaching product or parts of a teaching product."@en .


:DiscussionBoard    rdfs:subClassOf     :Functionality ;
                    rdfs:label          "discussion board"@en .

:Quiz               rdfs:subClassOf     :Functionality ;
                    rdfs:label          "quiz"@en .

:Assignment         rdfs:subClassOf     :Functionality ;
                    rdfs:label          "assignment"@en .

:Survey             rdfs:subClassOf     :Functionality ;
                    rdfs:label          "survey"@en .

:MailBox            rdfs:subClassOf     :Functionality ;
                    rdfs:label          "mailbox"@en .

:Content            rdfs:subClassOf     :Functionality ;
```

rdfs:label "content"@en ;

rdfs:comment "content of a teaching product or functional within a teaching product."@en .

:is a owl:ObjectProperty ;

rdfs:domain [

owl:unionOf (

:TP

:unitOfTP

:Functionality)

] ;

rdfs:range seequel:Descriptor ;

rdfs:label "is"@en ;

rdfs:comment "A relation, which indicates that a teaching product can be described by a descriptor from the Seequel framework (ontology)."@en .

:isNot a owl:ObjectProperty ;

rdfs:domain [

owl:unionOf (

:TP

:unitOfTP

:Functionality)

] ;

rdfs:range seequel:Descriptor ;

rdfs:label "is not"@en ;

rdfs:comment "A relation, which indicates that a teaching product can be described negatively by a descriptor from the Seequel framework (ontology)."@en .

:noRelation a owl:ObjectProperty ;

rdfs:domain [

owl:unionOf (

:TP

:unitOfTP

:Functionality)

] ;

rdfs:range seequel:Descriptor ;

rdfs:label "has no relation to"@en ;

rdfs:comment "A relation is not known to exist between a teaching product and the descriptor from the Seequel framework (ontology)."@en .

:hasUnit a owl:ObjectProperty ;

```
                    rdfs:domain         :TP ;
                    rdfs:range          :unitOfTP ;
                    rdfs:label          "has unit"@en ;
                    rdfs:comment        "A teaching product has
the given unit."@en .

:unitOf             a                   owl:ObjectProperty ;
                    rdfs:domain         :unitOfTP ;
                    rdfs:range          :TP ;
                    owl:inverseOf       :hasUnit ;
                    rdfs:label          "unit of"@en ;
                    rdfs:comment        "A unit is part of a
teaching product."@en .

:hasFunctionality   a                   owl:ObjectProperty ;
                    rdfs:domain         [
                                         owl:unionOf (
                                         :TP
                                        :unitOfTP
                                        :Functionality )
                                        ] ;
                    rdfs:range          :Functionality ;
                    rdfs:label          "has functionality"@en ;
```

```
                    rdfs:comment        "A teaching product or
unit of a tp has a certain functionality"@en .

:functionalityOf    a                   owl:ObjectProperty ;
                    rdfs:domain         :Functionality ;
                    rdfs:range          [
                                         owl:unionOf (
                                         :TP
                                        :unitOfTP
                                        :Functionality )
                                         ] ;
                    rdfs:label          "functionality of"@en ;
                    rdfs:comment        "functionality of a
teaching product"@en .

:bb                 a                   :TeachingProduct ;
                    rdfs:label          "Blackboard"@en ;
                    rdfs:comment        "Blackboard is a teaching
product, which is used at Reading University."@en ;
                    :hasFunctionality   :bbDB ;
```

```
                    :hasFunctionality       :bbTest ;
                    :hasFunctionality       :bbSurvey ;
                    :hasFunctionality       :bbMailBox ;
                    :hasFunctionality       :bbQuiz ;
                    rdfs:seeAlso        <http://www.bb.rdg.ac.uk> .

:bbDB               a                       :DiscussionBoard ;
                    rdfs:label              "discussion boards on
Blackboard"@en .

:bbAssignments      a                       :Assignment ;
                    rdfs:label              "assignments on
Blackboard"@en .

:bbSurvey           a                       :Survey ;
                    rdfs:label              "surveys on
Blackboard"@en .

:bbMailBox          a                       :MailBox ;
                    rdfs:label              "mailboxes on
Blackboard"@en .

:bbQuiz             a                       :Quiz ;
```

rdfs:label "quizzes on Blackboard"@en .

:programming a :unitOfTP ;
:unitOf :bb ;
:hasFunctionality :prgDB ;
:hasFunctionality :prgQuiz ;
rdfs:label "Programming"@en ;
rdfs:comment "The programming course with the module number SE1SA5."@en ;
rdfs:seeAlso <http://www.info.rdg.ac.uk/module/0506/SC/SE1SA5.htm> .

:prgDB a :DiscussionBoard ;
:hasFunctionality :cPrgDB ;
rdfs:label "discussion board in programming"@en .

:cPrgDB a :Content ;
rdfs:label "content of the discussion board in programming"@en .

:prgAssignment a :Assignment ;

:hasFunctionality :cPrgAssignment ;

rdfs:label "assignments in programming"@en .

:cPrgAssignment a :Content ;

:hasFunctionality :cPrgAssignment ;

rdfs:label "content of the assignments in programming"@en .

<http://www.cs.rdg.ac.uk/tp-feedback.owl>

a owl:Ontology ;

rdfs:comment "The Teaching Product Feedback ontology" ;

rdfs:label "teaching product feedback Ontology" .

:claim a seequel:Claim ;

:subject seequel:easyToUse ;

```
                    :predicate          :is ;
                    :object             :bbDB ;
                    :madeBy             seequel:Karsten .

:claim1             a                   seequel:Claim ;
                    :subject            seequel:easyToNavigate ;
                    :predicate          :isNot ;
                    :object             :bbDB ;
                    :madeBy             seequel:Karsten .

:claim2             a                   seequel:Claim ;
                    :subject            seequel:nonIntuitiveLayout ;
                    :predicate          :is ;
                    :object             :bbDB ;
                    :madeBy             seequel:Karsten .

:claim3             a                   seequel:Claim ;
                    :subject            seequel:easyToUse ;
                    :predicate          :is ;
                    :object             :bbAssignments;
                    :madeBy             seequel:Karsten .

:claim4             a                   seequel:Claim ;
```

```
                    :subject        seequel:easyNavigation;
                    :predicate      :is ;
                    :object         :bbAssignments;
                    :madeBy         seequel:Karsten .

:claim5             a               seequel:Claim ;
                    :subject        seequel:flexible;
                    :predicate      :isNot ;
                    :object         :bbMailBox;
                    :madeBy         seequel:Karsten .

:claim6             a               seequel:Claim ;
                    :subject        seequel:organised ;
                    :predicate      :is ;
                    :object         :prgDB ;
                    :madeBy         seequel:Karsten .

:claim7             a               seequel:Claim ;
                    :subject        seequel:easyNavigation ;
                    :predicate      :is ;
                    :object         :prgDB ;
                    :madeBy         seequel:Karsten .
```

```
:claim8        a              seequel:Claim ;
               :subject       seequel:relevant ;
               :predicate     :isNot ;
               :object        :cPrgDB ;
               :madeBy        seequel:Karsten .

:claim9        a              seequel:Claim ;
               :subject       seequel:offTopic ;
               :predicate     :is ;
               :object        :cPrgDB  ;
               :madeBy        seequel:Karsten .

:claim10       a              seequel:Claim ;
               :subject       seequel:tooLong ;
               :predicate     :is ;
               :object        :cPrgDB ;
               :madeBy        seequel:Karsten .

:claim11       a              seequel:Claim ;
               :subject       seequel:nonIntuitiveLayout ;
               :predicate     :isNot ;
               :object        :prgAssignment ;
               :madeBy        seequel:Karsten .
```

```
:claim12        a               seequel:Claim ;
                :subject        seequel:interactive ;
                :predicate      :is ;
                :object         :prgAssignment ;
                :madeBy         seequel:Karsten .

:claim13        a               seequel:Claim ;
                :subject        seequel:organised ;
                :predicate      :is ;
                :object         :prgAssignment  ;
                :madeBy         seequel:Karsten .

:claim14        a               seequel:Claim ;
                :subject        seequel:challenging ;
                :predicate      :is ;
                :object         :cPrgAssignment ;
                :madeBy         seequel:Karsten .

:claim15        a               seequel:Claim ;
                :subject        seequel:boring ;
                :predicate      :isNot ;
                :object         :cPrgAssignment ;
```

	:madeBy	seequel:Karsten .
:claim16	a	seequel:Claim ;
	:subject	seequel:tooSimple ;
	:predicate	:is ;
	:object	:cPrgAssignment ;
	:madeBy	seequel:Karsten .

Appendix B: OwlLang JavaCC code

```
options {
  LOOKAHEAD = 3;
  FORCE_LA_CHECK = true;
  STATIC = false;
}



PARSER_BEGIN(OwlLang)
package org.UoR.OwlLang;



import java.io.*;
import java.util.*;
import java.util.regex.*;


//Libraries used to communicate with ontologies.
import com.hp.hpl.jena.ontology.*;
import com.hp.hpl.jena.query.*;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.iterator.* ;
import org.UoR.GrammaCheck.*;
```

```
public class OwlLang
{
	public static class Types
	{
		public static final int CLASS = 0;
		public static final int PROPERTY = 1;
		public static final int INDIVIDUAL = 2;
	}
	private int investigateType;

	private ArrayList ONTresult;
	private OntModel ontmodel;
	private String base;
	private String prefix;
	private GrammaCheck gramma;

	private Random rand;

	private String thatReturn="";

	/**An unknown AIML input*/
	protected static final String O_UNKNOWN = "UNKNOWN";
```

```
//URI of rdfs:label is the default return string of any resource in an OntModel
protected static final String defaultReturnString = "rdfs:label";

public OwlLang(OntModel ontmodel, String baseURI)
{
        this(new StringReader("test"));

        Long seed = System.nanoTime();
        this.rand = new Random(seed);

        this.ontmodel = ontmodel;

        base = "BASE <" + baseURI + ">\n";
        prefix = "";
        setLabeledPrefix("rdfs","http://www.w3.org/2000/01/rdf-schema#");
        setLabeledPrefix("owl","http://www.w3.org/2002/07/owl#");

        gramma = new GrammaCheck();
}

public void setLabeledPrefix(String label, String URI)
{
```

```
        this.prefix += "PREFIX " + label + ": <" + URI + "> \n";
}



public String query(String input) throws ParseException
{
        thatReturn="";
        ONTresult = new ArrayList();
        StringReader inp = new StringReader(input);
        this.ReInit(inp);
        String res = this.OntQue();
        return res;
}

public String getThatReturn()
{
        return this.thatReturn;
}

private Object getSingleObject(Set possibleInstances)
{
        Object winner = null;;
```

```
		if (possibleInstances.size()!=0)
		{
			Object [] res = possibleInstances.toArray();
			winner = res[this.rand.nextInt(possibleInstances.size())];
		}

		return winner;
	}



	private String getPropertyVal(String subject, String property, String
stringToken)
	{
		if(property==null) return subject;

		String queryString =
			base + prefix +
			"SELECT ?str " +
			"WHERE { " + subject + " " + property +	" ?str . }";

		Query query = QueryFactory.create(queryString);
		QueryExecution qe = QueryExecutionFactory.create(query,ontmodel);
		ResultSet rs = qe.execSelect();
```

```
HashSet solutions = new HashSet();

for (;rs.hasNext();)
{
    QuerySolution qs = rs.nextSolution();
    RDFNode rn = qs.get("str");
    if(rn.isLiteral())
    {
        String lang = ((Literal)rn).getLanguage();
        if(lang==null && stringToken==null)
            solutions.add("\"" +
((Literal)rn).getLexicalForm() + "\"");
        else
            if(lang!=null && stringToken!=null)
            {
                if(lang.equals(stringToken))
                    solutions.add("\"" +
((Literal)rn).getLexicalForm() + "\"");
            }
    }
    else
        solutions.add("<" + ((Resource)rn).getURI() + ">");
```

```
        }

        qe.close();

        if(solutions.size()==0) return "\"" + O_UNKNOWN + "\"";
        Object [] reses = solutions.toArray();
        String res = (String)reses[this.rand.nextInt(solutions.size())];

        return res ;
    }

    private String getString(Token t)
    {
        if(t==null)
            return null;
        else
            return t.image.substring(1,t.image.length()-1);
    }

    private int getNumber(Token t) throws ParseException
    {
        if(t!=null)
            return Integer.parseInt(t.image.substring(1));
```

```
		else
			throw new ParseException("null pointer error in
getNumber(Token t)");
	}

	private String getURI(Token t)
	{
		if(t!=null)
			return "<" + ontmodel.expandPrefix(getString(t)) + ">";
		else
			return null;
	}

	private String getURI(String s)
	{
		if(s==null)
			return null;
		else
			return "<" + ontmodel.expandPrefix(s) + ">";
	}

	private String getAIMLURI(String s)
	{
```

```
        s = getPureURI(s);
        return ontmodel.shortForm(s);
}

private String getPureURI(String str)
{
        if(str==null)
                return null;
        String foo = str.substring(1,str.length()-1);
        return  ontmodel.expandPrefix(foo);
}

private String getPureURIfromAIML(String str)
{
        if(str==null)
                return null;

        String foo = str.substring(1,str.length()-1);

        if(!foo.contains(":"))
        {
                int pos = foo.indexOf(" ");
```

```
		if(pos==-1)
			return ontmodel.expandPrefix(":" + foo);

		foo = foo.substring(0,pos) + ":" + foo.substring(pos+1);
	}

	return ontmodel.expandPrefix(foo);
}

private String getPureURIfromAIML(Token t)
{

	if(t==null)
		return null;

	return getPureURIfromAIML(t.image);
}

private String getFullString(String propValue,String stringToken)
{
	if(propValue==null)
		return null;
```

```
        if(stringToken!=null)
                return " \"" + propValue + "\"@" + stringToken;
        else
                return " \"" + propValue + "\"";
}

private Set getSetFromIterator(Iterator i)
{
        HashSet res = new HashSet();

        for(;i.hasNext();)
        {
                String id = ((Resource)i.next()).getURI();
                if(id!=null)
                {
                        id=getURI(id);
                        res.add(id);
                }
        }

        return res;
}
```

```
private boolean isStringInSet(String test, Set set)
{
	if(test==null)
		return false;

	return set.contains(test);
}

private ExtendedIterator iteratorOverDomain(Resource res)
{
	String URI = res.getURI();

	if(URI!=null)
	{
		SingletonIterator si = new SingletonIterator(res);
		return si;
	}

	try
	{
		ExtendedIterator ni = new NiceIterator();

		UnionClass uc = (UnionClass)res.as(UnionClass.class);
```

```
                Iterator i = uc.listOperands();
                for(;i.hasNext();)
                {
                        Resource next = (Resource)i.next();
                        ni = ni.andThen(iteratorOverDomain(next));
                }

                return ni;
        } catch(Exception e)
        {
                return null;
        }
    }

    private Set getCLSes(String propName,Token cType,String propValue)
throws ParseException
    {
        Set solutions = new HashSet();

        if(propName==null)
        {
                solutions = new HashSet();
                OntClass oc = ontmodel.getOntClass(getPureURI(propValue));
```

```
                if(oc!=null)
                {
                        String id = oc.getURI();
                        if(id!=null)
                        {
                                id = getURI(id);
                                solutions.add(id);
                        }
                }
        }
        else
        {
                String queryString =
                        base + prefix +
                        "SELECT ?cls\n" +
                        "WHERE { ?cls a owl:Class . ?cls " + propName + " " +
propValue + " . }";

                Query query = QueryFactory.create(queryString);

                // Execute the query and obtain results
                QueryExecution qe = QueryExecutionFactory.create(query,
ontmodel);
```

```
        ResultSet results = qe.execSelect();

        for(;results.hasNext();)
        {
            QuerySolution qs = results.nextSolution();
            Resource rn = qs.getResource("cls");
            if(rn!=null)
            {
                String id = rn.getURI();
                if(id!=null)
                {
                    id = getURI(id);
                    solutions.add(getURI(id));
                }
            }
        }

        // Free up resources used running the query
        qe.close();
    }

    //Add subclasses to solution
    if(cType.image.equals("+="))
```

```
{
        Iterator i = solutions.iterator();
        for(;i.hasNext();)
        {
                OntClass oc =
ontmodel.getOntClass(getPureURI((String)i.next()));
                if(oc!=null)
                {
                        Iterator ei = oc.listSubClasses();
                        Set subClasses = getSetFromIterator(ei);
                        solutions.add(subClasses);
                }
        }
}

//It's a negation
if(cType.image.equals("!="))
{
        Set results = new HashSet();
        Iterator ei = ontmodel.listClasses();
        Set allClasses = getSetFromIterator(ei);
        allClasses.removeAll(solutions);
        solutions = allClasses;
```

```
}

switch (investigateType)
{
    case Types.CLASS:
        return solutions;

    case Types.PROPERTY:
        Set result = new HashSet();
        Iterator i = solutions.iterator();
        for(;i.hasNext();)
        {
            OntClass oc =
ontmodel.getOntClass(getPureURI((String)i.next()));
            if(oc!=null)
            {
                Iterator pi = oc.listDeclaredProperties();
                result.addAll(getSetFromIterator(pi));
            }
        }
        return result;

    case Types.INDIVIDUAL:
```

```
                    result = new HashSet();
                    i = solutions.iterator();
                    for(;i.hasNext();)
                    {
                        OntClass oc =
ontmodel.getOntClass(getPureURI((String)i.next()));
                        if(oc!=null)
                        {
                            Iterator ii = oc.listInstances();
                            result.addAll(getSetFromIterator(ii));
                        }
                    }
                    return result;

                default:
                    throw new ParseException("Illigal investigation type in
cls call");
            }
        }

        private Set getPROPERTYs(String propName,Token cType,String propValue,
Token t,String indSlotName,Token indType,String indValue) throws ParseException
        {
```

```
Set solutions = new HashSet();

if(propName==null)
{
    solutions = new HashSet();
    OntProperty op =
ontmodel.getOntProperty(getPureURI(propValue));
    if(op!=null)
    {
        String id = op.getURI();
        if(id!=null)
        {
            id = getURI(id);
            solutions.add(id);
        }
    }
}
else
{
    String queryString =
        base + prefix +
        "SELECT ?prop\n" +
```

```
                "WHERE { ?s ?prop ?o . ?prop " + propName + " " +
propValue + " . }";

        Query query = QueryFactory.create(queryString);

        // Execute the query and obtain results
        QueryExecution qe = QueryExecutionFactory.create(query,
ontmodel);
        ResultSet results = qe.execSelect();

        for(;results.hasNext();)
        {
            QuerySolution qs = results.nextSolution();
            Resource rn = qs.getResource("prop");
            if(rn!=null)
            {
                String id = rn.getURI();

                if(id!=null)
                {
                    id = getURI(id);
                    solutions.add(id);
                }
```

```
                }

            }

            // Free up resources used running the query
            qe.close();
        }

        //It's a negation
        if(cType.image.equals("!="))
        {
            Set res = new HashSet();
            ExtendedIterator iP = ontmodel.listObjectProperties();
            ExtendedIterator iProp =
iP.andThen(ontmodel.listDatatypeProperties());
            res.addAll(getSetFromIterator(iProp));

            res.removeAll(solutions);
            solutions=res;
        }

        switch (investigateType)
        {
```

```
                    case Types.CLASS:
                        //Returns domain class of Property
                        Set result = new HashSet();
                        Iterator i = solutions.iterator();
                        for(;i.hasNext();)
                        {
                            OntProperty op =
ontmodel.getOntProperty(getPureURI((String)i.next()));
                            if(op!=null)
                            {
                                Iterator pi =
((OntProperty)op).listDeclaringClasses();
                                for(;pi.hasNext();)
                                {
                                    Resource prop =
(Resource)pi.next();
                                    if(prop!=null)
                                    {
                                        Iterator iprop =
iteratorOverDomain(prop);

        result.addAll(getSetFromIterator(iprop));
```

```
                                        }
                                }
                        }
                }
                return result;

        case Types.PROPERTY:
                if(t==null)
                        return solutions;

                result = new HashSet();
                Set negationSet = new HashSet();

                i=solutions.iterator();
                for(;i.hasNext();)
                {
                        String queryString;

                        String nextProp = (String)i.next();
                        if(indType.image.equals("!="))
                        {
                                queryString = base + prefix + "SELECT
?ind\n";
```

```
if(indSlotName==null)
        queryString += "WHERE { ?ind
" + nextProp + " ?object . }";
else
        queryString += "WHERE { ?ind
" + nextProp + " ?o .\n?o " + indSlotName + " ?object . }";

Query query =
QueryFactory.create(queryString);
// Execute the query and obtain results
QueryExecution qe =
QueryExecutionFactory.create(query, ontmodel);
ResultSet rs = qe.execSelect();

for(;rs.hasNext();)
{
        QuerySolution qs =
rs.nextSolution();
        Resource rn =
qs.getResource("ind");

        if(rn!=null)
        {
```

```
                                                    Iterator iprop =
iteratorOverDomain(rn);

        negationSet.addAll(getSetFromIterator(iprop));
                                                    }
                                            }

                                            qe.close();
                                    }

                                    queryString = base + prefix + "SELECT ?ind\n";
                                    if(indSlotName==null)
                                            queryString += "WHERE { ?ind " +
nextProp + " " + indValue + " . }";
                                    else
                                            queryString += "WHERE { ?ind " +
nextProp + " ?o .\n?o " + indSlotName + " " + indValue + " . }";

                                    Query query =
QueryFactory.create(queryString);
                                    // Execute the query and obtain results
                                    QueryExecution qe =
QueryExecutionFactory.create(query, ontmodel);
```

```
ResultSet rs = qe.execSelect();

for(;rs.hasNext();)
{
    QuerySolution qs = rs.nextSolution();
    Resource rn = qs.getResource("ind");

    if(rn!=null)
    {
        Iterator iprop =
iteratorOverDomain(rn);

result.addAll(getSetFromIterator(iprop));
    }
}

qe.close();
}

if(indType.image.equals("=="))
    return result;
else
```

```
        {
            negationSet.removeAll(result);
            return negationSet;
        }



    case Types.INDIVIDUAL:
        result = new HashSet();

        if(t!=null)
        {
            negationSet = new HashSet();

            i=solutions.iterator();
            for(;i.hasNext();)
            {
                String queryString;

                String nextProp = (String)i.next();
                if(indType.image.equals("!="))
                {
```

```
queryString = base + prefix +
"SELECT ?ind\n";
if(indSlotName==null)
    queryString += "WHERE
{ ?ind " + nextProp + " ?object . }";
else
    queryString += "WHERE
{ ?ind " + nextProp + " ?o .\n?o " + indSlotName + " ?object . }";

Query query =
QueryFactory.create(queryString);

// Execute the query and obtain
results
QueryExecution qe =
QueryExecutionFactory.create(query, ontmodel);
ResultSet rs = qe.execSelect();

for(;rs.hasNext();)
{
    QuerySolution qs =
rs.nextSolution();
    Resource rn =
qs.getResource("ind");
```

```
                    if(rn!=null)
                    {
                        Iterator iprop =
iteratorOverDomain(rn);

        negationSet.addAll(getSetFromIterator(iprop));
                    }
                }

                qe.close();
            }

            queryString = base + prefix + "SELECT
?ind\n";
            if(indSlotName==null)
                queryString += "WHERE { ?ind
" + nextProp + " " + indValue + " . }";
            else
                queryString += "WHERE { ?ind
" + nextProp + " ?o .\n?o " + indSlotName + " " + indValue + " . }";
```

```
Query query =
QueryFactory.create(queryString);

// Execute the query and obtain results
QueryExecution qe =
QueryExecutionFactory.create(query, ontmodel);
ResultSet rs = qe.execSelect();

for(;rs.hasNext();)
{
    QuerySolution qs =
rs.nextSolution();
    Resource rn =
qs.getResource("ind");

    if(rn!=null)
    {
        Iterator iprop =
iteratorOverDomain(rn);

    result.addAll(getSetFromIterator(iprop));
    }
}
```

```
                    qe.close();
                }

                if(indType.image.equals("=="))
                    return result;
                else
                {
                    negationSet.removeAll(result);
                    return negationSet;
                }
            }

            Set classResult = new HashSet();
            i = solutions.iterator();
            for(;i.hasNext();)
            {
                OntProperty op =
ontmodel.getOntProperty(getPureURI((String)i.next()));
                if(op!=null)
                {
                    Iterator pi =
((OntProperty)op).listDeclaringClasses();
```

```
classResult.addAll(getSetFromIterator(pi));
                    }
            }

            i = classResult.iterator();
            for(;i.hasNext();)
            {
                    try
                    {
                            Iterator iclasses =
iteratorOverDomain((Resource)i.next());

                            for(;iclasses.hasNext();)
                            {
                                    Resource clsRes =
(Resource)iclasses.next();

                                    if(clsRes!=null)
                                    {
                                            OntClass oc =
ontmodel.getOntClass(((Resource)clsRes).getURI());
                                            if(oc!=null)
```

```
                                                                {
                                                                        Iterator iIns =
oc.listInstances();

            result.addAll(getSetFromIterator(iIns));
                                                                }
                                                        }
                                                }
                                        } catch(Exception e)
                                        {
                                        }
                                }

                                return result;

                        default:
                                throw new ParseException("Illigal investigation type in
cls call");
                }
        }

        private Set getINDes(String propName,Token cType,String propValue)
throws ParseException
```

```
{
        if(investigateType==Types.PROPERTY)
                return new HashSet();

        Set solutions = new HashSet();



        if(propName==null)
        {
                solutions = new HashSet();
                Individual oi =
ontmodel.getIndividual(getPureURI(propValue));
                if(oi!=null)
                {
                        String id = oi.getURI();
                        if(id!=null)
                        {
                                id = getURI(id);
                                solutions.add(id);
                        }
                }
        }
        else
```

```
{
        String queryString =
            base + prefix +
            "SELECT ?ind\n" +
            "WHERE { ?ind a ?s . ?s a owl:Class . ?s
rdfs:subClassOf owl:Thing . ?ind " + propName + " " + propValue + " . }";

        Query query = QueryFactory.create(queryString);

        // Execute the query and obtain results
        QueryExecution qe = QueryExecutionFactory.create(query,
ontmodel);
        ResultSet results = qe.execSelect();

        for(;results.hasNext();)
        {
            QuerySolution qs = results.nextSolution();
            Resource rn = qs.getResource("ind");
            if(rn!=null)
            {
                String id = rn.getURI();
                if(id!=null)
                {
```

```
                                        id = getURI(id);
                                        solutions.add(id);
                                }
                        }
                }

                // Free up resources used running the query
                qe.close();
        }

        //It's a negation
        if(cType.image.equals("!="))
        {
                Set res = new HashSet();

                String queryString =
                        base + prefix +
                        "SELECT ?ind\n" +
                        "WHERE { ?ind a ?s . ?s a owl:Class . ?s
rdfs:subClassOf owl:Thing . }";

                Query query = QueryFactory.create(queryString);
```

```
// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query,
ontmodel);
ResultSet results = qe.execSelect();

for(;results.hasNext();)
{
    QuerySolution qs = results.nextSolution();
    Resource rn = qs.getResource("ind");
    String rnName = rn.getURI();

    if(rnName!=null)
    {
        rnName = getURI(rnName);
        res.add(rnName);
    }
}

// Free up resources used running the query
qe.close();

res.removeAll(solutions);
```

```
                solutions = res;
            }

            switch (investigateType)
            {
                case Types.CLASS:
                    Set res = new HashSet();
                    Iterator i = solutions.iterator();
                    for(;i.hasNext();)
                    {
                        String indURI = getPureURI((String)i.next());
                        Individual ind =
ontmodel.getIndividual(indURI);

                        if(ind!=null)
                        {
                            Iterator iCls = ind.listRDFTypes(true);
                            res.addAll(getSetFromIterator(iCls));
                        }
                    }
                    return res;

                case Types.INDIVIDUAL:
```

```
                return solutions;

            default:
                throw new ParseException("Illigal investigation type in
cls call");
        }
    }

    private Set getRESTs(Token cType,String propValue,Token t) throws
ParseException
    {
        return null;
    }

    private Resource getUniqueResource(String str)
    {
        String uri = ontmodel.expandPrefix(":" + str);
        Resource res = ontmodel.createResource(uri);

        while(ontmodel.containsResource(res))
        {
            uri = ontmodel.expandPrefix(":" + str + System.nanoTime());
            res = ontmodel.createResource(uri);
```

```
                }

                return res;
        }
}

PARSER_END(OwlLang)

SKIP : /* WHITE SPACE AND COMMENTS*/
{
        " "
|       "\t"
|       "\n"
|       "\r"
|       "\f"
}

SPECIAL_TOKEN : /* COMMENTS */
{
  <SINGLE_LINE_COMMENT: "//" (~["\n","\r"])* ("\n"|"\r"|"\r\n")>
| <MULTI_LINE_COMMENT: "/*" (~["*"])* "*" ("*" | (~["*","/"] (~["*"])* "*"))*
"/">
}
```

```
TOKEN :
{
        <ONTQUE:        "$OWLLANG">                     //General
command
|       <ONTSTORE:               "ONTSTORE">                     //Store
single statement or reified statement with one extra property
|       <ONTUR:                  "ONTUNIQUEID">          //Retrieve unique
resource
|       <ONTISR:          "ONTISRESOURCE">        //is resource (return
unknown if not known else the URI.
|       <ONTGRAMMA:              "ONTGRAMMA">            //Gramma check
|       <ONTREGEX:               "ONTREGEX">                     //Regular
expresion check
|       <ONTIND:         "ONTIND">               //Find individual
|       <ONTCLS:         "ONTCLS">               //Find class
|       <ONTPRO:         "ONTPRO">               //Find property
|       <CLS:            "cls">                  //class
|       <PROPERTY:       "prop">
|       <IND:            "ind">                  //individual
|       <REST:                    "rest">                         //NOT
IMPLEMENTED
|       <THAT:                    "THAT">                         //that
return part
```

```
|       <COLON:                   ":">
|       <LPAREN:          "(" >
|       <RPAREN:          ")" >
|       <ASSIGN:          "=" >
|       <EQ:              "==" >
|       <NQ:              "!=" >
|       <PLUSEQ:          "+=" >
|       <SC_OR:           "OR" >
|       <SC_AND:          "AND" >
|       <SC_XOR:          "XOR" >
|       <GETIND:          ">>" >
|       <REFNUMBER:       "#" ["1"-"9"] ( ["0"-"9"] )* >
|       <STRING:          """ ( (~["","\n","\r"]) )* """ >
|       <URI:             "[" ( (~["<",">","{","}","|","[","]","\\","^","~","\n","\r"] )
)* "]" > // Not allowed char in URI (BNF): { | } | vline | [ | ] | \ | ^ | ~ | < | >
}

String OntQue() :
{
        String res;
        Token t;
}
{
```

```
        t=<ONTQUE> res=CommandList() ThatReturnPart() <EOF>
        {
                return t.image + " " + res;
        }
}



//Returns the strin
String CommandList() :
{
        String res,r1;
}
{
        <LPAREN> res=CommandPart() ( r1=CommandPart() {res+=" "+r1;} )*
<RPAREN>
        {
                return res;
        }
}


//Initiates the Collection to find depending on command
//Return a string with "<command-type> <single-fit from all possible>"
String CommandPart() :
```

```
{
        String res,resIns, regMatch;
        Token uriToken = null;
        Set possibles = null;
        Token stringToken=null;

        Resource subj, pred;
        Resource say=null;
        RDFNode obj;
        RDFNode what=null;

        Token t;

        Token strg, exp;
}
{
        res=Command() possibles=PredicateStatement() [ ":" uriToken=<URI> ] [
"@" stringToken=<STRING> ]
        {
                String str = (String)getSingleObject(possibles);

                if(str!=null)
                {
```

```
                resIns =
getPropertyVal(str,getURI(getPureURIfromAIML(uriToken)),
getString(stringToken));
                ONTresult.add(resIns);
                res +=" " + getAIMLURI(resIns);
                return res;
            }
            else
            {
                resIns = """ + O_UNKNOWN + """;
                ONTresult.add(resIns);
                res +=" " + getAIMLURI(resIns);
                return res;
            }
        }
        |
        <ONTSTORE> <LPAREN> subj=claimRefURI() pred=claimRefURI()
obj=claimRefStringURI() [ say=claimRefURI() what=claimRefStringURI() ]
<RPAREN>
        {
            //TODO: Make generic. But first off ok.
            try
            {
```

```
                    if(say==null)
                    {

        ontmodel.add(ontmodel.createStatement(subj,(Property)pred.as(Property.class
),obj));

                        res = getURI(subj.getURI());
                        ONTresult.add(res);
                        return "ONTSTORE " + getAIMLURI(res);
                    }
                    else
                    {
                        java.util.List addStatements = new ArrayList();

                        Resource claim =
ontmodel.getResource("http://www.cs.rdg.ac.uk/seequel#Claim");
                        Property subject =
ontmodel.getProperty("http://www.cs.rdg.ac.uk/seequel#subject");
                        Property predicate =
ontmodel.getProperty("http://www.cs.rdg.ac.uk/seequel#predicate");
                        Property object =
ontmodel.getProperty("http://www.cs.rdg.ac.uk/seequel#object");
                        Resource statement =
ontmodel.getResource("http://www.w3.org/1999/02/22-rdf-syntax-ns#statement");
```

```
                    Property a =
ontmodel.getProperty("http://www.w3.org/1999/02/22-rdf-syntax-ns#type");

                    Resource reifiedResource =
getUniqueResource("CLAIM");


        addStatements.add(ontmodel.createStatement(reifiedResource,a,claim));

        addStatements.add(ontmodel.createStatement(reifiedResource,subject,subj));

        addStatements.add(ontmodel.createStatement(reifiedResource,predicate,pred))
;

        addStatements.add(ontmodel.createStatement(reifiedResource,object,obj));

        addStatements.add(ontmodel.createStatement(reifiedResource,(Property)say.a
s(Property.class),what));
                    ontmodel.add(addStatements);

                    res = getURI(reifiedResource.getURI());
                    ONTresult.add(res);
                    return "ONTSTORE " + getAIMLURI(res);
                }
```

```
        }catch(Exception e)
        {
                throw new ParseException("Type mismatch in ONTCLAIM");
        }
}
|
<ONTUR> <LPAREN> t=<STRING> <RPAREN>
{
        Resource r = getUniqueResource(getString(t));
        res = getURI(r.getURI());
        ONTresult.add(res);
        return "ONTUNIQUEID " + getAIMLURI(res);
}
|
<ONTISR> <LPAREN> t=<URI> <RPAREN>
{
        String URI = getPureURIfromAIML(t.image);
        Resource r = ontmodel.createResource(URI);

        URI = getURI(URI);
        ONTresult.add(URI);

        if (ontmodel.containsResource(r))
```

```
                return "ONTISRESOURCE " + getAIMLURI(URI);
        else
                return "ONTISRESOURCE " + O_UNKNOWN ;
}
|
<ONTGRAMMA> <LPAREN> strg=<STRING> <RPAREN>
{
        String string = getString(strg);
        boolean b = gramma.isStringValid(string);


        if(b)
        {
                ONTresult.add(string);
                return "ONTGRAMMA " + string;
        }
        else
        {
                ONTresult.add("\"" + O_UNKNOWN +"\"");
                return "ONTGRAMMA " + "\"" + O_UNKNOWN + "\"";
        }
}
|
```

```
	<ONTREGEX> <LPAREN> strg=<STRING> <EQ>
regMatch=RefStringURI() <RPAREN>
	{
		String string = getString(strg);
		Pattern p = Pattern.compile(string);
		Matcher m = p.matcher(regMatch);
		boolean b = m.matches();

		if(b)
			return "ONTREGEX TRUE";
		else
			return "ONTREGEX FALSE";
	}
}

Resource claimRefURI() :
{
	Token t;
}
{
	t=<URI>
	{
		String URI = getPureURIfromAIML(t.image);
```

```
                return ontmodel.createResource(URI);
        }
        |
        t=<REFNUMBER>
        {
                int i = getNumber(t);
                try
                {
                        return
ontmodel.createResource(getPureURI((String)ONTresult.get(i-1)));
                } catch(IndexOutOfBoundsException ie)
                {
                        throw new ParseException("Reference out of bounds. #REF < "
+ ONTresult.size() + " expected.");
                }
        }
}

RDFNode claimRefStringURI() :
{
        Token t;
        Token langToken = null;
        Token typeToken = null;
```

```
}
{
        t=<STRING> [ "@" langToken=<STRING> | "^^" typeToken=<URI> ]
        {
                String rdfString = getString(t);


                try
                {
                        if(langToken!=null)
                        {
                                return
ontmodel.createLiteral(rdfString,getString(langToken));
                        }


                        if(typeToken!=null)
                                return
ontmodel.createTypedLiteral(rdfString,getPureURIfromAIML(typeToken.image));




                        return ontmodel.createLiteral(rdfString);
                }catch(Exception e)
                {
```

```
                    throw new ParseException("Resource not properly defined.");
            }
        }
        |
        t=<URI>
        {
            String URI = getPureURIfromAIML(t.image);
            return ontmodel.createResource(URI);
        }
        |
        t=<REFNUMBER>
        {
            int i = getNumber(t);
            try
            {
                return
ontmodel.createResource(getPureURI((String)ONTresult.get(i-1)));
            } catch(IndexOutOfBoundsException ie)
            {
                throw new ParseException("Reference out of bounds. #REF < "
+ ONTresult.size() + " expected.");
            }
        }
```

```
}

String Command() :
{
        Token t;
}
{
        t=<ONTCLS>
        {
                this.investigateType = Types.CLASS;
                return t.image;
        }
        |
        t=<ONTIND>
        {
                this.investigateType = Types.INDIVIDUAL;
                return t.image;
        }
        |
        t=<ONTPRO>
        {
                this.investigateType = Types.PROPERTY;
                return t.image;
```

```
        }
}



//Return Collection of possibles using logic between predicates
Set  PredicateStatement() :
{
        Set pos = null;
}
{
        <LPAREN> pos=PredicateList() <RPAREN>
        {
                return pos;
        }
}


Set  PredicateList() :
{
        Set pos = null;
}
{
        pos=PredicateXOR()
```

```
        {
                return pos;
        }
}

Set  PredicateXOR() :
{
        Set pos1 = null;
        Set pos2 = null;
        Set mpos = null;
}
{
        pos1=PredicateOR()
        ( <SC_XOR> pos2=PredicateOR()
        {
                if(mpos==null) mpos=pos1;

                HashSet dummy = new HashSet(mpos);
                dummy.retainAll(pos2);

                mpos.addAll(pos2);
                mpos.removeAll(dummy);
        }
```

```
        )*



        {
                if(mpos==null) return pos1;
                return mpos;
        }
}


Set  PredicateOR() :
{
        Set pos1 = null;
        Set pos2 = null;
        Set mpos = null;
}
{
        pos1=PredicateAND()
        ( <SC_OR> pos2=PredicateAND()
        {
                if(mpos==null) mpos=pos1;


                mpos.addAll(pos2);
        }
```

```
        )*



        {
                if(mpos==null) return pos1;
                return mpos;
        }
}


Set  PredicateAND() :
{
        Set pos1 = null;
        Set pos2 = null;
        Set mpos = null;
}
{
        pos1=PredicateUNARY()
        ( <SC_AND> pos2=PredicateUNARY()
        {
                if(mpos==null) mpos=pos1;


                mpos.retainAll(pos2);
        }
```

```
        )*



        {
                if(mpos==null) return pos1;
                return mpos;
        }
}


Set  PredicateUNARY() :
{
        Set pos;
}
{
        pos=Predicate()
        {
                return pos;
        }
        |
        <LPAREN> pos=PredicateList() <RPAREN>
        {
                return pos;
        }
```

```
}

//Return Collection of possibles
Set Predicate() :
{
        Set pos=null;

        String propName=null;
        String propValue=null;
        String indSlotName=null;
        String indValue=null;

        Token t=null;
        Token cType;
        Token indType=null;
}
{
        <PROPERTY> [ propName=ColonStringURI() ] cType=CheckType()
propValue=RefStringURI() [ t=<GETIND> [ indSlotName=ColonStringURI() ]
indType=CheckType() indValue=RefStringURI() ]
        {
                pos=getPROPERTYs(propName,cType,propValue,t,indSlotName,
indType, indValue);
```

```
                return pos;
        }
        |
        <CLS> [ propName=ColonStringURI() ] cType=ClsCheckType()
propValue=RefStringURI()
        {
                pos = getCLSes(propName,cType,propValue);
                return pos;
        }
        |
        <IND> [ propName=ColonStringURI() ] cType=CheckType()
propValue=RefStringURI()
        {
                pos=getINDes(propName, cType,propValue);
                return pos;
        }
        |
        <REST> cType=CheckType() propValue=RefStringURI()
        {
                pos=getRESTs(cType,propValue,t);
                return pos;
        }
}
```

```
Token ClsCheckType() :
{
        Token t;
}
{
        t=<PLUSEQ>
        {
                return t;
        }
        |
        t=CheckType()
        {
                return t;
        }
}

Token CheckType() :
{
        Token t;
}
{
        t=<EQ>
```

```
        {
                return t;
        }
        |
        t=<NQ>
        {
                return t;
        }
}

String RefStringURI() :
{
        Token t;
        Token langToken = null;
        Token typeToken = null;
}
{
        t=<STRING> [ "@" langToken=<STRING> | "^^" typeToken=<URI> ]
        {
                String retString = "\"" + getString(t) + "\"";

                if(langToken!=null)
                        return retString + "@" + getString(langToken);
```

```
                if(typeToken!=null)
                        return retString + "^^" +
getURI(getPureURIfromAIML(typeToken.image));


                return retString;
        }
        |
        t=<URI>
        {
                return getURI(getPureURIfromAIML(t.image));
        }
        |
        t=<REFNUMBER> [ "@" langToken=<STRING> | "^^" typeToken=<URI> ]
        {
                int i = getNumber(t);
                try
                {
                        String retString = (String)ONTresult.get(i-1);
                        if(langToken!=null)
                        {
                                return retString + "@" + getString(langToken);
                        }
```

```
                    if(typeToken!=null)
                    {
                        return retString + "^^" +
getURI(getPureURIfromAIML(typeToken.image));
                    }

                    return retString;
                } catch(IndexOutOfBoundsException ie)
                {
                    throw new ParseException("Reference out of bounds. #REF < "
+ ONTresult.size() + " expected.");
                }
        }
}

String ColonStringURI() :
{
        String str;
}
{
        ":" str=RefStringURI()
        {
```

```
                return str;
        }
}

String ColonString() :
{
        Token t;
}
{
        ":" t=<STRING>
        {
                return getString(t);
        }
}


void ThatReturnPart() :
{
        Token t;
}
{
        <THAT> <ASSIGN> t=<STRING>
        {
```

```
                this.thatReturn = getString(t);
        }
}
```

Appendix C: E*NET ontologies

rcd.owl

```
<?xml version="1.0"?>

<rdf:RDF

   xmlns="http://www.cs.reading.ac.uk/traceSIMPLE#"

   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.cs.reading.ac.uk/traceSIMPLE">

  <owl:Ontology rdf:about=""/>

  <owl:Class rdf:ID="VSRCM">

    <rdfs:label xml:lang="en">Competency Profile</rdfs:label>

    <rdfs:subClassOf>

      <owl:Class rdf:ID="competency"/>

    </rdfs:subClassOf>

    <rdfs:subClassOf>

      <owl:Restriction>

        <owl:onProperty>

          <owl:FunctionalProperty rdf:ID="graph"/>

        </owl:onProperty>

        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
```

```
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment xml:lang="en">Very Simple Reusable Competency
Map</rdfs:comment>
 </owl:Class>
 <owl:Class rdf:ID="graph_type">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="defaultEntryNode"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="types"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
```

```
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="entryNodes"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RCD">
  <rdfs:label xml:lang="en">Single Competency</rdfs:label>
  <rdfs:comment xml:lang="en">Reusable Competency Definition</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#competency"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="proficiency_score">
  <rdfs:subClassOf rdf:resource="#types"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >A score of proficiency</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#competency">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
```

```
            <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >1</owl:minCardinality>
            <owl:onProperty>
              <owl:DatatypeProperty rdf:ID="title"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:DatatypeProperty rdf:about="#title"/>
            </owl:onProperty>
            <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >20</owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >20</owl:maxCardinality>
        <owl:onProperty>
```

```
        <owl:DatatypeProperty rdf:ID="description"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:label xml:lang="en">Competency</rdfs:label>
  <rdfs:comment xml:lang="en">Common competency descriptor</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="node_if_type">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="node_type"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="statement_type">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="statement_name"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="statement_id"/>
    </owl:onProperty>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#types"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >20</owl:maxCardinality>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="statement_text"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="statement_token"/>
```

```
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="definition_type">
  <rdfs:subClassOf rdf:resource="#types"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="statements"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >10</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#node_type">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
```

```
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasCompetency"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="proficienceRequired"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#types"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="proficienceDesired"/>
    </owl:onProperty>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>
  </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="dataRequired"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="metadata_type">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >RCD &amp; SRCM prefix removed to cater for re-use of class
In SRCM draft specification schema information is not part of meta-data, whereas it is
in RCD specification. Schema information has been added to SRCM for
consistency</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#types"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
        <owl:onProperty>
```

```
        <owl:FunctionalProperty rdf:ID="schema_version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="additional_metadata"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >10</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="node_weight_type">
  <rdfs:subClassOf rdf:resource="#node_type"/>
</owl:Class>
<owl:Class rdf:ID="node_noweight_type">
  <rdfs:subClassOf rdf:resource="#node_type"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#additional_metadata">
  <rdfs:domain rdf:resource="#metadata_type"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >additional metadata</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#proficienceDesired">
    <rdfs:domain rdf:resource="#node_type"/>
    <rdfs:range rdf:resource="#proficiency_score"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A desired proficiency score.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="children_plain">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:ID="children"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#node_noweight_type"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="any">
    <rdfs:subPropertyOf rdf:resource="#children_plain"/>
    <rdfs:domain rdf:resource="#node_noweight_type"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >one of the children assigned to this property must achieve it's proficiency
scores</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="if">
    <rdfs:domain rdf:resource="#node_noweight_type"/>
```

```
  <rdfs:range rdf:resource="#node_if_type"/>
  <rdfs:subPropertyOf rdf:resource="#children_plain"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="metadata">
  <rdfs:domain rdf:resource="#competency"/>
  <rdfs:range rdf:resource="#metadata_type"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >consists of any embedded metadata about the RCD</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#entryNodes">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >possible entry nodes (Nodes without parents)</rdfs:comment>
  <rdfs:range rdf:resource="#node_type"/>
  <rdfs:domain rdf:resource="#graph_type"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="if_false">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="children_if"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#node_if_type"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#children">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

>Specifies children of the node. Subproperties of this property specify logic relationships between children.</rdfs:comment>
<rdfs:range rdf:resource="#node_type"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="all">
<rdfs:domain rdf:resource="#node_noweight_type"/>
<rdfs:subPropertyOf rdf:resource="#children_plain"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>all of the children assigned to this property must achieve their proficiency scores</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="fraction">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>A fraction of the children assigned to this property must achieve their proficiency scores. The fraction is specified in the weight of the node.</rdfs:comment>
<rdfs:domain rdf:resource="#node_weight_type"/>
<rdfs:subPropertyOf>
<owl:ObjectProperty rdf:ID="children_with_weight"/>
</rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="units">
<rdfs:subPropertyOf>
<owl:ObjectProperty rdf:about="#children_with_weight"/>

```
    </rdfs:subPropertyOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A number of the children assigned to this property must achieve their proficiency
scores. The number is specified in the weight of the node.</rdfs:comment>
    <rdfs:domain rdf:resource="#node_weight_type"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="if_true">
    <rdfs:domain rdf:resource="#node_if_type"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#children_if"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#statements">
    <rdfs:domain rdf:resource="#definition_type"/>
    <rdfs:range rdf:resource="#statement_type"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >defined chararcteristics of the competency</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasCompetency">
    <rdfs:domain rdf:resource="#node_type"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
```

```
          <owl:Class rdf:about="#competency"/>
          <owl:Class rdf:about="#node_type"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#statement_token">
    <rdfs:domain rdf:resource="#statement_type"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Vocabulary token, if controlled terms(vocabularies) are used instead of
statements</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="mean">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A mean score of the children assigned to this property must achieve the
proficiency score of the node. The mean is specified in the weight of the
node.</rdfs:comment>
    <rdfs:domain rdf:resource="#node_weight_type"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#children_with_weight"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#defaultEntryNode">
```

```
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >default node for entry</rdfs:comment>
    <rdfs:range rdf:resource="#node_type"/>
    <rdfs:domain rdf:resource="#graph_type"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#children_if">
    <rdfs:range rdf:resource="#node_if_type"/>
    <rdfs:subPropertyOf rdf:resource="#children"/>
    <rdfs:domain rdf:resource="#node_if_type"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#proficienceRequired">
    <rdfs:domain rdf:resource="#node_type"/>
    <rdfs:range rdf:resource="#proficiency_score"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A required proficiency score for the node to be satisfied.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasProficiencyScore">
    <rdfs:domain rdf:resource="#node_type"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A proficiency that is held by the described "unit", such as person, group, or
company.</rdfs:comment>
    <rdfs:range rdf:resource="#proficiency_score"/>
  </owl:ObjectProperty>
```

```
  <owl:ObjectProperty rdf:ID="definition">
    <rdfs:range rdf:resource="#definition_type"/>
    <rdfs:domain rdf:resource="#RCD"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A more complete description of the RCD using statements or another
model.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#statement_id">
    <rdfs:domain rdf:resource="#statement_type"/>
    <rdfs:range rdf:resource="#statement_type"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Uniquely identifier of statement.
Is only needed when using anonymous identification, as the URI of the individual
competency normally can be used.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#children_with_weight">
    <rdfs:domain rdf:resource="#node_weight_type"/>
    <rdfs:subPropertyOf rdf:resource="#children"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Specifies children of the node where weights are needed. Subproperties of this
property specify logic relationships between children.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="weight">
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Only used when the node is a child in a rollup operation where a weight is used.
I.e. if the rollup methode is mean.
Value between 0 and 1.</rdfs:comment>
    <rdfs:domain rdf:resource="#node_weight_type"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#statement_text">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Unstructures, textual description of those aspects of the RCD reffered to by
statement name element</rdfs:comment>
    <rdfs:domain rdf:resource="#statement_type"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#title">
    <rdfs:domain rdf:resource="#competency"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Text label of competency</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#description">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Human readable description of competency</rdfs:comment>
```

```
    <rdfs:domain rdf:resource="#competency"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#dataRequired">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >default = true
Indicates if the child is used in rollup operations</rdfs:comment>
    <rdfs:domain rdf:resource="#node_type"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#statement_name">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >name of statement</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#statement_type"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:about="#graph">
    <rdfs:range rdf:resource="#graph_type"/>
    <rdfs:domain rdf:resource="#VSRCM"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#schema_version">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
```

```
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >version of the rcd schema used</rdfs:comment>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#normalizedString"/>
    <rdfs:domain rdf:resource="#metadata_type"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="schema">
    <rdfs:domain rdf:resource="#metadata_type"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >label for the schema that defines and controls the RCD instance</rdfs:comment>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#normalizedString"/>
  </owl:FunctionalProperty>
  <owl:InverseFunctionalProperty rdf:ID="identifier">
    <rdfs:domain rdf:resource="#competency"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Globally unique identifier of competency.
Is only needed when using anonymous identification, as the URI of the individual
competency normally can be used.</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#competency"/>
```

```
  </owl:InverseFunctionalProperty>

</rdf:RDF>


<!-- Created with Protege (with OWL Plugin 3.2, Build 355)

http://protege.stanford.edu -->
```

competencyUpper.owl


```
<?xml version="1.0"?>

<rdf:RDF

    xmlns:trace="http://www.cs.reading.ac.uk/traceSIMPLE#"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

    xmlns="http://www.cs.reading.ac.uk/competencyUpper#"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

    xmlns:owl="http://www.w3.org/2002/07/owl#"

    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"

    xmlns:dc="http://purl.org/dc/elements/1.1/"

  xml:base="http://www.cs.reading.ac.uk/competencyUpper">

  <owl:Ontology rdf:about="">

    <owl:imports rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE"/>

  </owl:Ontology>

  <owl:Class rdf:ID="Other">

    <rdfs:label xml:lang="en">Other</rdfs:label>

    <rdfs:comment xml:lang="en">Other</rdfs:comment>

    <rdfs:subClassOf

rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#RCD"/>

  </owl:Class>

  <owl:Class rdf:ID="Mother_tongue">

    <rdfs:subClassOf rdf:resource="#Other"/>
```

```
    <rdfs:label xml:lang="en">Language in mother tongue</rdfs:label>
    <rdfs:comment xml:lang="en">Language in ones mother tongue.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="Skill">
    <rdfs:comment xml:lang="en">Skill is goal-directed, well-organized behavior that
is acquired through practice and performed with economy of effort. (Typology of
knowledge, skills and competences: clarification of the concept and
prototype)</rdfs:comment>
    <rdfs:label xml:lang="en">Skill</rdfs:label>
    <rdfs:subClassOf
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#RCD"/>
  </owl:Class>
  <owl:Class rdf:ID="Human_capacity">
    <rdfs:comment xml:lang="en">Capacities related to physical attributes, such as
height, eye sight and hearing.</rdfs:comment>
    <rdfs:label xml:lang="en">Human capacity</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Other"/>
  </owl:Class>
  <owl:Class rdf:ID="Context">
    <rdfs:comment xml:lang="en">A context of another competency. I.e. a specific
workplace or application which further defines the competency.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Other"/>
    <rdfs:label xml:lang="en">Context</rdfs:label>
```

```
  </owl:Class>
  <owl:Class rdf:ID="Language">
    <rdfs:subClassOf rdf:resource="#Other"/>
    <rdfs:comment xml:lang="en">Ability to speak a language</rdfs:comment>
    <rdfs:label xml:lang="en">Language</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Knowledge">
    <rdfs:comment xml:lang="en">Knowledge includes underpinning theory and
concepts, as well as tacit knowledge gained as a result of the experience of
performing certain tasks. Understanding refers to more holistic knowledge of
processes and contexts, and may be distinguished as know-why, as opposed know-
that. It also includes general world knowledge (generally measured by vocabulary
tests that are part of many intelligence measurements, and overlapping considerably
with what is defined as crystallized intelligence), and more arbitrary specialized
knowledge. This specialized knowledge is necessary for meeting content specific
demands and solving content-specific tasks. In contrast to general intellectual
abilities, one can consider arbitrary knowledge as a demand-specific competence.
(Typology of knowledge, skills and competences: clarification of the concept and
prototype)</rdfs:comment>
    <rdfs:label xml:lang="en">Knowledge</rdfs:label>
    <rdfs:subClassOf
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#RCD"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="caseOf">
```

```
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >If context A is caseOf knowledge B, then A is a special instantiation of
knowledge B of which all other relations to B also holds true.


Knowledge of Windows Operating System is a special case of Knowledge of
Computer Software.</rdfs:comment>
    <rdfs:range rdf:resource="#Knowledge"/>
    <rdfs:domain rdf:resource="#Other"/>
  </owl:ObjectProperty>
  <owl:TransitiveProperty rdf:ID="hyponymOf">
    <rdfs:label xml:lang="en">more specific than</rdfs:label>
    <rdfs:comment xml:lang="en">If indivdual x is a hyponym of y => x semantically
means the same as y plus more


Crimson is a hyponym of red. red is a hyponym of colour</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
    <rdfs:domain
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:ID="hypernymOf"/>
    </owl:inverseOf>
```

```
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:ID="holonymOf">
    <rdfs:comment xml:lang="en">If indivdual x is a holonym of y => y is part of x


Arm is holonym of hand. Hand is holonym of finger</rdfs:comment>
    <rdfs:label xml:lang="en">has part</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
```

```
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf>
    <owl:TransitiveProperty rdf:ID="meronymOf"/>
  </owl:inverseOf>
 </owl:TransitiveProperty>
 <owl:TransitiveProperty rdf:about="#meronymOf">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment xml:lang="en">If indivdual x is a meronym of y => x is part of y


finger is a meronym of hand. hand is meronym of arm.</rdfs:comment>
```

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:label xml:lang="en">part of</rdfs:label>
    <owl:inverseOf rdf:resource="#holonymOf"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#hypernymOf">
    <rdfs:comment xml:lang="en">If indivdual x is a hypernym of y => y semantically
means the same as x plus more

Colour is a hypernym of Red. Red is a hypernym of crimson.</rdfs:comment>
    <rdfs:domain
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
    <rdfs:range
rdf:resource="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
```

```
    <rdfs:label xml:lang="en">more generic than</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#hyponymOf"/>
  </owl:TransitiveProperty>
  <owl:SymmetricProperty rdf:ID="antonymOf">
    <rdfs:label xml:lang="en">antonym of</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
```

```
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:comment xml:lang="en">If indivdual x is a synonym of y => y semantically
means the opposite of x</rdfs:comment>
  <owl:inverseOf rdf:resource="#antonymOf"/>
 </owl:SymmetricProperty>
 <owl:SymmetricProperty rdf:ID="synonymOf">
  <owl:inverseOf rdf:resource="#synonymOf"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
        <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
```

```
    <rdfs:comment xml:lang="en">If indivdual x is a synonym of y => y semantically
means the same as x  (or closely related to)


</rdfs:comment>
    <rdfs:label xml:lang="en">synonym of</rdfs:label>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#competency"/>
          <rdf:Description
rdf:about="http://www.cs.reading.ac.uk/traceSIMPLE#proficiency_score"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:SymmetricProperty>
  <Skill rdf:ID="Skill_Content">
    <trace:description xml:lang="en">Background structures needed to work with and
acquire more specific skills in a variety of different domains</trace:description>
    <hypernymOf>
      <Skill rdf:ID="Skill_Reading_Comprehension">
        <trace:title xml:lang="en">Reading Comprehension</trace:title>
        <hyponymOf rdf:resource="#Skill_Content"/>
```

```
        <trace:description xml:lang="en">Understanding written sentences and
paragraphs in work related documents.</trace:description>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Science">
        <hyponymOf rdf:resource="#Skill_Content"/>
        <trace:title xml:lang="en">Science</trace:title>
        <trace:description xml:lang="en">Using scientific rules and methods to solve
problems.</trace:description>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Speaking">
        <trace:title xml:lang="en">Speaking</trace:title>
        <hyponymOf rdf:resource="#Skill_Content"/>
        <trace:description xml:lang="en">Talking to others to convey information
effectively.</trace:description>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Mathematics">
```

```
        <trace:description xml:lang="en">Using mathematics to solve
problems.</trace:description>
        <trace:title xml:lang="en">Mathematics</trace:title>
        <hyponymOf rdf:resource="#Skill_Content"/>
      </Skill>
    </hypernymOf>
    <trace:title xml:lang="en">Content</trace:title>
    <hypernymOf>
      <Skill rdf:ID="Skill_Active_Listening">
        <trace:description xml:lang="en">Giving full attention to what other people are
saying, taking time to understand the points being made, asking questions as
appropriate, and not interrupting at inappropriate times.</trace:description>
        <trace:title xml:lang="en">Active Listening</trace:title>
        <hyponymOf rdf:resource="#Skill_Content"/>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Writing">
        <hyponymOf rdf:resource="#Skill_Content"/>
        <trace:description xml:lang="en">Communicating effectively in writing as
appropriate for the needs of the audience.</trace:description>
        <trace:title xml:lang="en">Writing</trace:title>
      </Skill>
```

```
    </hypernymOf>
    <hyponymOf>
      <Skill rdf:ID="Skill_Basic_Skills">
        <hypernymOf rdf:resource="#Skill_Content"/>
        <hypernymOf>
          <Skill rdf:ID="Skill_Process">
            <hypernymOf>
              <Skill rdf:ID="Skill_Critical_Thinking">
                <hyponymOf rdf:resource="#Skill_Process"/>
                <trace:title xml:lang="en">Critical Thinking</trace:title>
                <trace:description xml:lang="en">Using logic and reasoning to identify the
strengths and weaknesses of alternative solutions, conclusions or approaches to
problems.</trace:description>
              </Skill>
            </hypernymOf>
            <hypernymOf>
              <Skill rdf:ID="Skill_Learning_Strategies">
                <trace:description xml:lang="en">Selecting and using training/instructional
methods and procedures appropriate for the situation when learning or teaching new
things.</trace:description>
                <hyponymOf rdf:resource="#Skill_Process"/>
                <trace:title xml:lang="en">Learning Strategies</trace:title>
              </Skill>
```

```
</hypernymOf>
<hypernymOf>
  <Skill rdf:ID="Skill_Monitoring">
    <trace:title xml:lang="en">Monitoring</trace:title>
    <trace:description xml:lang="en">Monitoring/Assessing performance of yourself, other individuals, or organizations to make improvements or take corrective action.</trace:description>
    <hyponymOf rdf:resource="#Skill_Process"/>
  </Skill>
</hypernymOf>
<hypernymOf>
  <Skill rdf:ID="Skill_Active_Learning">
    <trace:title xml:lang="en">Active Learning</trace:title>
    <trace:description xml:lang="en">Understanding the implications of new information for both current and future problem-solving and decision-making.</trace:description>
    <hyponymOf rdf:resource="#Skill_Process"/>
  </Skill>
</hypernymOf>
<trace:description xml:lang="en">Procedures that contribute to the more rapid acquisition of knowledge and skill across a variety of domains</trace:description>
<hyponymOf rdf:resource="#Skill_Basic_Skills"/>
<trace:title xml:lang="en">Process</trace:title>
```

```
          </Skill>
        </hypernymOf>
        <trace:title xml:lang="en">Basic Skills</trace:title>
        <trace:description xml:lang="en">Developed capacities that facilitate learning or
the more rapid acquisition of knowledge </trace:description>
      </Skill>
    </hyponymOf>
  </Skill>
  <Language rdf:ID="French">
    <trace:title xml:lang="en">French</trace:title>
  </Language>
  <Knowledge rdf:ID="Knowledge_Sales_and_Marketing">
    <trace:description xml:lang="en">Knowledge of principles and methods for
showing, promoting, and selling products or services.  This includes marketing
strategy and tactics, product demonstration, sales techniques, and sales control
systems.</trace:description>
    <trace:title xml:lang="en">Sales and Marketing</trace:title>
    <hyponymOf>
      <Knowledge rdf:ID="Knowledge_Business_and_Management">
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_Administration_and_Management">
            <hyponymOf rdf:resource="#Knowledge_Business_and_Management"/>
```

<trace:description xml:lang="en">Knowledge of business and management principles involved in strategic planning, resource allocation, human resources modeling, leadership technique, production methods, and coordination of people and resources.</trace:description>

<trace:description xml:lang="da">Viden om forretnings of ledelses principper indenfor strategisk planlaegning, resource allokation, HR modellering, lederskab teknikker, produktions metoder og allocation af personale of andre resourser.</trace:description>

<trace:title xml:lang="da">Administration og ledelse</trace:title>

<trace:title xml:lang="en">Administration and Management</trace:title>

</Knowledge>

</hypernymOf>

<hypernymOf>

<Knowledge rdf:ID="Knowledge_Economics_and_Accounting">

<hyponymOf rdf:resource="#Knowledge_Business_and_Management"/>

<trace:title xml:lang="en">Economics and Accounting</trace:title>

<trace:description xml:lang="en">Knowledge of economic and accounting principles and practices, the financial markets, banking and the analysis and reporting of financial data.</trace:description>

</Knowledge>

</hypernymOf>

<hypernymOf>

<Knowledge rdf:ID="Knowledge_Customer_and_Personal_Service">

<trace:title xml:lang="en">Customer and Personal Service</trace:title>

```
        <hyponymOf rdf:resource="#Knowledge_Business_and_Management"/>
        <trace:description xml:lang="en">Knowledge of principles and processes for providing customer and personal services.  This includes customer needs assessment, meeting quality standards for services, and evaluation of customer satisfaction.</trace:description>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Personnel_and_Human_Resources">
        <trace:title xml:lang="en">Personnel and Human Resources</trace:title>
        <hyponymOf rdf:resource="#Knowledge_Business_and_Management"/>
        <trace:description xml:lang="en">Knowledge of policies and practices involved in personnel/human resources functions. This includes recruitment, selection, training, and promotion regulations and procedures; compensation and benefits packages; labor relations and negotiation strategies; and personnel information system.</trace:description>
      </Knowledge>
    </hypernymOf>
    <hypernymOf rdf:resource="#Knowledge_Sales_and_Marketing"/>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Clerical">
        <hyponymOf rdf:resource="#Knowledge_Business_and_Management"/>
        <trace:title xml:lang="en">Clerical knowledge</trace:title>
```

```
        <trace:description xml:lang="en">Knowledge of administrative and clerical procedures and systems such as word processing, managing files and records, stenography and transcription, designing forms, and other office procedures and terminology.</trace:description>
        </Knowledge>
      </hypernymOf>
      <trace:description xml:lang="en">Knowledge of principles and facts related to business administration and accounting, human and material resource management in organizations, sales and marketing, economics, and office information and organizing systems</trace:description>
      <trace:title xml:lang="en">Business and Management</trace:title>
    </Knowledge>
  </hyponymOf>
</Knowledge>
<Knowledge rdf:ID="Knowledge_Mathematics_and_Science">
  <hypernymOf>
    <Knowledge rdf:ID="Knowledge_Mathematics">
      <trace:description xml:lang="en">Knowledge of arithmetic, algebra, geometry, calculus, statistics, and their applications.</trace:description>
      <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
      <trace:title xml:lang="en">Mathematics</trace:title>
    </Knowledge>
  </hypernymOf>
  <hypernymOf>
```

```
      <Knowledge rdf:ID="Knowledge_Biology">
        <trace:title xml:lang="en">Biology</trace:title>
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
        <trace:description xml:lang="en">Knowledge of plant and animal organisms,
their tissues, cells, functions, interdependencies, and interactions with each other and
the environment.</trace:description>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Chemistry">
        <trace:title xml:lang="en">Chemistry</trace:title>
        <trace:description xml:lang="en">Knowledge of the chemical composition,
structure, and properties of substances and of the chemical processes and
transformations that they undergo.  This includes uses of chemicals and their
interactions, danger signs, production techniques, and disposal
methods.</trace:description>
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Sociology_and_Anthropology">
        <trace:description xml:lang="en">Knowledge of group behavior and dynamics,
societal trends and influences, human migrations, ethnicity, cultures and their history
and origins.</trace:description>
```

```
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
        <trace:title xml:lang="en">Knowledge of Sociology and
Anthropology</trace:title>
      </Knowledge>
    </hypernymOf>
    <trace:title xml:lang="en">Mathematics and Science</trace:title>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Geography">
        <trace:description xml:lang="en">Knowledge of principles and methods for
describing the features of land, sea, and air masses, including their physical
characteristics, locations, interrelationships, and distribution of plant, animal, and
human life.</trace:description>
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
        <trace:title xml:lang="en">Knowledge of Geography</trace:title>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Psychology">
        <trace:title xml:lang="en">Knowledge of Psychology</trace:title>
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
        <trace:description xml:lang="en">Knowledge of human behavior and
performance; individual differences in ability, personality, and interests; learning and
motivation; psychological research methods; and the assessment and treatment of
behavioral and affective disorders.</trace:description>
```

```
      </Knowledge>
    </hypernymOf>
    <trace:description xml:lang="en">Knowledge of the history, theories, methods,
and applications of the physical, biological, social, mathematical, and
geography</trace:description>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Physics">
        <trace:description xml:lang="en">Knowledge and prediction of physical
principles, laws, their interrelationships, and applications to understanding fluid,
material, and atmospheric dynamics, and mechanical, electrical, atomic and sub-
atomic structures and processes.</trace:description>
        <hyponymOf rdf:resource="#Knowledge_Mathematics_and_Science"/>
        <trace:title xml:lang="en">Physics</trace:title>
      </Knowledge>
    </hypernymOf>
  </Knowledge>
  <Skill rdf:ID="Skill_Testing">
    <hyponymOf>
      <Skill rdf:ID="Skill_Technical_Skills">
        <hypernymOf>
          <Skill rdf:ID="Skill_Operation_and_Control">
            <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
```

```
      <trace:description xml:lang="en">Controlling operations of equipment or
systems.</trace:description>
      <trace:title xml:lang="en">Operation and Control</trace:title>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Troubleshooting">
      <trace:description xml:lang="en">Determining causes of operating errors and
deciding what to do about it.</trace:description>
      <trace:title xml:lang="en">Troubleshooting</trace:title>
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Equipment_Selection">
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
      <trace:description xml:lang="en">Determining the kind of tools and
equipment needed to do a job.</trace:description>
      <trace:title xml:lang="en">Equipment Selection</trace:title>
    </Skill>
  </hypernymOf>
  <trace:title xml:lang="en">Technical Skills</trace:title>
  <hypernymOf>
```

```
<Skill rdf:ID="Skill_Programming">
<trace:description xml:lang="en">Writing computer programs for various purposes.</trace:description>
<trace:title xml:lang="en">Programming</trace:title>
<hyponymOf rdf:resource="#Skill_Technical_Skills"/>
</Skill>
</hypernymOf>
<hypernymOf>
<Skill rdf:ID="Skill_Equipment_Maintenance">
<trace:title xml:lang="en">Equipment Maintenance</trace:title>
<hyponymOf rdf:resource="#Skill_Technical_Skills"/>
<trace:description xml:lang="en">Performing routine maintenance on equipment and determining when and what kind of maintenance is needed.</trace:description>
</Skill>
</hypernymOf>
<hypernymOf rdf:resource="#Skill_Testing"/>
<hypernymOf>
<Skill rdf:ID="Skill_Installation">
<trace:description xml:lang="en">Installing equipment, machines, wiring, or programs to meet specifications.</trace:description>
<trace:title xml:lang="en">Installation</trace:title>
<hyponymOf rdf:resource="#Skill_Technical_Skills"/>
```

```
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Product_Inspection">
      <trace:description xml:lang="en">Inspecting and evaluating the quality of
products.</trace:description>
      <trace:title xml:lang="en">Product inspection</trace:title>
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Repairing">
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
      <trace:description xml:lang="en">Repairing machines or systems using the
needed tools.</trace:description>
      <trace:title xml:lang="en">Repairing</trace:title>
    </Skill>
  </hypernymOf>
  <trace:description xml:lang="en">Developed capacities used to design, set-up,
operate, and correct malfunctions involving application of machines or technological
systems</trace:description>
  <hypernymOf>
    <Skill rdf:ID="Skill_Quality_Control_Analysis">
```

```
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
      <trace:description xml:lang="en">Conducting tests and inspections of products, services, or processes to evaluate quality or performance.</trace:description>
      <trace:title xml:lang="en">Quality Control Analysis</trace:title>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Operation_Monitoring">
      <trace:title xml:lang="en">Operation Monitoring</trace:title>
      <trace:description xml:lang="en">Watching gauges, dials, or other indicators to make sure a machine is working properly.</trace:description>
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Technology_Design">
      <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
      <trace:description xml:lang="en">Generating or adapting equipment and technology to serve user needs.</trace:description>
      <trace:title xml:lang="en">Technology Design</trace:title>
    </Skill>
  </hypernymOf>
```

```
<hypernymOf>
  <Skill rdf:ID="Skill_Operations_Analysis">
    <trace:title xml:lang="en">Operations Analysis</trace:title>
    <trace:description xml:lang="en">Analyzing needs and product requirements
to create a design.</trace:description>
    <hyponymOf rdf:resource="#Skill_Technical_Skills"/>
  </Skill>
</hypernymOf>
<hyponymOf>
  <Skill rdf:ID="Skill_Cross-Functional_Skills">
    <trace:description xml:lang="en">Developed capacities that facilitate
performance of activities that occur across jobs</trace:description>
    <hypernymOf>
      <Skill rdf:ID="Skill_Resource_Management_Skills">
        <hyponymOf rdf:resource="#Skill_Cross-Functional_Skills"/>
        <trace:title xml:lang="en">Resource Management Skills</trace:title>
        <hypernymOf>
          <Skill rdf:ID="Skill_Management_of_Personnel_Resources">
            <trace:title xml:lang="en">Management of Personnel
Resources</trace:title>
            <trace:description xml:lang="en">Motivating, developing, and directing
people as they work, identifying the best people for the job.</trace:description>
            <hyponymOf rdf:resource="#Skill_Resource_Management_Skills"/>
```

```
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Management_of_Material_Resources">
      <hyponymOf rdf:resource="#Skill_Resource_Management_Skills"/>
      <trace:description xml:lang="en">Obtaining and seeing to the
appropriate use of equipment, facilities, and materials needed to do certain
work.</trace:description>
      <trace:title xml:lang="en">Management of Material
Resources</trace:title>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Time_Management">
      <hyponymOf rdf:resource="#Skill_Resource_Management_Skills"/>
      <trace:description xml:lang="en">Managing one's own time and the time
of others.</trace:description>
      <trace:title xml:lang="en">Time Management</trace:title>
    </Skill>
  </hypernymOf>
  <hypernymOf>
    <Skill rdf:ID="Skill_Management_of_Financial_Resources">
```

```
<trace:description xml:lang="en">Determining how money will be spent to get the work done, and accounting for these expenditures.</trace:description>
<trace:title xml:lang="en">Management of Financial Resources</trace:title>
<hyponymOf rdf:resource="#Skill_Resource_Management_Skills"/>
</Skill>
</hypernymOf>
<trace:description xml:lang="en">Developed capacities used to allocate resources efficiently</trace:description>
</Skill>
</hypernymOf>
<hypernymOf>
<Skill rdf:ID="Skill_Systems_Skills">
<trace:description xml:lang="en">Developed capacities used to understand, monitor, and improve socio-technical systems</trace:description>
<trace:title xml:lang="en">Systems Skills</trace:title>
<hyponymOf rdf:resource="#Skill_Cross-Functional_Skills"/>
<hypernymOf>
<Skill rdf:ID="Skill_Systems_Evaluation">
<trace:title xml:lang="en">Systems Evaluation</trace:title>
<trace:description xml:lang="en">Identifying measures or indicators of system performance and the actions needed to improve or correct performance, relative to the goals of the system.</trace:description>
```

```
        <hyponymOf rdf:resource="#Skill_Systems_Skills"/>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Systems_Analysis">
        <trace:title xml:lang="en">Systems Analysis</trace:title>
        <hyponymOf rdf:resource="#Skill_Systems_Skills"/>
        <trace:description xml:lang="en">Determining how a system should
work and how changes in conditions, operations, and the environment will affect
outcomes.</trace:description>
      </Skill>
    </hypernymOf>
    <hypernymOf>
      <Skill rdf:ID="Skill_Judgment_and_Decision_Making">
        <trace:title xml:lang="en">Judgment and Decision Making</trace:title>
        <trace:description xml:lang="en">Considering the relative costs and
benefits of potential actions to choose the most appropriate one.</trace:description>
        <hyponymOf rdf:resource="#Skill_Systems_Skills"/>
      </Skill>
    </hypernymOf>
  </Skill>
</hypernymOf>
<hypernymOf>
```

```
<Skill rdf:ID="Skill_Complex_Problem_Solving">
  <trace:title xml:lang="en">Complex problem solving</trace:title>
  <hyponymOf rdf:resource="#Skill_Cross-Functional_Skills"/>
  <trace:description xml:lang="en">Identifying complex problems and
reviewing related information to develop and evaluate options and implement
solutions.</trace:description>
</Skill>
</hypernymOf>
<hypernymOf>
<Skill rdf:ID="Skill_Social_Skills">
  <hypernymOf>
    <Skill rdf:ID="Skill_Persuasion">
      <hyponymOf rdf:resource="#Skill_Social_Skills"/>
      <trace:title xml:lang="en">Persuasion</trace:title>
      <trace:description xml:lang="en">Persuading others to change their
minds or behavior.</trace:description>
    </Skill>
  </hypernymOf>
  <trace:description xml:lang="en">Developed capacities used to work with
people to achieve goals</trace:description>
  <hypernymOf>
    <Skill rdf:ID="Skill_Negotiation">
      <hyponymOf rdf:resource="#Skill_Social_Skills"/>
```

```
<trace:description xml:lang="en">Bringing others together and trying to reconcile differences.</trace:description>
<trace:title xml:lang="en">Negotiation</trace:title>
</Skill>
</hypernymOf>
<hypernymOf>
<Skill rdf:ID="Skill_Coordination">
<trace:description xml:lang="en">Adjusting actions in relation to others' actions.</trace:description>
<trace:title xml:lang="en">Coordination</trace:title>
<hyponymOf rdf:resource="#Skill_Social_Skills"/>
</Skill>
</hypernymOf>
<hypernymOf>
<Skill rdf:ID="Skill_Service_Orientation">
<hyponymOf rdf:resource="#Skill_Social_Skills"/>
<trace:title xml:lang="en">Service Orientation</trace:title>
<trace:description xml:lang="en">Actively looking for ways to help people.</trace:description>
</Skill>
</hypernymOf>
<trace:title xml:lang="en">Social Skills</trace:title>
<hypernymOf>
```

```
<Skill rdf:ID="Skill_Instructing">
  <trace:title xml:lang="en">Instructing</trace:title>
  <trace:description xml:lang="en">Teaching others how to do
something.</trace:description>
  <hyponymOf rdf:resource="#Skill_Social_Skills"/>
</Skill>
</hypernymOf>
<hyponymOf rdf:resource="#Skill_Cross-Functional_Skills"/>
<hypernymOf>
  <Skill rdf:ID="Skill_Social_Perceptiveness">
    <trace:description xml:lang="en">Being aware of others' reactions and
understanding why they react as they do.</trace:description>
    <trace:title xml:lang="en">Social Perceptiveness</trace:title>
    <hyponymOf rdf:resource="#Skill_Social_Skills"/>
  </Skill>
</hypernymOf>
</Skill>
</hypernymOf>
<hypernymOf rdf:resource="#Skill_Technical_Skills"/>
<trace:title xml:lang="en">Cross-Functional Skills</trace:title>
</Skill>
</hyponymOf>
</Skill>
```

```
    </hyponymOf>
    <trace:description xml:lang="en">Conducting tests to determine whether
equipment, software, or procedures are operating as expected.</trace:description>
    <trace:title xml:lang="en">Testing</trace:title>
  </Skill>
  <Mother_tongue rdf:ID="mo_Danish">
    <trace:title xml:lang="en">Danish</trace:title>
  </Mother_tongue>
  <Language rdf:ID="English">
    <trace:title xml:lang="en">English</trace:title>
  </Language>
  <Language rdf:ID="Dutch">
    <trace:title xml:lang="en">Dutch</trace:title>
    <trace:description xml:lang="en">Dutch; Flemish</trace:description>
  </Language>
  <trace:proficiency_score rdf:ID="proficiency_score_5">
    <rdfs:label xml:lang="en">level 5</rdfs:label>
    <rdfs:comment xml:lang="en">Systematic and critical understanding of body of
knowledge and its limits
```


Able to create a research-based diagnosis to problems even with incomplete, conflicting or limited information; Appropriate and context-dependent selection of instruments and tools for analysis

Manage and transform work or study contexts that are complex, unpredictable and require new strategic approaches take responsibility for contributing to professional knowledge and practice and/or for reviewing the strategic performance of teams.</rdfs:comment>

<holonymOf>

<trace:proficiency_score rdf:ID="proficiency_score_4">

<rdfs:comment xml:lang="en">Understanding of theory, concepts and methods some of it at its limits

Able to autonomously analyse abstract or unfamiliar situations based on broad spectrum of methods; Able to analyse contradictory data and to perform guided research

Manage complex technical or professional activities or projects, taking responsibility for decision-making in unpredictable work or study contexts take responsibility for managing professional development of individuals and groups.</rdfs:comment>

<rdfs:label xml:lang="en">level 4</rdfs:label>

<holonymOf>

<trace:proficiency_score rdf:ID="proficiency_score_3">

<rdfs:label xml:lang="en">level 3</rdfs:label>

<meronymOf rdf:resource="#proficiency_score_4"/>

<rdfs:comment xml:lang="en">Recognising innovation and limits of oneâ€™s own knowledge base

Able to analyse wide range of situations / problems by applying general theory; Able to identify missing information and methods to access this information

Exercise self-management within the guidelines of work or study contexts that are usually predictable, but are subject to change supervise the routine work of others, taking some responsibility for the evaluation and improvement of work or study activities.</rdfs:comment>

<holonymOf>

<trace:proficiency_score rdf:ID="proficiency_score_2">

<meronymOf rdf:resource="#proficiency_score_3"/>

<holonymOf>

<trace:proficiency_score rdf:ID="proficiency_score_1">

<rdfs:comment xml:lang="en">Basic understanding in reference to practical contexts

Able to interpret situation and to perform guided analysis based on given classifications or principles

Work or study under supervision with some autonomy.</rdfs:comment>

<meronymOf rdf:resource="#proficiency_score_2"/>

<rdfs:label xml:lang="en">level 1</rdfs:label>

</trace:proficiency_score>

```
            </holonymOf>
            <rdfs:comment xml:lang="en">Showing awareness when applying
concepts, knowledge and terminology in practical contexts


Able to analyse selected information and gather required information


Take responsibility for completion of tasks in work or study adapt own behaviour to
circumstances in solving problems.</rdfs:comment>
            <rdfs:label xml:lang="en">level 2</rdfs:label>
          </trace:proficiency_score>
         </holonymOf>
        </trace:proficiency_score>
       </holonymOf>
       <meronymOf rdf:resource="#proficiency_score_5"/>
     </trace:proficiency_score>
   </holonymOf>
 </trace:proficiency_score>
 <Language rdf:ID="Spanish">
   <trace:description xml:lang="en">Spanish; Castilian</trace:description>
   <trace:title xml:lang="en">Spanish</trace:title>
 </Language>
 <Mother_tongue rdf:ID="mo_Italian">
   <trace:title xml:lang="en">Italian</trace:title>
```

```
  </Mother_tongue>
  <Mother_tongue rdf:ID="mo_Turkish">
    <trace:title xml:lang="en">Turkish</trace:title>
  </Mother_tongue>
  <Language rdf:ID="Romanian">
    <trace:title xml:lang="en">Romanian</trace:title>
  </Language>
  <Mother_tongue rdf:ID="mo_Polish">
    <trace:title xml:lang="en">Polish</trace:title>
  </Mother_tongue>
  <Knowledge rdf:ID="Knowledge_Production_and_Processing">
    <hyponymOf>
      <Knowledge rdf:ID="Knowledge_Manufacturing_and_Production">
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_Food_Production">
            <trace:title xml:lang="en">Food Production</trace:title>
            <hyponymOf rdf:resource="#Knowledge_Manufacturing_and_Production"/>
            <trace:description xml:lang="en">Knowledge of techniques and equipment
for planting, growing, and harvesting food products (both plant and animal) for
consumption, including storage/handling techniques.</trace:description>
          </Knowledge>
        </hypernymOf>
        <hypernymOf rdf:resource="#Knowledge_Production_and_Processing"/>
```

<trace:description xml:lang="en">Knowledge of principles and facts related to the production, processing, storage, and distribution of manufactured and agricultural goods</trace:description>

<trace:title xml:lang="en">Manufacturing and Production</trace:title>

</Knowledge>

</hyponymOf>

<trace:title xml:lang="en">Production and Processing</trace:title>

<trace:description xml:lang="en">Knowledge of raw materials, production processes, quality control, costs, and other techniques for maximizing the effective manufacture and distribution of goods.</trace:description>

</Knowledge>

<Language rdf:ID="Bulgarian">

<trace:title xml:lang="en">Bulgarian</trace:title>

</Language>

<Mother_tongue rdf:ID="mo_Spanish">

<trace:description xml:lang="en">Spanish; Castilian</trace:description>

<trace:title xml:lang="en">Spanish</trace:title>

</Mother_tongue>

<Knowledge rdf:ID="Knowledge_Education_and_Training">

<trace:title xml:lang="en">Education and Training</trace:title>

<trace:description xml:lang="en">Knowledge of principles and methods for curriculum and training design, teaching and instruction for individuals and groups, and the measurement of training effects.</trace:description>

```
  </Knowledge>
  <Mother_tongue rdf:ID="mo_Dutch">
    <trace:description xml:lang="en">Dutch; Flemish</trace:description>
    <trace:title xml:lang="en">Dutch</trace:title>
  </Mother_tongue>
  <Mother_tongue rdf:ID="mo_French">
    <trace:title xml:lang="en">French</trace:title>
  </Mother_tongue>
  <Mother_tongue rdf:ID="mo_Czech">
    <trace:title xml:lang="en">Czech</trace:title>
  </Mother_tongue>
  <Knowledge rdf:ID="Knowledge_Therapy_and_Counseling">
    <trace:description xml:lang="en">Knowledge of principles, methods, and
procedures for diagnosis, treatment, and rehabilitation of physical and mental
dysfunctions, and for career counseling and guidance.</trace:description>
    <trace:title xml:lang="en">Therapy and Counseling</trace:title>
    <hyponymOf>
      <Knowledge rdf:ID="Knowledge_Health_Services">
        <trace:title xml:lang="en">Health Services</trace:title>
        <hypernymOf rdf:resource="#Knowledge_Therapy_and_Counseling"/>
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_Medicine_and_Dentistry">
            <hyponymOf rdf:resource="#Knowledge_Health_Services"/>
```

```
            <trace:title xml:lang="en">Medicine and Dentistry</trace:title>
            <trace:description xml:lang="en">Knowledge of the information and
techniques needed to diagnose and treat human injuries, diseases, and deformities.
This includes symptoms, treatment alternatives, drug properties and interactions, and
preventive health-care measures.</trace:description>
          </Knowledge>
        </hypernymOf>
        <trace:description xml:lang="en">Knowledge of principles and facts regarding
diagnosing, curing, and preventing disease, and improving and preserving physical
and mental health and well-being</trace:description>
      </Knowledge>
    </hyponymOf>
  </Knowledge>
  <Language rdf:ID="Russian">
    <trace:title xml:lang="en">Russian</trace:title>
  </Language>
  <Mother_tongue rdf:ID="mo_Russian">
    <trace:title xml:lang="en">Russian</trace:title>
  </Mother_tongue>
  <Language rdf:ID="Icelandic">
    <trace:title xml:lang="en">Icelandic</trace:title>
  </Language>
  <Knowledge rdf:ID="Knowledge_Public_Safety_and_Security">
```

```
<trace:title xml:lang="en">Public Safety and Security</trace:title>
<trace:description xml:lang="en">Knowledge of relevant equipment, policies, procedures, and strategies to promote effective local, state, or national security operations for the protection of people, data, property, and institutions.</trace:description>
<hyponymOf>
  <Knowledge rdf:ID="Knowledge_Law_and_Public_Safety">
    <trace:description xml:lang="en">Knowledge of regulations and methods for maintaining people and property free from danger, injury, or damage; the rules of public conduct established and enforced by legislation, and the political process establishing such rules.</trace:description>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Law_and_Government">
        <trace:description xml:lang="en">Knowledge of laws, legal codes, court procedures, precedents, government regulations, executive orders, agency rules, and the democratic political process.</trace:description>
        <hyponymOf rdf:resource="#Knowledge_Law_and_Public_Safety"/>
        <trace:title xml:lang="en">Law and Government</trace:title>
      </Knowledge>
    </hypernymOf>
    <hypernymOf rdf:resource="#Knowledge_Public_Safety_and_Security"/>
    <trace:title xml:lang="en">Law and Public Safety</trace:title>
  </Knowledge>
</hyponymOf>
```

```
  </Knowledge>
  <Language rdf:ID="Italian">
    <trace:title xml:lang="en">Italian</trace:title>
  </Language>
  <Language rdf:ID="Greek">
    <trace:title xml:lang="en">Greek</trace:title>
    <trace:description xml:lang="en">Greek, Modern (1453-)</trace:description>
  </Language>
  <Mother_tongue rdf:ID="mo_Norwegian">
    <trace:title xml:lang="en">Norwegian</trace:title>
  </Mother_tongue>
  <Knowledge rdf:ID="Knowledge_Fine_Arts">
    <trace:title xml:lang="en">Fine Arts</trace:title>
    <trace:description xml:lang="en">Knowledge of the theory and techniques
required to compose, produce, and perform works of music, dance, visual arts, drama,
and sculpture.</trace:description>
    <hyponymOf>
      <Knowledge rdf:ID="Knowledge_Arts_and_Humanities">
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_Foreign_Language">
            <trace:description xml:lang="en">Knowledge of the structure and content of a
foreign (not Mother Tongue) language including the meaning and spelling of words,
rules of composition and grammar, and pronunciation.</trace:description>
```

```
      <trace:title xml:lang="en">Foreign Language</trace:title>
      <hyponymOf rdf:resource="#Knowledge_Arts_and_Humanities"/>
    </Knowledge>
  </hypernymOf>
  <hypernymOf>
    <Knowledge rdf:ID="Knowledge_First_Language">
      <trace:description xml:lang="en">Knowledge of the structure and content of the first language (Mother Tongue) including the meaning and spelling of words, rules of composition, and grammar.</trace:description>
      <trace:title xml:lang="en">First (Mother Tongue) Language</trace:title>
      <hyponymOf rdf:resource="#Knowledge_Arts_and_Humanities"/>
    </Knowledge>
  </hypernymOf>
  <hypernymOf>
    <Knowledge rdf:ID="Knowledge_Philosophy_and_Theology">
      <trace:title xml:lang="en">Philosophy and Theology</trace:title>
      <hyponymOf rdf:resource="#Knowledge_Arts_and_Humanities"/>
      <trace:description xml:lang="en">Knowledge of different philosophical systems and religions. This includes their basic principles, values, ethics, ways of thinking, customs, practices, and their impact on human culture.</trace:description>
    </Knowledge>
  </hypernymOf>
  <trace:title xml:lang="en">Arts and Humanities</trace:title>
```

```
        <trace:description xml:lang="en">Knowledge of facts and principles related to
the branches of learning concerned with human thought, language, and the
arts.</trace:description>
        <trace:title xml:lang="da">Kunst og humaniora</trace:title>
        <hypernymOf rdf:resource="#Knowledge_Fine_Arts"/>
        <trace:description xml:lang="da">Viden om fakta og principper relateret til fag
grupper med interesse i humanistisk tankegang, sprog og kunst</trace:description>
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_History_and_Archeology">
            <hyponymOf rdf:resource="#Knowledge_Arts_and_Humanities"/>
            <trace:title xml:lang="en">History and Archeology</trace:title>
            <trace:description xml:lang="en">Knowledge of historical events and their
causes, indicators, and effects on civilizations and cultures.</trace:description>
          </Knowledge>
        </hypernymOf>
      </Knowledge>
    </hyponymOf>
  </Knowledge>
  <Language rdf:ID="Norwegian">
    <trace:title xml:lang="en">Norwegian</trace:title>
  </Language>
  <Language rdf:ID="Polish">
    <trace:title xml:lang="en">Polish</trace:title>
```

```
  </Language>
  <Language rdf:ID="Portuguese">
    <trace:title xml:lang="en">Portuguese</trace:title>
  </Language>
  <Language rdf:ID="Croatian">
    <trace:title xml:lang="en">Croatian</trace:title>
  </Language>
  <Knowledge rdf:ID="Knowledge_Engineering_Technology_ALL">
    <trace:description xml:lang="en">Knowledge of the design, development, and
application of technology for specific purposes.</trace:description>
    <trace:title xml:lang="en">Engineering and Technology</trace:title>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Engineering_and_Technology">
        <hyponymOf rdf:resource="#Knowledge_Engineering_Technology_ALL"/>
        <trace:title xml:lang="en">Engineering and Technology</trace:title>
        <trace:description xml:lang="en">Knowledge of the practical application of
engineering science and technology.  This includes applying principles, techniques,
procedures, and equipment to the design and production of various goods and
services.</trace:description>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Mechanical">
```

```
        <hyponymOf rdf:resource="#Knowledge_Engineering_Technology_ALL"/>
        <trace:description xml:lang="en">Knowledge of machines and tools, including their designs, uses, repair, and maintenance.</trace:description>
        <trace:title xml:lang="en">Mechanical knowledge</trace:title>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Building_and_Construction">
        <hyponymOf rdf:resource="#Knowledge_Engineering_Technology_ALL"/>
        <trace:description xml:lang="en">Knowledge of materials, methods, and the tools involved in the construction or repair of houses, buildings, or other structures such as highways and roads.</trace:description>
        <trace:title xml:lang="en">Building and Construction</trace:title>
      </Knowledge>
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Design">
        <trace:title xml:lang="en">Design</trace:title>
        <trace:description xml:lang="en">Knowledge of design techniques, tools, and principles involved in production of precision technical plans, blueprints, drawings, and models.</trace:description>
        <hyponymOf rdf:resource="#Knowledge_Engineering_Technology_ALL"/>
      </Knowledge>
```

```
    </hypernymOf>
    <hypernymOf>
      <Knowledge rdf:ID="Knowledge_Computers_and_Electronics">
        <hyponymOf rdf:resource="#Knowledge_Engineering_Technology_ALL"/>
        <trace:description xml:lang="en">Knowledge of circuit boards, processors,
chips, electronic equipment, and computer hardware and software, including
applications and programming.</trace:description>
        <trace:title xml:lang="en">Computers and Electronics</trace:title>
      </Knowledge>
    </hypernymOf>
  </Knowledge>
  <Mother_tongue rdf:ID="mo_English">
    <trace:title xml:lang="en">English</trace:title>
  </Mother_tongue>
  <Language rdf:ID="Slovak">
    <trace:title xml:lang="en">Slovak</trace:title>
  </Language>
  <Mother_tongue rdf:ID="mo_Finnish">
    <trace:title xml:lang="en">Finnish</trace:title>
  </Mother_tongue>
  <Language rdf:ID="Maltese">
    <trace:title xml:lang="en">Maltese</trace:title>
  </Language>
```

```
<Mother_tongue rdf:ID="mo_Croatian">
  <trace:title xml:lang="en">Croatian</trace:title>
</Mother_tongue>
<Language rdf:ID="Turkish">
  <trace:title xml:lang="en">Turkish</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_Slovak">
  <trace:title xml:lang="en">Slovak</trace:title>
</Mother_tongue>
<Mother_tongue rdf:ID="mo_Icelandic">
  <trace:title xml:lang="en">Icelandic</trace:title>
</Mother_tongue>
<Mother_tongue rdf:ID="mo_Bulgarian">
  <trace:title xml:lang="en">Bulgarian</trace:title>
</Mother_tongue>
<Language rdf:ID="Danish">
  <trace:title xml:lang="en">Danish</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_Romanian">
  <trace:title xml:lang="en">Romanian</trace:title>
</Mother_tongue>
<Language rdf:ID="Hungarian">
  <trace:title xml:lang="en">Hungarian</trace:title>
```

```
  </Language>
  <Language rdf:ID="Swedish">
    <trace:title xml:lang="en">Swedish</trace:title>
  </Language>
  <Mother_tongue rdf:ID="mo_Portuguese">
    <trace:title xml:lang="en">Portuguese</trace:title>
  </Mother_tongue>
  <Knowledge rdf:ID="Knowledge_Transportation">
    <hyponymOf>
      <Knowledge rdf:ID="Knowledge_Communications">
        <trace:description xml:lang="en">Knowledge of the science and art of delivering information</trace:description>
        <hypernymOf rdf:resource="#Knowledge_Transportation"/>
        <hypernymOf>
          <Knowledge rdf:ID="Knowledge_Communications_and_Media">
            <trace:description xml:lang="en">Knowledge of media production, communication, and dissemination techniques and methods. This includes alternative ways to inform and entertain via written, oral, and visual media.</trace:description>
            <hyponymOf rdf:resource="#Knowledge_Communications"/>
            <trace:title xml:lang="en">Communications and Media</trace:title>
          </Knowledge>
        </hypernymOf>
        <hypernymOf>
```

```
        <Knowledge rdf:ID="Knowledge_Telecommunications">
          <trace:description xml:lang="en">Knowledge of transmission, broadcasting, switching, control, and operation of telecommunications systems.</trace:description>
          <trace:title xml:lang="en">Telecommunications</trace:title>
          <hyponymOf rdf:resource="#Knowledge_Communications"/>
        </Knowledge>
      </hypernymOf>
      <trace:title xml:lang="en">Communications</trace:title>
    </Knowledge>
  </hyponymOf>
  <trace:description xml:lang="en">Knowledge of principles and methods for moving people or goods by air, rail, sea, or road, including the relative costs and benefits.</trace:description>
  <trace:title xml:lang="en">Transportation</trace:title>
</Knowledge>
<Language rdf:ID="Latvian">
  <trace:title xml:lang="en">Latvian</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_Swedish">
  <trace:title xml:lang="en">Swedish</trace:title>
</Mother_tongue>
<Mother_tongue rdf:ID="mo_Latvian">
  <trace:title xml:lang="en">Latvian</trace:title>
```

```
</Mother_tongue>
<Language rdf:ID="Czech">
  <trace:title xml:lang="en">Czech</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_Hungarian">
  <trace:title xml:lang="en">Hungarian</trace:title>
</Mother_tongue>
<Language rdf:ID="German">
  <trace:title xml:lang="en">German</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_German">
  <trace:title xml:lang="en">German</trace:title>
</Mother_tongue>
<Mother_tongue rdf:ID="mo_Estonian">
  <trace:title xml:lang="en">Estonian</trace:title>
</Mother_tongue>
<Language rdf:ID="Finnish">
  <trace:title xml:lang="en">Finnish</trace:title>
</Language>
<Mother_tongue rdf:ID="mo_Greek">
  <trace:title xml:lang="en">Greek</trace:title>
  <trace:description xml:lang="en">Greek, Modern (1453-)</trace:description>
</Mother_tongue>
```

```
  <Mother_tongue rdf:ID="mo_Maltese">
    <trace:title xml:lang="en">Maltese</trace:title>
  </Mother_tongue>
  <Language rdf:ID="Estonian">
    <trace:title xml:lang="en">Estonian</trace:title>
  </Language>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355)
http://protege.stanford.edu -->
```

Appendix D

ENet.java

```
package org.UoR.competency.mapping;


import java.io.InputStream;

import java.rmi.server.UID;

import java.util.ArrayList;

import java.util.List;


import org.UoR.competency.util.NotURIException;

import org.UoR.competency.util.OntologyCorruptException;


import com.hp.hpl.jena.ontology.OntClass;

import com.hp.hpl.jena.ontology.OntProperty;

import com.hp.hpl.jena.query.Query;

import com.hp.hpl.jena.query.QueryExecution;

import com.hp.hpl.jena.query.QueryExecutionFactory;

import com.hp.hpl.jena.query.QueryFactory;

import com.hp.hpl.jena.query.QuerySolution;

import com.hp.hpl.jena.query.ResultSet;

import com.hp.hpl.jena.rdf.model.RDFNode;

import com.hp.hpl.jena.rdf.model.Resource;
```

```
import com.hp.hpl.jena.rdf.model.StmtIterator;

import com.hp.hpl.jena.util.iterator.ExtendedIterator;


/**

 * ENET competency mapping. This implements the full ENETinterface and

 * ModelHandler interface. The model will load in the Trace upper competency

 * ontology together with other needed datastructures during the initialisation

 * processes. Care should be taken by developers not to delete or modify these

 * datastructures to ensure integrity with other Trace enabled tools.<br>

 * The functionality of deleting and modification has been allowed to enable the

 * production of tools for internal handling of Trace ontologies, however tools

 * should not rely on these functionalities, because they might (or will) be

 * removed at a later stage from the ENet class, when an ENetInternal class

 * would enable manipulation of the before mentioned datastructures.<br>

 * URI's in the system are all URI references (URIrefs

 * http://www.w3.org/TR/rdf-primer Appendix A). Prefixes are set using

 * setUserModel methods. The prefix of the model in userspace is always an empty

 * string, so the prefix that is being set is the prefix of the model being

 * moved away from the userspace.<br>

 * n3-notation URIrefs can be used, so <br>

 * <br>

 * this.model = new ENet();<br>

 * this.model.setUserNamespace("http://www.cs.rdg.ac.uk/test");<br>
```

```
* this.model.addCompetencyProfile(":test"); <br>
* <br>
* is the same as: <br>
* <br>
* this.model = new ENet();<br>
* this.model.setUserNamespace("http://www.cs.rdg.ac.uk/test");<br>
* this.model.addCompetencyProfile("http://www.cs.rdg.ac.uk/test#test"); <br>
* this creates a new competency profile with the uri
* http://www.cs.rdg.ac.uk/test#test<br>
* <br>
* <br>
* The default prefixes are:<br>
* trace, compUpper, rdf, rdfs<br>
* <br>
* These are mostly added for the convenience of the API developer (me), but can
* be used, although using trace and compUpper should be done with care (or
* preferably not at all.) The datamodel consists of a model in userspace and
* many submodels. A submodel can be put into the userspace by setUserModel and
* the old usermodel is put into a submodel given the basePrefix (or a random
* prefix). All changes are performed in userspace. The default saveModel method
* only save the userspace model Saving the complete model is possible but
* discouraged because it saves all models as one big model including the trace
* ontologies.<br>
```

```
* <br>
* The following figure is a figure of the ENet datamodel:<br>
* <img src="doc-files/ENet_data_model.jpg" alt="should have been the
* datamodel!"/><br>
* <br>
*
* The figure shows 5 different functionalities:<br>
* <br>
* Loaded: loadModel methods<br>
* Saved: saveModel methods (preferably the 1 argument version)<br>
* Swopped: setUserModel methods. Swops a model into userspace. Old userspace
* becomes another model outside userspace<br>
* Changed: All set* add* and delete* methods work only in userspace. URIrefs to
* other models can be used in userspace, but the new knowledge will be store in
* the userspace model. (Caution - Deletion will only happen within userspace
* too, so if the content is outside the userspace nothing will be deleted!)<br>
* Referenced: All is*, get* and list* methods uses the complete knowledge of
* the data model to produce the results<br>
*
*
* @author Karsten Oster Lundqvist
* @author k.o.lundqvist at reading "dot" ac "dot" uk
* @author University of Reading
```

```
 * @author TRACE Leonardo project
 */
public class ENet extends OWLModelHandler implements ENetInterface
{

    /**
        * URI of structures holding the TRACE competency ontology. For
        * conveniece this is stored in OWL
        */
    protected String traceURI = "http://www.cs.reading.ac.uk/traceSIMPLE";

    public String getTraceURI()
    {
        return this.traceURI;
    }

    public String getCompetencyURI()
    {
        return getTraceURI() + "#competency";
    }

    private String getTraceAllURI()
    {
```

```
        return getTraceURI() + "#all";
}

private String getTraceAnyURI()
{
        return getTraceURI() + "#any";
}

private String getTraceIfURI()
{
        return getTraceURI() + "#if";
}

private String getTraceIfTrueURI()
{
        return getTraceURI() + "#if_true";
}

private String getTraceIfFalseURI()
{
        return getTraceURI() + "#if_false";
}
```

```
@SuppressWarnings("unused")
private String getTraceChildrenPlainURI()
{
	return getTraceURI() + "#children_plain";
}

private String getTraceChildrenURI()
{
	return getTraceURI() + "#children";
}

public String getTraceHasProficiencyScoreURI()
{
	return getTraceURI() + "#hasProficiencyScore";
}

private String getTraceHasCompetencyURI()
{
	return getTraceURI() + "#hasCompetency";
}

public String getTraceProficiencyRequiredURI()
{
```

```
        return getTraceURI() + "#proficienceRequired";
}

public String getTraceProficiencyDesiredURI()
{
        return getTraceURI() + "#proficienceDesired";
}

private String getTraceProficiencyURI()
{
        return getTraceURI() + "#proficiency_score";
}

private String getTraceGraphURI()
{
        return getTraceURI() + "#graph";
}

private String getTraceTitleURI()
{
        return getTraceURI() + "#title";
}
```

```
private String getTraceDescriptionURI()
{
	return this.getTraceURI() + "#description";
}

public String getTraceCompetencyProfileURI()
{
	return getTraceURI() + "#VSRCM";
}

public String getTraceRCDURI()
{
	return this.getTraceURI() + "#RCD";
}

private String getTraceNodeURI()
{
	return this.getTraceURI() + "#node";
}

private String getTraceEntrynodesURI()
{
	return getTraceURI() + "#entryNodes";
```

```
    }

    private String getTraceDefaultEntrynodeURI()
    {
        return getTraceURI() + "#defaultEntryNode";
    }

    /**
     * URI of the upper ontology of the TRACE ontology
     */
    private String competencyUpperURI =
"http://www.cs.reading.ac.uk/competencyUpper";

    public String getCompetencyUpperURI()
    {
        return this.competencyUpperURI;
    }

    public String getRelationMeronym()
    {
        return this.competencyUpperURI + "#meronymOf";
    }
```

```
public String getRelationHolonym()
{
    return this.competencyUpperURI + "#holonymOf";
}

public String getRelationHyponym()
{
    return this.competencyUpperURI + "#hyponymOf";
}

public String getRelationHypernym()
{
    return this.competencyUpperURI + "#hypernymOf";
}

public String getRelationAntonym()
{
    return this.competencyUpperURI + "#antonymOf";
}

public String getRelationSynonym()
{
    return this.competencyUpperURI + "#synonymOf";
```

```
    }

    public ENet()
    {
        InputStream is = getClass().getResourceAsStream("/rcd.owl");
        if (is != null)
            super.loadModel(is, "trace");
        else
            super.loadModel("trace", this.traceURI, "RDF/XML",
"file:ontology/rcd.owl");

        is = getClass().getResourceAsStream("/competencyUpper.owl");
        if (is != null)
            super.loadModel(is, "compUpper");
        else
            super.loadModel("compUpper", this.competencyUpperURI, "RDF/XML",
"file:ontology/competencyUpper.owl");

        this.addPrefix("rdfs", "http://www.w3.org/2000/01/rdf-schema");
        this.addPrefix("rdf", "http://www.w3.org/1999/02/22-rdf-syntax-ns");

        this.setUserNamespace("http://example.cs.reading.ac.uk/" + new UID());
```

```
        // Model m = this.model.getBaseModel();
        // this.modelMap.put(this.getUserNamespace(), m);
    }

    public ENet(String uri)
    {
        InputStream is = getClass().getResourceAsStream("/rcd.owl");
        if (is != null)
            super.loadModel(is, "trace");
        else
            super.loadModel("trace", this.traceURI, "RDF/XML",
"file:ontology/rcd.owl");

        is = getClass().getResourceAsStream("/competencyUpper.owl");
        if (is != null)
            super.loadModel(is, "compUpper");
        else
            super.loadModel("compUpper", this.competencyUpperURI, "RDF/XML",
"file:ontology/competencyUpper.owl");

        this.addPrefix("rdfs", "http://www.w3.org/2000/01/rdf-schema");
        this.addPrefix("rdf", "http://www.w3.org/1999/02/22-rdf-syntax-ns");
```

```
        try
        {
            uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
            return;
        }

        this.setUserNamespace(uri);

        // Model m = this.model.getBaseModel();
        // this.modelMap.put(uri, m);
    }

    public ENet(ENet old)
    {
        super(old);

        this.setUserNamespace(old.getUserNamespace());
    }

    /*
     * (non-Javadoc)
```

```
        *
        * @see
org.UoR.competency.mapping.ENetInterface#listEntryNodes(java.lang.String)
        */
    public List<String> listEntryNodes(String competencyURI)
    {
            ArrayList<String> result = new ArrayList<String>();


            try
            {
                competencyURI = this.getFullURI(competencyURI);
            } catch (NotURIException e)
            {
                return result;
            }


            String queryString = this.prefix + "SELECT ?entryURI " + "WHERE" + "{ <"
+ competencyURI + "> trace:graph ?graph . " + "?graph trace:entryNodes ?entryURI
.}";


            Query query = QueryFactory.create(queryString);
            QueryExecution qexec = QueryExecutionFactory.create(query, this.model);
            try
```

```
        {
            ResultSet results = qexec.execSelect();
            while (results.hasNext())
            {
                QuerySolution qs = results.nextSolution();
                RDFNode node = qs.get("?entryURI");
                if (node != null)
                {
                    if (node.isResource())
                    {
                        result.add(((Resource) node).getURI());
                    }
                }
            }
        } catch (Exception e)
        {
        } finally
        {
            qexec.close();
        }

        return result;
    }
```

```
/*
     * (non-Javadoc)
     *
     * @see
org.UoR.competency.mapping.ENetInterface#listChildNodesALL(java.lang.String)
     */
public List<String> listALLnodes(String parentNode)
{
        return this.getChildNodes(parentNode, this.getTraceAllURI());
}


/*
     * (non-Javadoc)
     *
     * @see
org.UoR.competency.mapping.ENetInterface#listChildNodesANY(java.lang.String)
     */
public List<String> listANYnodes(String parentNode)
{
        return this.getChildNodes(parentNode, this.getTraceAnyURI());
}
```

```
    /*
        * (non-Javadoc)
        *
        * @see
org.UoR.competency.mapping.ENetInterface#listIFnodes(java.lang.String)
        */
    public List<String> listIfNodes(String parentNode)
    {
            return this.getChildNodes(parentNode, this.getTraceIfURI());
    }


    /*
        * (non-Javadoc)
        *
        * @see
org.UoR.competency.mapping.ENetInterface#listIFTruenodes(java.lang.String)
        */
    public List<String> listIfTrueNodes(String parentNode)
    {
            return this.getChildNodes(parentNode, this.getTraceIfTrueURI());
    }


    /*
```

```
	* (non-Javadoc)
	*
	* @see
org.UoR.competency.mapping.ENetInterface#listIFFalsenodes(java.lang.String)
	*/
	public List<String> listIfFalseNodes(String parentNode)
	{
		return this.getChildNodes(parentNode, this.getTraceIfFalseURI());
	}

	/*
	* (non-Javadoc)
	*
	* @see
org.UoR.competency.mapping.ENetInterface#listEveryChildNode(java.lang.String)
	*/
	public List<String> listEveryChildNode(String parentNode)
	{
		return this.getChildNodes(parentNode, this.getTraceChildrenURI());
	}

	/*
	* (non-Javadoc)
```

```
        *
        * @see
org.UoR.competency.mapping.ENetInterface#listEveryParentNode(java.lang.String)
        */
    public List<String> listEveryParentNode(String childNode)
    {
            return this.getParentNodes(childNode, this.getTraceChildrenURI());
    }


    /*
        * (non-Javadoc)
        *
        * @see
org.UoR.competency.mapping.ENetInterface#getCompetency(java.lang.String)
        */
    public String getCompetencyOfNode(String node)
    {
            try
            {
                node = this.getFullURI(node);
            } catch (NotURIException e)
            {
                return null;
```

```
        }

        List<String> comp = this.getChildNodes(node,
this.getTraceHasCompetencyURI());
        if (comp.isEmpty())
            return null;
        else
            return (String) comp.toArray()[0];
    }

    /*
        * (non-Javadoc)
        *
        * @see
org.UoR.competency.mapping.ENetInterface#setCompetencyOfNode(java.lang.Strin
g,
        *     java.lang.String)
        */
    public void setCompetencyOfNode(String node, String competencyURI)
    {
        // remove any old competency
        String oldCompetency = this.getCompetencyOfNode(node);
        if (oldCompetency != null)
```

```
            this.removeStatement(node,                    getTraceHasCompetencyURI(),
oldCompetency);

        this.addStatement(node, getTraceHasCompetencyURI(), competencyURI);
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeCompetencyOfNode(java.lang.
String)
        */
    public void removeCompetencyOfNode(String node)
    {
        this.removeStatement(node, getTraceHasCompetencyURI(), null);
    }

    /*
        * (non-Javadoc)
        *
        * @see org.UoR.competency.mapping.ENetInterface#listRelations()
        */
```

```
public List<String> listCompetencyRelations()
{
        ArrayList<String> result = new ArrayList<String>();

        result.add(this.getCompetencyUpperURI() + "#hyponymOf");
        result.add(this.getCompetencyUpperURI() + "#holonymOf");
        result.add(this.getCompetencyUpperURI() + "#meronymOf");
        result.add(this.getCompetencyUpperURI() + "#hypernymOf");
        result.add(this.getCompetencyUpperURI() + "#antonymOf");
        result.add(this.getCompetencyUpperURI() + "#synonymOf");

        return result;
}

/*
    * (non-Javadoc)
    *
    * @see org.UoR.competency.mapping.ENetInterface#listProficiencyRelations()
    */
public List<String> listProficiencyRelations()
{
        ArrayList<String> result = new ArrayList<String>();
```

```
        result.add(this.getCompetencyUpperURI() + "#holonymOf");
        result.add(this.getCompetencyUpperURI() + "#meronymOf");
        result.add(this.getCompetencyUpperURI() + "#antonymOf");
        result.add(this.getCompetencyUpperURI() + "#synonymOf");

        return result;
    }

    /*
     * (non-Javadoc)
     *
     *                                                                                    @see
org.UoR.competency.mapping.ENetInterface#addProficiencyScore(java.lang.String)
     */
    public void addProficiencyScore(String URI)
    {
        if (this.isUniqueURI(URI))
            this.addStatement(URI, getRDFtypeURI(), this.getTraceProficiencyURI());
    }

    /*
     * (non-Javadoc)
     *
```

```
     * @see org.UoR.competency.mapping.ENetInterface#listProficiencyScores()
     */
    public List<String> listProficiencyScores()
    {
        return this.getParentNodes(this.getTraceProficiencyURI(),
this.getRDFtypeURI());
    }

    /*
     * (non-Javadoc)
     *
     * @see
org.UoR.competency.mapping.ENetInterface#addProficiencyScore(java.lang.String)
     */
    public void deleteProficiencyScore(String URI)
    {
        this.removeStatement(URI, this.getRDFtypeURI(),
this.getTraceProficiencyURI());
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getRequiredProficiency(java.lang.Strin
g)
     */
   public String getRequiredProficiency(String node)
   {
        List<String>          prof          =          this.getChildNodes(node,
this.getTraceProficiencyRequiredURI());
        if (prof.isEmpty())
        {
           List<String> parents = this.listEveryParentNode(node);


           String score = null;
           for(String p : parents)
           {
                 String dummy = this.getRequiredProficiency(p);


                 if(score == null)
                 {
                    score = dummy;
                 }
                 else
                 {
```

```
                    if(this.listRelatedProficiencyScores(score,
this.getRelationHolonym()).contains(dummy))
                        score = dummy;
                }
            }


            return score;
        }
        else
            return (String) prof.toArray()[0];
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setRequiredProficiency(java.lang.Strin
g,
        *      java.lang.String)
        */
    public void setRequiredProficiency(String node, String proficiencyURI)
    {
        String oldCompetency = this.getRequiredProficiency(node);
```

```
        if (oldCompetency != null)
            this.removeStatement(node,                getTraceProficiencyRequiredURI(),
oldCompetency);

        this.addStatement(node, getTraceProficiencyRequiredURI(), proficiencyURI);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                          @see
org.UoR.competency.mapping.ENetInterface#removeRequiredProficiency(java.lang.
String)
     */
    public void removeRequiredProficiency(String node)
    {
        this.removeStatement(node, getTraceProficiencyRequiredURI(), null);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getDesiredProficiency(java.lang.String)
     */
   public String getDesiredProficiency(String node)
   {
        List<String>          prof          =          this.getChildNodes(node,
this.getTraceProficiencyDesiredURI());
        if (prof.isEmpty())
        {
           List<String> parents = this.listEveryParentNode(node);


           String score = null;
           for(String p : parents)
           {
                 String dummy = this.getDesiredProficiency(p);


                 if(score == null)
                 {
                    score = dummy;
                 }
                 else
                 {
```

```
                    if(this.listRelatedProficiencyScores(score,
this.getRelationHolonym()).contains(dummy))
                        score = dummy;
                }
            }


            return score;
        }
        else
            return (String) prof.toArray()[0];
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setDesiredProficiency(java.lang.String,
        *      java.lang.String)
        */
    public void setDesiredProficiency(String node, String proficiencyURI)
    {
        String oldCompetency = this.getRequiredProficiency(node);
        if (oldCompetency != null)
```

```
        this.removeStatement(node, getTraceProficiencyDesiredURI(), oldCompetency);

        this.addStatement(node, getTraceProficiencyDesiredURI(), proficiencyURI);
    }

    /*
     * (non-Javadoc)
     *
     * @see org.UoR.competency.mapping.ENetInterface#removeDesiredProficiency(java.lang.String)
     */
    public void removeDesiredProficiency(String node)
    {
        this.removeStatement(node, getTraceProficiencyDesiredURI(), null);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getHasProficiencyScore(java.lang.Strin
g)
     */
   public String getHasProficiencyScore(String node)
   {
        List<String>          prof          =          this.getChildNodes(node,
this.getTraceHasProficiencyScoreURI());
        if (prof.isEmpty())
        {
           List<String> parents = this.listEveryParentNode(node);


           String score = null;
           for(String p : parents)
           {
                 String dummy = this.getHasProficiencyScore(p);


                 if(score == null)
                 {
                    score = dummy;
                 }
                 else
                 {
```

```
                    if(this.listRelatedProficiencyScores(score,
this.getRelationHolonym()).contains(dummy))
                        score = dummy;
                }
            }


            return score;
        }
        else
            return (String) prof.toArray()[0];
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setHasProficiencyScore(java.lang.Strin
g,
        *      java.lang.String)
        */
    public void setHasProficiencyScore(String node, String proficiencyURI)
    {
        String oldCompetency = this.getRequiredProficiency(node);
```

```
        if (oldCompetency != null)
            this.removeStatement(node,                getTraceHasProficiencyScoreURI(),
oldCompetency);

        this.addStatement(node,                       getTraceHasProficiencyScoreURI(),
proficiencyURI);
    }

    /*
        * (non-Javadoc)
        *
        *                                                                          @see
org.UoR.competency.mapping.ENetInterface#setHasProficiencyScore(java.lang.Strin
g)
        */
    public void removeHasProficiencyScore(String node)
    {
        this.removeStatement(node, getTraceHasProficiencyScoreURI(), null);
    }

    /*
        * (non-Javadoc)
        *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaCompetency(java.lang.String)
     */
   public boolean isURIaCompetency(String uri)
   {
        try
        {
            uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
            return false;
        }

        String queryString = this.prefix + "ASK { <" + uri + "> rdf:type ?a FILTER
(?a = trace:RCD || ?a = trace:VSRCM) . }";

        Query query = QueryFactory.create(queryString);
        QueryExecution qexec = QueryExecutionFactory.create(query, this.model);

        if (qexec.execAsk() == true)
            return true;
        else
            return false;
```

```
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setCompetencyTitle(java.lang.String,
        *      java.lang.String, java.lang.String)
        */
    public void setCompetencyTitle(String competencyURI, String title,
                                        String language)
    {
            String oldCompetency;
            try
            {
                oldCompetency = this.getCompetencyTitle(competencyURI, language);
                this.removeStatement(competencyURI,                getTraceTitleURI(),
oldCompetency);
            } catch (OntologyCorruptException e)
            {
            }

            this.addStatement(competencyURI, getTraceTitleURI(), title, language);
```

```
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addProfileMapping(java.lang.String)
        */
    public void addCompetencyProfile(String uri)
    {
          if (this.isUniqueURI(uri))
          {
              this.addStatement(uri,                                getRDFtypeURI(),
getTraceCompetencyProfileURI());
          }
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeProfileMapping(java.lang.Strin
g)
```

```
        */
    public void deleteCompetencyProfile(String uri)
    {
        if (this.isURIaCompetencyProfile(uri))
        {
            this.deleteGraph(uri);
            this.removeStatement(uri,                                    getRDFtypeURI(),
getTraceCompetencyProfileURI());
        }
    }


    /*
        * (non-Javadoc)
        *
        *                                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeRCDCompetency(java.lang.Stri
ng)
        */
    public void deleteRCDCompetency(String uri)
    {
        try
        {
            uri = this.getFullURI(uri);
```

```
        } catch (NotURIException e)
        {
            return;
        }

        if (this.isURIaRCDcompetency(uri))
            this.model.getOntResource(uri).remove();
    }

    /*
        * (non-Javadoc)
        *
        *                                                                                    @see
org.UoR.competency.mapping.ENetInterface#addCompetencyOfClass(java.lang.Strin
g,
        *      java.lang.String, java.lang.String)
        */
    public void addRCDCompetencyOfClass(String uri, String classUri)
    {
        if (this.isUniqueURI(uri))
        {
            this.addStatement(uri, getRDFtypeURI(), classUri);
        }
```

```
}

public String getKnowledgeURI()
{
        return this.getCompetencyUpperURI() + "#Knowledge";
}

public String getSkillURI()
{
        return this.getCompetencyUpperURI() + "#Skill";
}

public String getOtherURI()
{
        return this.getCompetencyUpperURI() + "#Other";
}

/*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listCompetenciesOfClass(java.lang.Stri
ng)
```

```
     */
    public List<String> listCompetenciesOfClass(String uri)
    {
        return this.getParentNodes(uri, getRDFtypeURI());
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeTitle(java.lang.String,
     *      java.lang.String, java.lang.String)
     */
    public void deleteCompetencyTitle(String uri, String oldTitle,
                                      String language)
    {
        if (language != null)
            this.removeStatement(uri, this.getTraceTitleURI(), oldTitle, language);
        else
            this.removeStatement(uri, this.getTraceTitleURI(), oldTitle);
    }

    /*
```

```
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeDescription(java.lang.String,
        *      java.lang.String)
        */
    /*
        * public void removeDescription(String uri, String description) {
        * this.removeStatement(uri, getTraceDescriptionURI(), description); }
        */


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addDescription(java.lang.String,
        *      java.lang.String, java.lang.String)
        */
    public void setCompetencyDescription(String uri, String description,
                                         String language)
    {
            if (language != null)
                this.addStatement(uri, getTraceDescriptionURI(), description, language);
```

```
        else
            this.addStatement(uri, getTraceDescriptionURI(), description);
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getDescription(java.lang.String,
        *     java.lang.String)
        */
    public String getCompetencyDescription(String uri, String language) throws
OntologyCorruptException
    {
        List<String> result;

        result = this.getChildNodes(uri, getTraceDescriptionURI(), language);

        if (result.size() == 0)
            return null;

        if (result.size() == 1)
            return result.iterator().next();
```

```
        else
        {
            throw new OntologyCorruptException();
        }
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getCompetencyTitle(java.lang.String,
     *      java.lang.String)
     */
    public String getCompetencyTitle(String uri, String language) throws
OntologyCorruptException
    {
        String result = this.getString(uri, this.getTraceTitleURI(), language);

        return result;
    }

    /*
     * (non-Javadoc)
```

```
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listCompetencyTitles(java.lang.String)
     */
   public List<String> listCompetencyTitles(String uri)
   {
         List<String> result = this.getChildNodes(uri, "trace:title");


         // ------------------------------------------------------------//
         // Should be deprecated!!
         if (result.size() == 0)
             result.addAll(this.getChildNodes(uri, "rdfs:label"));


         if (result.size() == 0)
             result.addAll(this.getChildNodes(uri, "rdfs:comment"));
         // ------------------------------------------------------------//


         return result;
   }


   /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listDescription(java.lang.String)
     */
   public List<String> listCompetencyDescription(String uri)
   {
         return this.getChildNodes(uri, getTraceDescriptionURI());
   }


   /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeDescription(java.lang.String,
     *      java.lang.String, java.lang.String)
     */
   public void deleteCompetencyDescription(String uri, String description,
                                                 String language)
   {
         if (language != null)
            this.removeStatement(uri,        getTraceDescriptionURI(),        description,
language);
         else
            this.removeStatement(uri, getTraceDescriptionURI(), description);
```

```
}

/*
 * (non-Javadoc)
 *
 *                                                                    @see
org.UoR.competency.mapping.ENetInterface#changeDescription(java.lang.String,
 *     java.lang.String)
 */
/*
 * public void changeDescription(String uri, String newDescription) {
 * this.removeStatement(uri, getTraceDescriptionURI(), null);
 * this.addStatement(uri, getTraceDescriptionURI(), newDescription); }
 */

/*
 * (non-Javadoc)
 *
 *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addRelation(java.lang.String,
 *     java.lang.String, java.lang.String)
 */
public void addRelationBetweenCompetencies(String compFromURI,
```

```
                                                    String relationURI,
                                                    String compToURI)
    {
        this.addStatement(compFromURI, relationURI, compToURI);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeRelation(java.lang.String,
     *     java.lang.String, java.lang.String)
     */
    public void removeRelationBetweenCompetencies(String compFromURI,
                                                    String relationURI,
                                                    String compToURI)
    {
        this.removeStatement(compFromURI, relationURI, compToURI);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaGraph(java.lang.String)
     */
   public boolean isURIaGraph(String uri)
   {
        try
        {
           uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
           return false;
        }

        String queryString = this.prefix + "ASK { <" + uri + "> <" +
this.getRDFtypeURI() + "> <" + this.getTraceGraphURI() + "> . }";

        Query query = QueryFactory.create(queryString);
        QueryExecution qexec = QueryExecutionFactory.create(query, this.model);

        return qexec.execAsk();

   }
```

```
    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addGraph(java.lang.String,
     *      java.lang.String)
     */
    public void setGraph(String vsrcmURI, String graphURI)
    {
        this.removeGraph(vsrcmURI);
        this.addStatement(graphURI, getRDFtypeURI(), getTraceGraphURI());
        this.addStatement(vsrcmURI, getTraceGraphURI(), graphURI);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeGraph(java.lang.String)
     */
    public void removeGraph(String vsrcmURI)
    {
        try
```

```
        {
            String graphURI = this.getGraph(vsrcmURI);
            this.removeStatement(vsrcmURI, getTraceGraphURI(), graphURI);
            // TODO consider delete method
        } catch (OntologyCorruptException e)
        {
        }
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getGraph(java.lang.String)
     */
    public String getGraph(String vsrcmURI) throws OntologyCorruptException
    {
        List<String>        graphURIs        =        this.getChildNodes(vsrcmURI,
this.getTraceGraphURI());

        if (graphURIs.size() == 0)
        {
            return null;
```

```
        }
        else
            if (graphURIs.size() == 1)
            {
                return (String) graphURIs.iterator().next();
            }
            else
                throw new OntologyCorruptException();
    }

    public List<String> listGraphs()
    {
        ArrayList<String> result = new ArrayList<String>();

        OntClass graphClass = this.model.createClass(this.getTraceGraphURI());
        OntProperty typeProperty =
this.model.createOntProperty(this.getRDFtypeURI());

        StmtIterator it = this.model.listStatements((Resource) null, typeProperty,
graphClass);
        String uri;
        while (it.hasNext())
        {
```

```
            uri = ((Resource) it.next()).getURI();
            if (uri != null)
                result.add(uri);
        }

        return result;
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#deleteGraph(java.lang.String)
     */
    public void deleteGraph(String graphURI)
    {

        List<String> childNodes = this.listEntryNodes(graphURI);

        for (String nodeURI : childNodes)
        {
            this.deleteNode(nodeURI);
        }
```

```
        this.removeStatement(graphURI, null, null);
        this.removeStatement(null, null, graphURI);
    }

    private void addNode(String nodeURI)
    {
        this.addStatement(nodeURI, this.getRDFtypeURI(), this.getTraceNodeURI());
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaNode(java.lang.String)
     */
    public boolean isURIaNode(String uri)
    {
        try
        {
            uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
```

```
            return false;
        }

        String queryString = this.prefix + "ASK { <" + uri + "> <" +
this.getRDFtypeURI() + "> <" + this.getTraceNodeURI() + "> . }";
        // this.prefix + "ASK { ?a ?b <"+ uri + "> FILTER (?b = trace:entryNodes
        // || ?b = trace:if || ?b = trace:any || ?b = trace:all) . }";

        Query query = QueryFactory.create(queryString);
        QueryExecution qexec = QueryExecutionFactory.create(query, this.model);

        return qexec.execAsk();

    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addEntryNode(java.lang.String,
     *     java.lang.String)
     */
    public void addEntryNode(String graphURI, String nodeURI)
```

```
    {
        this.addNode(nodeURI);
        this.addStatement(graphURI, getTraceEntrynodesURI(), nodeURI);

        try
        {
            this.getDefaultEntryNode(graphURI);
        } catch (OntologyCorruptException ex)
        {
            this.setDefaultEntryNode(graphURI, nodeURI);
        }
    }

    /*
        * (non-Javadoc)
        *
        *                                                                                    @see
org.UoR.competency.mapping.ENetInterface#addDefaultEntryNode(java.lang.String,
        *      java.lang.String)
        */
    public void setDefaultEntryNode(String graphURI, String nodeURI)
    {
        this.removeDefaultEntryNode(graphURI);
```

```
        this.addStatement(graphURI, getTraceDefaultEntrynodeURI(), nodeURI);
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeDefaultEntryNode(java.lang.Str
ing,
        *      java.lang.String)
        */
    public void removeDefaultEntryNode(String graphURI)
    {
        this.removeStatement(graphURI, getTraceDefaultEntrynodeURI(), null);
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getDefaultEntryNode(java.lang.String)
        */
```

```
public String getDefaultEntryNode(String graphURI) throws OntologyCorruptException
{
        // TODO ensure ontology integrity
        List<String> result = this.getChildNodes(graphURI, getTraceDefaultEntrynodeURI());

        if (result.size() == 1)
            return result.iterator().next();

        throw new OntologyCorruptException();
}

/*
     * (non-Javadoc)
     *
     * @see org.UoR.competency.mapping.ENetInterface#removeEntryNode(java.lang.String,
     *     java.lang.String)
     */
public void removeEntryNode(String graphURI, String nodeURI) throws OntologyCorruptException
{
```

```
        this.removeStatement(graphURI, getTraceEntrynodesURI(), nodeURI);


        // TODO ensure integrity => should not remove a defaultEntryNode without
        // a new.
        try
        {
            String defaultEntryNodeURI = this.getDefaultEntryNode(graphURI);
            if (defaultEntryNodeURI.equals(nodeURI))
            {
                this.removeDefaultEntryNode(graphURI);
                throw new OntologyCorruptException();
            }
        } catch (OntologyCorruptException e)
        {
            throw new OntologyCorruptException();
        }
    }


    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addAllNode(java.lang.String,
```

```
     *      java.lang.String)
     */
    public void addAllNode(String parentNodeURI, String nodeURI)
    {
        this.addNode(nodeURI);
        this.addStatement(parentNodeURI, getTraceAllURI(), nodeURI);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeAllNode(java.lang.String,
     *      java.lang.String)
     */
    public void removeAllNode(String parentNodeURI, String nodeURI)
    {
        this.removeStatement(parentNodeURI, getTraceAllURI(), nodeURI);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addAnyNode(java.lang.String,
     *      java.lang.String)
     */
   public void addAnyNode(String parentNodeURI, String nodeURI)
   {
         this.addNode(nodeURI);
         this.addStatement(parentNodeURI, getTraceAnyURI(), nodeURI);
   }

   /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeAnyNode(java.lang.String,
     *      java.lang.String)
     */
   public void removeAnyNode(String parentNodeURI, String nodeURI)
   {
         this.removeStatement(parentNodeURI, getTraceAnyURI(), nodeURI);
   }

   /*
```

```
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addIfNode(java.lang.String,
     *     java.lang.String)
     */
    public void addIfNode(String parentNodeURI, String nodeURI)
    {
        this.addNode(nodeURI);
        this.addStatement(parentNodeURI, getTraceIfURI(), nodeURI);
    }


    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeIfNode(java.lang.String,
     *     java.lang.String)
     */
    public void removeIfNode(String parentNodeURI, String nodeURI)
    {
        this.removeStatement(parentNodeURI, getTraceIfURI(), nodeURI);
    }
```

```
/*
 * (non-Javadoc)
 *
 *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addIfNode(java.lang.String,
 *     java.lang.String)
 */
public void addIfTrueNode(String parentNodeURI, String nodeURI)
{
    try
    {
        parentNodeURI = this.getFullURI(parentNodeURI);
    } catch (NotURIException e)
    {
        return;
    }

    List<String> nodes = this.listEveryParentNode(parentNodeURI);

    for (String n : nodes)
    {
        if (this.listIfNodes(n).contains(parentNodeURI))
```

```
            {
                this.addNode(nodeURI);
                this.addStatement(parentNodeURI, getTraceIfTrueURI(), nodeURI);
                return;
            }
        }
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeIfNode(java.lang.String,
     *      java.lang.String)
     */
    public void removeIfTrueNode(String parentNodeURI, String nodeURI)
    {
        this.removeStatement(parentNodeURI, getTraceIfTrueURI(), nodeURI);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addIfNode(java.lang.String,
     *      java.lang.String)
     */
   public void addIfFalseNode(String parentNodeURI, String nodeURI)
   {
        try
        {
            parentNodeURI = this.getFullURI(parentNodeURI);
        } catch (NotURIException e)
        {
            return;
        }

        List<String> nodes = this.listEveryParentNode(parentNodeURI);

        for (String n : nodes)
        {
            if (this.listIfNodes(n).contains(parentNodeURI))
            {
                this.addNode(nodeURI);
                this.addStatement(parentNodeURI, getTraceIfFalseURI(), nodeURI);
                return;
```

```
            }
        }
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeIfNode(java.lang.String,
        *     java.lang.String)
        */
    public void removeIfFalseNode(String parentNodeURI, String nodeURI)
    {
        this.removeStatement(parentNodeURI, getTraceIfFalseURI(), nodeURI);
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeEveryChildNode(java.lang.Strin
g,
        *     java.lang.String)
```

```
    */
public void removeEveryChildNode(String parentNodeURI, String nodeURI)
{
        this.removeAllNode(parentNodeURI, nodeURI);
        this.removeAnyNode(parentNodeURI, nodeURI);
        this.removeIfNode(parentNodeURI, nodeURI);
        this.removeIfTrueNode(parentNodeURI, nodeURI);
        this.removeIfFalseNode(parentNodeURI, nodeURI);

}

/*
    * (non-Javadoc)
    *
    *                                                                    @see
org.UoR.competency.mapping.ENetInterface#deleteNode(java.lang.String)
    */
public void deleteNode(String nodeURI)
{
        List<String> childNodes = this.listEveryChildNode(nodeURI);

        for (String childURI : childNodes)
        {
```

```
            this.deleteNode(childURI);
        }

        this.removeStatement(nodeURI, null, null);
        this.removeStatement(null, null, nodeURI);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addProficiency(java.lang.String,
     *      java.lang.String, java.lang.String)
     */
    public void addProficiency(String nodeURI, String proficiencyTypeURI,
                                    String proficiencyURI)
    {
        this.removeProficiency(nodeURI, proficiencyTypeURI);
        this.addStatement(nodeURI, proficiencyTypeURI, proficiencyURI);
    }

    /*
     * (non-Javadoc)
```

```
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#removeProficiency(java.lang.String,
     *      java.lang.String, java.lang.String)
     */
   public void removeProficiency(String nodeURI, String proficiencyTypeURI)
   {
         this.removeStatement(nodeURI, proficiencyTypeURI, null);
   }


   /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isCompetenciesRelated(java.lang.Strin
g,
     *      java.lang.String, java.lang.String)
     */
   public boolean isCompetenciesRelated(String profFromURI,
                                        String relationURI, String profToURI)
   {
         try
         {
```

```
            profFromURI = this.getFullURI(profFromURI);
            relationURI = this.getFullURI(relationURI);
            profToURI = this.getFullURI(profToURI);
        } catch (NotURIException e)
        {
            return false;
        }

        /*
      * if(this.isProperURI(profFromURI)) profFromURI = "<"+profFromURI+">";
      * if(this.isProperURI(relationURI)) relationURI = "<"+relationURI+">";
      * if(this.isProperURI(profToURI)) profToURI = "<"+profToURI+">";
      */

        String queryString = this.prefix + "ASK " + "{ <" + profFromURI + "> <" +
relationURI + "> <" + profToURI + "> .} ";

        Query query = QueryFactory.create(queryString);

        QueryExecution qexec = QueryExecutionFactory.create(query, this.model);

        return qexec.execAsk();
    }
```

```
/*
 * (non-Javadoc)
 *
 *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listRelatedCompetencies(java.lang.Stri
ng,
 *     java.lang.String)
 */
public List<String> listRelatedCompetencies(String competencyURI,
                                            String relationURI)
{
    return this.getChildNodes(competencyURI, relationURI);
}


/*
 * (non-Javadoc)
 *
 *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listRelatedCompetencies(java.lang.Stri
ng,
 *     java.lang.String, java.lang.String)
 */
```

```
public boolean isProficienciesRelated(String profFromURI,
                                      String relationURI, String profToURI)
{
        try
        {
            profFromURI = this.getFullURI(profFromURI);
            relationURI = this.getFullURI(relationURI);
            profToURI = this.getFullURI(profToURI);
        } catch (NotURIException e)
        {
            return false;
        }

        /*
       * if(this.isProperURI(profFromURI)) profFromURI = "<"+profFromURI+">";
       * if(this.isProperURI(relationURI)) relationURI = "<"+relationURI+">";
       * if(this.isProperURI(profToURI)) profToURI = "<"+profToURI+">";
       */

        String queryString = this.prefix + "ASK " + "{ <" + profFromURI + "> <" +
relationURI + "> <" + profToURI + "> .} ";

        Query query = QueryFactory.create(queryString);
```

```
        QueryExecution qexec = QueryExecutionFactory.create(query, this.model);

        return qexec.execAsk();
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listRelatedProficiencyScores(java.lang.
String,
     *      java.lang.String)
     */
    public List<String> listRelatedProficiencyScores(
                                                        String proficiencyURI,
                                                        String relationURI)
    {
        return this.getChildNodes(proficiencyURI, relationURI);
    }

    /*
     * (non-Javadoc)
```

```
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaRCD(java.lang.String)
     */
   public boolean isURIaRCDcompetency(String uri)
   {
        try
        {
            uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
            return false;
        }
        // if(this.isProperURI(URI) == false) return false;

        List<String> classesURI = this.getChildNodes(uri, this.getRDFtypeURI());

        if (classesURI.contains(getTraceRCDURI()))
            return true;
        else
            return false;
   }
```

```
    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaCompetencyProfile(java.lang.St
ring)
        */
    public boolean isURIaCompetencyProfile(String uri)
    {
            try
            {
                uri = this.getFullURI(uri);
            } catch (NotURIException e)
            {
                return false;
            }
            // if(this.isProperURI(uri) == false) return false;

            List<String> classesURI = this.getChildNodes(uri, this.getRDFtypeURI());

            if (classesURI.contains(this.getTraceCompetencyProfileURI()))
                return true;
            else
```

```
            return false;
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaProficiencyScore(java.lang.Strin
g)
        */
    public boolean isURIaProficiencyScore(String uri)
    {
        try
        {
            uri = this.getFullURI(uri);
        } catch (NotURIException e)
        {
            return false;
        }

        List<String> proficiencyScores = this.listProficiencyScores();

        if (proficiencyScores.contains(uri))
```

```
            return true;
        else
            return false;
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#addSubClass(java.lang.String,
     *      java.lang.String)
     */
    public void addSubClass(String classURI, String subClassURI)
    {
        if (this.isURIaClass(classURI))
        {
            List<String> superClasses = this.listSuperClassesOf(classURI, false);

            if (superClasses.contains(this.getTraceRCDURI()))
                this.addStatement(subClassURI, "rdfs:subClassOf", classURI);
            return;
        }
```

```
        /*
        * if (this.isURIaProficiencyScore(classURI)) {
        * this.addStatement(subClassURI, "rdfs:subClassOf", classURI); }
        */
    }

    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#deleteClass(java.lang.String)
        */
    public void deleteClass(String classURI)
    {
        if (this.isURIaClass(classURI))
        {
            List<String> superClasses = this.listSuperClassesOf(classURI, false);

            if (superClasses.contains(this.getCompetencyUpperURI() + "#knowledge")
&&  superClasses.contains(this.getCompetencyUpperURI() + "#skill") &&
superClasses.contains(this.getCompetencyUpperURI() + "#other"))
            {
                List<String> individuals = this.listCompetenciesOfClass(classURI);
```

```
                individuals.addAll(this.listProficiencyScores());

                if (this.isURIaRCDcompetency(classURI))
                    for (String individual : individuals)
                        this.deleteRCDCompetency(individual);

                if (this.isURIaCompetencyProfile(classURI))
                    for (String individual : individuals)
                        this.deleteCompetencyProfile(individual);

                this.removeStatement(classURI, null, null);
                this.removeStatement(null, null, classURI);
            }
        }
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isProperURI(java.lang.String)
     */
    public boolean isProperURI(String uri)
```

```
    {
        boolean result = super.isProperURI(uri);

        return result || super.isPrefixURI(uri);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isURIaClass(java.lang.String)
     */
    public boolean isURIaClass(String uri)
    {
        return super.isURIaClass(uri);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#isUniqueURI(java.lang.String)
     */
```

```
public boolean isUniqueURI(String uri)
{
        return super.isUniqueURI(uri);
}

/*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getRDFSlabel(java.lang.String,
     *     java.lang.String)
     */
public String getRDFSlabel(String uri, String language) throws
OntologyCorruptException
{
        return super.getRDFSlabel(uri, language);
}

/*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listRDFSlabel(java.lang.String)
```

```
     */
    public List<String> listRDFSlabel(String uri)
    {
        return super.listRDFSlabel(uri);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setRDFSlabel(java.lang.String,
     *     java.lang.String, java.lang.String)
     */
    public void setRDFSlabel(String uri, String label, String language)
    {
        super.setRDFSlabel(uri, label, language);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#deleteRDFSlabel(java.lang.String,
```

```
     *      java.lang.String)
     */
    public void deleteRDFSlabel(String uri, String language)
    {
        super.deleteRDFSlabel(uri, language);
    }

    /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#getRDFScomment(java.lang.String,
     *      java.lang.String)
     */
    public String getRDFScomment(String uri, String language) throws
OntologyCorruptException
    {
        return super.getRDFScomment(uri, language);
    }

    /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listRDFScomment(java.lang.String)
     */
   public List<String> listRDFScomment(String uri)
   {
        return super.listRDFScomment(uri);
   }

   /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#setRDFScomment(java.lang.String,
     *     java.lang.String, java.lang.String)
     */
   public void setRDFScomment(String uri, String comment, String language)
   {
        super.setRDFScomment(uri, comment, language);
   }

   /*
     * (non-Javadoc)
     *
```

```
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#deleteRDFScomment(java.lang.String,
     *      java.lang.String)
     */
   public void deleteRDFScomment(String uri, String language)
   {
         super.deleteRDFScomment(uri, language);
   }


   /*
     * (non-Javadoc)
     *
     *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listClasses(java.lang.String,
     *      java.lang.String)
     */
   public List<String> listClasses()
   {
         List<String> result = new ArrayList<String>();


         OntClass                    competencyClass                    =
this.model.getOntClass(this.getCompetencyURI());
         ExtendedIterator ei = competencyClass.listSubClasses();
```

```
        result.add(this.getCompetencyURI());
        while (ei.hasNext())
        {
            result.add(((Resource) ei.next()).getURI());
        }

        OntClass                    proficiencyClass                    =
this.model.getOntClass(this.getTraceProficiencyURI());
        ei = proficiencyClass.listSubClasses();

        result.add(this.getTraceProficiencyURI());
        while (ei.hasNext())
        {
            result.add(((Resource) ei.next()).getURI());
        }

        result.remove("http://www.w3.org/2002/07/owl#Nothing");

        return result;
    }

    /*
```

```
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listSubClassesOf(java.lang.String,
        *      java.lang.String)
        */
    public List<String> listSubClassesOf(String uri, boolean direct)
    {
            return super.listSubClassesOf(uri, direct);
    }


    /*
        * (non-Javadoc)
        *
        *                                                                    @see
org.UoR.competency.mapping.ENetInterface#listSuperClassesOf(java.lang.String,
        *      java.lang.String)
        */
    public List<String> listSuperClassesOf(String uri, boolean direct)
    {
            return super.listSuperClassesOf(uri, direct);
    }
}
```

Comparer.java

```
package org.UoR.competency.comparer;


import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;


import org.UoR.competency.mapping.ENet;


import com.hp.hpl.jena.rdf.model.Model;

import com.hp.hpl.jena.rdf.model.ModelFactory;

import com.hp.hpl.jena.rdf.model.Property;

import com.hp.hpl.jena.rdf.model.RSIterator;

import com.hp.hpl.jena.rdf.model.ReifiedStatement;

import com.hp.hpl.jena.rdf.model.Resource;

import com.hp.hpl.jena.rdf.model.Statement;

import com.hp.hpl.jena.rdf.model.StmtIterator;


/**

 * The comparer class
```

```
 *
 * @author Karsten Oster Lundqvist
 * @author k.o.lundqvist at reading "dot" ac "dot" uk
 * @author University of Reading
 * @author TRACE Leonardo project
 */
public class Comparer extends ENet
{

    public String master = null;

    public List<String> masterEntryNodes = null;

    public String comparisonProfile = null;

    // public List<String> profileEntryNodes = null;

    public static final String failureURI =
"http://www.cs.reading.ac.uk/compEvaluate#failure";

    public static final String successURI =
"http://www.cs.reading.ac.uk/compEvaluate#success";
```

```
    public static final String childrenURI =
"http://www.cs.reading.ac.uk/compEvaluate#children";

    public static final String isaURI =
"http://www.cs.reading.ac.uk/compEvaluate#isa";

    public static final String byURI = "http://www.cs.reading.ac.uk/compEvaluate#by";

    public static final String levelURI =
"http://www.cs.reading.ac.uk/compEvaluate#level";

    public static final String atlevelURI =
"http://www.cs.reading.ac.uk/compEvaluate#atLevel";

    public static final String notAchievedURI =
"http://www.cs.reading.ac.uk/compEvaluate#notAchieved";

    public static final String desiredURI =
"http://www.cs.reading.ac.uk/compEvaluate#desired";

    public static final String requiredURI =
"http://www.cs.reading.ac.uk/compEvaluate#required";
```

```
    public static final String becauseURI =
"http://www.cs.reading.ac.uk/compEvaluate#matchBecause";

    public static final String satisfiesURI =
"http://www.cs.reading.ac.uk/compEvaluate#satisfy";

    public static final String exactURI =
"http://www.cs.reading.ac.uk/compEvaluate#exactMatch";

    public static final String synonymURI =
"http://www.cs.reading.ac.uk/competencyUpper#synonymOf";

    public static final String meronymURI =
"http://www.cs.reading.ac.uk/competencyUpper#meronymOf";

    public static final String holonymURI =
"http://www.cs.reading.ac.uk/competencyUpper#holonymOf";

    public static final String hyponymURI =
"http://www.cs.reading.ac.uk/competencyUpper#hyponymOf";

    public static final String hypernymURI =
"http://www.cs.reading.ac.uk/competencyUpper#hypernymOf";
```

```
    public static final String unknownLevelURI =
"http://www.cs.reading.ac.uk/competencyUpper#unknownLevel";

    private String compEvalURI = "http://www.cs.reading.ac.uk/competencyUpper";

    public Comparer()
    {
        this.loadCompareModels();
    }

    public Comparer(ENet old)
    {
        super(old);

        this.loadCompareModels();
    }

    private void loadCompareModels()
    {
        InputStream is = getClass().getResourceAsStream("/compEval.owl");
        if (is != null)
            super.loadModel(is, "compEval");
        else
```

```
            super.loadModel("compEval", this.compEvalURI, "RDF/XML",
"file:ontology/compEval.owl");

    }

    public void SetMaster(String master) throws CompareMasterProfileException
    {
        this.master = master;
        this.masterEntryNodes = this.listEntryNodes(master);

        List<String> res = new ArrayList<String>();

        for (String str : this.masterEntryNodes)
        {
            res.addAll(this.getChildNodes(str, "trace:hasProficiency"));
        }

        if (res.size() > 0)
            throw new CompareMasterProfileException(master);
    }

    public String getMaster()
    {
```

```
        return this.master;
}

public List<String> getMasterEntryNodes()
{
        return this.masterEntryNodes;
}

private boolean compareTwoProfiles(String profileA, String profileB,
                                   Model result)
{
        // Find entry nodes
        List<String> profileEntryNodes = this.listEntryNodes(profileB);

        // Check every start node in master with the profile
        // boolean res = true;
        boolean res = true;

        for (String n : this.listEntryNodes(profileA))
        {
            Model nodeResult = ModelFactory.createDefaultModel();
            res = this.valuateNodeInProfile(n, profileEntryNodes, nodeResult) && res;
            if (nodeResult != null)
```

```
                result.add(nodeResult);

            }

            if (res)
                this.addResult(profileB, Comparer.childrenURI, Comparer.successURI,
result);
            else
                this.addResult(profileB, Comparer.childrenURI, Comparer.failureURI,
result);

            return res;
        }

    /* private boolean compareProfileWithProfileNodes(String profileA, List<String>
profileEntryNodes, Model result)
        {
            boolean res = true;

            for (String n : this.listEntryNodes(profileA))
            {
                Model nodeResult = ModelFactory.createDefaultModel();
                res = this.valuateNodeInProfile(n, profileEntryNodes, nodeResult) && res;
```

```
            if (nodeResult != null)
                result.add(nodeResult);

        }

        if (res)
            this.addResult(profileA, Comparer.childrenURI, Comparer.successURI,
result);
        else
            this.addResult(profileA, Comparer.childrenURI, Comparer.failureURI,
result);

        return res;
    }*/

    /**
     * The comparison starter
     *
     * @param profile
     *            to compare with the master
     * @param result
     *            resulting model
     * @return the result
```

```
        * @throws CompareException
        */
    public boolean compareProfileWithMaster(String profile, Model result) throws
CompareException
    {
        if (this.master == null)
            throw new CompareNoMasterException("Master Competency Mapping not
yet initialised");

        if (profile == null)
            throw new CompareNoMasterException("Profile Competency Mapping
cannot be null");
        else
            this.comparisonProfile = profile;

        // setup caches
        if (this.proficiencyMap == null)
        {
            this.setupProficiencyMap();
        }

        boolean res = this.compareTwoProfiles(this.master, profile, result);
```

```
        // reset cache
        this.proficiencyMap = null;

        return res;
    }

    /**
     * Evalute a node
     *
     * @param node
     * @param profileNodes
     *          the profile nodes that should be used in the
     *          evaluation
     * @param result
     * @return true when node is satisfied by a node in the profileNodes
     */
    public boolean valuateNodeInProfile(String node,
                                        List<String> profileNodes,
                                        Model result)
    {
        boolean res = this.valuateNodeInProfileRecurseOnProfile(node, profileNodes,
null, result);
        boolean resWithChildren = true;
```

```
        // if check node has children check them
        if (!this.listEveryChildNode(node).isEmpty())
        {
            // result has to be found in child nodes AND ind children
            resWithChildren = this.valuateChildNodes(node, profileNodes, result) &&
res;
        }

        if (res)
        {
            this.addResult(node, Comparer.isaURI, Comparer.successURI, result);
            if (!resWithChildren)
                this.addResult(node, Comparer.childrenURI, Comparer.failureURI,
result);
        }
        else
        {
            this.addResult(node, Comparer.isaURI, Comparer.failureURI, result);
            if (resWithChildren && !this.listEveryChildNode(node).isEmpty())
                this.addResult(node, Comparer.childrenURI, Comparer.successURI,
result);
        }
```

```
        return res && resWithChildren;
}

protected boolean valuateChildNodes(String node,
                                    List<String> profileNodes,
                                    Model result)
{
        boolean res;

        res = this.valuateChildNodesALL(node, profileNodes, result);
        res = this.valuateChildNodesANY(node, profileNodes, result) && res;
        res = this.valuateChildNodesIF(node, profileNodes, result) && res;

        return res;
}

protected boolean valuateChildNodesALL(String node,
                                       List<String> profileNodes,
                                       Model result)
{
        boolean res = true;

        List<String> nodes = this.listALLnodes(node);
```

```
        if (!nodes.isEmpty())
        {
            for (String n : nodes)
            {
                res = this.valuateNodeInProfile(n, profileNodes, result) && res;
            }
        }

        return res;
    }

    protected boolean valuateChildNodesANY(String node,
                                           List<String> profileNodes,
                                           Model result)
    {
        boolean res = true;

        List<String> nodes = this.listANYnodes(node);

        if (!nodes.isEmpty())
        {
            res = false;
```

```
            for (String n : nodes)
            {
                res = this.valuateNodeInProfile(n, profileNodes, result) || res;
            }
        }

        return res;
}

protected boolean valuateChildNodesIF(String node,
                                      List<String> profileNodes,
                                      Model result)
{
        // TODO NOT IMPLEMENTED YET
        return true;
}

/**
 * Evaluate a node, recursing over the the childnodes of the
 * profileNodes
 *
 * @param node
```

```
 * @param profileNodes
 * @param result
 * @return
 */
/*protected boolean valuateNodeInProfileRecurseOnProfile(
                                                        String node,
                                                        List<String> profileNodes,
                                                        Model result)
{
        return this.valuateNodeInProfileRecurseOnProfile(node, profileNodes, null,
result);
}*/


/**
    * Evaluate a node, recursing over the the childnodes of the
    * profileNodes
    *
    * @param node
    * @param profileNodes
    * @param sourceCompetency
    * @param result
    * @return true when node is satisfied by a node in the profileNodes
    */
```

```
    protected boolean valuateNodeInProfileRecurseOnProfile(
                                                        String node,
                                                        List<String> profileNodes,
                                                        String sourceCompetency,
                                                        Model result)
    {
        boolean res = false;

        String competency = this.getCompetencyOfNode(node);
        String reqProficience = this.getRequiredProficiency(node);
        String desProficience = this.getDesiredProficiency(node);

        // if competency only relies on children return result of them
        if (competency == null)
        {
            ReifiedStatement s = null;
            res = this.valuateChildNodes(node, profileNodes, result);
            if (res)
            {
                s = this.getReifiedStatement(node, Comparer.isaURI,
Comparer.successURI, result);
                this.addResult(s, Comparer.byURI, Comparer.childrenURI, result);
            }
```

```
            else
            {
                s = this.getReifiedStatement(node, Comparer.isaURI,
Comparer.failureURI, result);
                this.addResult(s, Comparer.byURI, Comparer.childrenURI, result);
            }

            return res;
        }

        if (reqProficience != null && desProficience == null)
            desProficience = reqProficience;
        if (desProficience != null && reqProficience == null)
            reqProficience = desProficience;

        if (reqProficience == null && desProficience == null)
        {
            reqProficience = this.getHasProficiencyScore(node);

            // TODO look for evidence of proficiency further up the parent
            // tree...
```

```
    if(reqProficience == null)
    {
        reqProficience = Comparer.unknownLevelURI;
    }

    desProficience = reqProficience;
}

ReifiedStatement s = null;
res = false;
for (String n : profileNodes)
{
    String competencyNode = this.getCompetencyOfNode(n);
    if (competencyNode == null)
        continue;

    ArrayList<Statement> reifMes = new ArrayList<Statement>();

    String profileProficiencyScore = null;

    if (this.checkCompetencyUsingRule(competency, competencyNode,
reifMes, result))
    {
```

```
                profileProficiencyScore = this.getProficiencyScoreComparator(n);


                if (reqProficience == Comparer.unknownLevelURI && desProficience
== Comparer.unknownLevelURI)
                {
                    res = true;
                    s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.requiredURI, result);
                    this.addResult(s, Comparer.byURI, competencyNode, result);
                    this.addResult(s, Comparer.atlevelURI, profileProficiencyScore,
result);
                }
                else
                    if (checkProficiency(desProficience, profileProficiencyScore) &&
!(desProficience == reqProficience))
                    {
                        res = true;
                        s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.desiredURI, result);
                        this.addResult(s, Comparer.byURI, competencyNode, result);
                        this.addResult(s, Comparer.atlevelURI,
profileProficiencyScore, result);
                    }
```

```
                else
                        if (checkProficiency(reqProficience, profileProficiencyScore))
                        {
                            res = true;
                            s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.requiredURI, result);
                            this.addResult(s, Comparer.byURI, competencyNode,
result);
                            this.addResult(s, Comparer.atlevelURI,
profileProficiencyScore, result);
                        }
                        else
                        {
                            s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.notAchievedURI, result);
                            this.addResult(s, Comparer.byURI, competencyNode,
result);
                            this.addResult(s, Comparer.atlevelURI,
profileProficiencyScore, result);
                        }

            for (Statement rsStmt : reifMes)
            {
```

```
                ReifiedStatement rs = this.getReifiedStatement(rsStmt, result);
                this.addResult(s, Comparer.becauseURI, rs, result);
            }

            if(sourceCompetency != null)
            {
                this.addResult(s, Comparer.byURI, sourceCompetency, result);



                ReifiedStatement rs = this.getReifiedStatement(sourceCompetency,
Comparer.satisfiesURI, competency, result);
                this.addResult(s, Comparer.becauseURI, rs, result);
            }
        }
        else
        {

            for (Statement rsStmt : reifMes)
            {
                s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.notAchievedURI, result);
                this.addResult(s, Comparer.byURI, competencyNode, result);
                ReifiedStatement rs = this.getReifiedStatement(rsStmt, result);
```

```
                this.addResult(s, Comparer.becauseURI, rs, result);
            }

            // Check Competency Profile references in comparer.
            if (!res && this.isURIaCompetencyProfile(competencyNode))
            {
                //res = this.compareTwoProfiles(competencyNode, profileB, result);
                List<String> newNodes = this.listEntryNodes(competencyNode);
                res = this.valuateNodeInProfileRecurseOnProfile(node, newNodes,
competencyNode, result);

                s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.requiredURI, result);
                this.addResult(s, Comparer.byURI, competencyNode, result);

                ReifiedStatement rs = this.getReifiedStatement(competencyNode,
Comparer.satisfiesURI, competency, result);
                    this.addResult(s, Comparer.becauseURI, rs, result);

            }
        }
```

```
        }

        // Check Competency Profiles
        if (!res && this.isURIaCompetencyProfile(competency))
        {
            boolean profileRes = this.compareTwoProfiles(competency,
this.comparisonProfile, result);

            if(profileRes)
            {
                s = this.getReifiedStatement(node, Comparer.levelURI,
Comparer.requiredURI, result);
                this.addResult(s, Comparer.byURI, competency, result);
            }

            return profileRes;
        }

        // Check children nodes of profiles if no result has been found
        if (!res)
        {
            ArrayList<String> childNodes = new ArrayList<String>();
            for (String m : profileNodes)
```

```
            {
                childNodes.addAll(this.listALLnodes(m));// this.getChildNodes(m,
                                        // "trace:all"));
                childNodes.addAll(this.listANYnodes(m));// this.getChildNodes(m,
                                        // "trace:any"));
            }

            if (childNodes.size() > 0)
                res = this.valuateNodeInProfileRecurseOnProfile(node, childNodes,
sourceCompetency, result);
        }

        return res;
    }

    private String getProficiencyScoreComparator(String n)
    {
        String profileProficiencyScore = this.getHasProficiencyScore(n);

        if (profileProficiencyScore == null)
        {
            //If node doesn't have a hasProficiency then assume that the node is a
requiredProficiency
```

```
            profileProficiencyScore = this.getRequiredProficiency(n);
        }

        if (profileProficiencyScore == null)
        {
            //If node doesn't have a requiredProficiency then assume that the node is a
desiredProficiency
            profileProficiencyScore = this.getDesiredProficiency(n);
        }

        if (profileProficiencyScore == null)
        {
            //If node doesn't have a requiredProficiency then find parent score
            List<String> parents = this.listEveryParentNode(n);

            for(String p : parents)
            {
                String dummy = this.getProficiencyScoreComparator(p);

                if(profileProficiencyScore == null)
                    profileProficiencyScore = dummy;
                else
                    if(this.checkProficiency(profileProficiencyScore, dummy))
```

```
                    profileProficiencyScore = dummy;
            }
        }

        if(profileProficiencyScore == null) profileProficiencyScore =
Comparer.unknownLevelURI;

        return profileProficiencyScore;
    }



    /**
     * The method which should be extended when intruducing new rules to the
     * comparison
     *
     * @param competency
     *          the needed competency
     * @param contender
     *          the possible competency
     * @param message
     *          statements used to finalise the comparison
     * @param result
```

```
     *                results model
     * @return true whe contender satisfies competency using rules
     */
   protected boolean checkCompetencyUsingRule(String competency,
                                              String contender,
                                              ArrayList<Statement> message,
                                              Model result)
   {
        if (competency.equals(contender))
        {
           this.addResultToArrayList(competency, Comparer.exactURI, contender,
message, result);
           return true;
        }

        return false;
   }

   protected void addResultToArrayList(String competency, String property,
                                       String contender,
                                       ArrayList<Statement> message,
                                       Model result)
   {
```

```
        Resource comp = result.createResource(competency);

        Resource cont = result.createResource(contender);

        Property prop = result.createProperty(property);

        Statement s = result.createStatement(comp, prop, cont);

        message.add(s);

}


protected void addResult(String competency, String property, String value,

                                Model result)

{

        if (result == null)

            return;


        Resource r = result.createResource(competency);

        Property p = result.createProperty(property);

        Resource v = result.createResource(value);


        r.addProperty(p, v);

}


protected void addResult(ReifiedStatement s, String property, String value,

                                Model result)

{
```

```
        if (result == null || s == null)
            return;

        Property p = result.createProperty(property);
        Resource v = result.createResource(value);

        s.addProperty(p, v);
    }

    protected void addResult(ReifiedStatement s, String property,
                             ReifiedStatement value, Model result)
    {
        if (result == null || s == null || value == null)
            return;

        Property p = result.createProperty(property);
        result.add(s, p, value);
    }

    protected ReifiedStatement getReifiedStatement(String obj, String prop,
                                                   String subj, Model result)
    {
        Resource r = result.createResource(obj);
```

```
        Property p = result.createProperty(prop);
        Resource v = result.createResource(subj);

        StmtIterator sIter = result.listStatements(r, p, v);

        if (sIter.hasNext())
        {
            Statement s = sIter.nextStatement();
            RSIterator rsIter = result.listReifiedStatements(s);

            if (!rsIter.hasNext())
                return result.createReifiedStatement(s);
            else
                return rsIter.nextRS();
        }
        else
        {
            r.addProperty(p, v);
            sIter = result.listStatements(r, p, v);
            return result.createReifiedStatement(sIter.nextStatement());
        }
    }
```

```
protected ReifiedStatement getReifiedStatement(Statement s, Model result)
{
    RSIterator rsIter = result.listReifiedStatements(s);

    if (!rsIter.hasNext())
        return result.createReifiedStatement(s);
    else
        return rsIter.nextRS();
}

private HashMap<String, ArrayList<String>> proficiencyMap = null;

protected void setupProficiencyMap()
{
    this.proficiencyMap = new HashMap<String, ArrayList<String>>();

    List<String> profs = this.listProficiencyScores();

    for (String p : profs)
    {
        ArrayList<String> list = new ArrayList<String>();
        list.add(p);
```

```
            List<String> dummy = this.listProficiencyScores();
            for (String d : dummy)
            {
                if (this.listRelatedProficiencyScores(p,
this.getRelationSynonym()).contains(d))
                    list.add(d);
                else
                    if (this.listRelatedProficiencyScores(p,
this.getRelationMeronym()).contains(d))
                        list.add(d);
            }

            this.proficiencyMap.put(p, list);
        }
    }

    protected boolean checkProficiency(String desired, String has)
    {
        if (has == null)
            return false;
        if (this.proficiencyMap.get(desired).contains(has))
            return true;
        else
```

```
            return false;
    }

    public void saveResult(Model result, String filename)
    {
        try
        {
            FileOutputStream f = new FileOutputStream(filename);
            result.write(f);
        } catch (FileNotFoundException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```