A comparison of UML and OWL in the travel domain

Jennifer Sampson

Department of Computer and Information Science,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract. Recent research has focused towards determining the efficacy of using UML as an ontology modelling language [6, 7, 8, 12]. In this paper we compare the use of UML with the Web Ontology Language – OWL for modelling the travel domain. One conclusion resulting from this study is that OWL and UML were devised with different motivations, and for supporting different types of application domains. Owl benefits from a solid theoretical basis in description logic, which means it has a well defined semantics, formal properties are well understood, and there are known reasoning algorithms and implementations of the language. In addition to the model comparison we propose an ontology quality framework. The quality framework is one step towards defining quality measures for ontology modelling.

1 Introduction

Ontology can be described as 'what there is', and relies on the use of specific terms to construct a description of reality. In the context of the Semantic Web we think about ontology as referring to the consensual and formal description of shared concepts in a domain. Ontologies are said to be a way to aid communication between humans and machines and also between machines for agent communication. A number of recommendations have been proposed for languages that support the creation of ontologies for such communication and understanding, however, now the research question is to determine whether people and organizations will actually use these languages to help achieve the 'Semantic Web' vision. In addition, the problem is not so much associated with creating ontologies but with providing support for reasoning and mapping between ontologies. The challenge is to create a language which is formal enough to support machine to machine processing and also one which is simple to use and understand.

2 UML

The Unified Modelling Language (UML) is a standard language for writing software blueprints [5] whose constructs originate from object oriented programming. The OMG [15] state that UML is a language for documenting: software systems, modelling business systems and other non-software systems. The language is for

Sampson J. (2004).
A comparison of UML and OWL in the travel domain.
In *Proceedings of the 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure*, pages 36-48
DOI: 10.5220/0002678300360048
Copyright © SciTePress

specifying and constructing object-oriented systems, and also for visualising and documenting such systems. The three building blocks of the language are: things, relationships and diagrams. Booch, Rumbaugh and Jacobson [5] describe the three building blocks, firstly, ‘things’ are the abstractions that are first-class citizens in a model; relationships link the ‘things’ together and the diagrams group ‘interesting’ collections of things. In UML ‘things’ may be either: structural, behavioural, grouping or annotational. The UML defines several graphical diagrams such as: the use case diagram, the class diagram, behaviour diagrams and implementation diagrams. However, in this particular exercise we are mainly concerned with the use of the UML class diagram for modelling a travel agency domain.

Baclawski, et al. [3] have undertaken research to use UML class diagrams to develop and display complex DAML+OIL ontologies. They also researched the differences between UML and DAML+OIL and provide initial suggestions on how to overcome the differences. While there are similarities between UML and knowledge representation languages, the fundamental problem with extending UML for ontology development, is due to the important distinction between knowledge representation approaches and object-oriented approaches: that of monotocity [3]. They comment, “Both RDF and DAML+OIL are monotonic: asserting a new fact can never cause a previously known fact to become false. By contrast, UML and other OO systems are typically not monotonic. There are many forms of nonmonotonic logic, but the one that is closest to UML and OO systems is a logic that assumes a closed world” [3]. While Baclawski, et al. [3] attempted to provide rules for translating UML concepts to DAML+OIL concepts they came to the conclusion that some of the concepts are significantly incompatible. For example, the ‘property’ concept of DAML+OIL although similar to the UML ‘association’ concept, they claim it cannot be mapped easily. Indeed, they suggest, “this is the main obstacle to using UML (and UML tools) for DAML-based ontology development” [3]. However they do propose extensions to the UML metamodel, with the motivation of making UML more acceptable to the knowledge representation community as the “preferred graphical notation for KR languages” [3].

3 What is OWL?

Owl is one such Web ontology language developed by the Web-Ontology Working Group whose main goal was to provide a semantic markup language for disseminating and sharing ontologies on the World Wide Web. There are a wide variety of languages for explicit specification, Owl is a description logic based ontology language. The Web Ontology Group [20] explicitly say that OWL “is designed for use by applications that need to process the content of information instead of just presenting information to humans”. They go further and claim that “OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics” [20].
OWL comes in three different versions to allow for the different communities of implementers and users. In effect, this means the Web Ontology group have provided three increasingly expressive sublanguages; OWL Lite, OWL DL and OWL Full. As

the name implies, OWL Lite, is a ‘trimmed’ down version of OWL DL and OWL Full, mostly to support those users who require a language to construct classification hierarchies and to define simple constraints. Whereas OWL DL (where DL stands for Description Logic) is to support users who “want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time)” [20]. OWL DL is a ‘fuller style’ of using OWL as it contains all OWL language constructs, but the use of these constructs is restricted through Descriptive Logic. The WebOnto group designed OWL DL to “support the existing Description Logic business segment and to provide a language subset that has desirable computational properties for reasoning systems” [23].

When the limitation of the language constructs is relaxed a full version of the language – OWL Full is presented. According to the Web Ontology group OWL Full is to make available features which may be of use to many database and knowledge representation systems, but which violate the constraints of Description Logic reasoners. In the description of OWL the working group have described a high-level, abstract syntax for both OWL Lite and OWL DL. In addition they use two formal semantics for OWL. First a model-theoretic semantics to provide formal ‘meaning’ for OWL ontologies. Model theoretic semantics is to define the semantics of the concepts in terms of a set theoretic interpretation of the concepts in logic. Second they extend RDF semantics to provide formal semantics for OWL ontologies depicted as RDF graphs (this is for OWL Full).

3.1 Historical background - RDF and DAML+OIL

OWL is one of the recommendations from the W3C towards making the notion of the ‘Semantic Web’ feasible. OWL builds on earlier languages for enabling the Semantic Web, the progression went from XML, XML Schema, RDF, RDF Schema to OWL. The Web Ontology Working Group [20] comment that,

“The first level above RDF required for the Semantic Web is an ontology language what can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema”.

OWL has been developed as a vocabulary extension of RDF (the Resource Description Framework). The purpose of RDF is to produce a language for the exchange of machine-understandable information using the Web [21]. RDF is mostly for capturing metadata about Web resources. RDF is a graphical formalism using XML syntax and semantics, for representing metadata and describing the semantics of information in a machine-processable way. RDFS was proposed to extend RDF with schema vocabulary (e.g. class, property, type, subclassOf etc) to add meaning to the modelling primitives. However, one problem with RDFS is that it lacks a clear, formal semantics. In addition it relies on DAML+OIL to ‘give’ it the semantics, which makes RDFS not such a satisfactory schema layer Semantic Web language [16]. Moreover, the progression from RDF to Owl came by the way of two other languages, OIL and DAML. DAML and OIL were merged to extend the description logic subset of RDF. The W3C Web Ontology Working Group have developed OWL as a revision of the DAML+OIL web ontology language. They claim that OWL is essentially stable as a web ontology language. Nonetheless they recognize that the modelling primitives of OWL are still informally stated.

4 Travel domain description

The domain we have modelled is for supporting and providing information for customers regarding travel and accommodation options between cities [1]. This domain was chosen for pedagogical reasons, in future research we will be comparing different ontology languages in the medical domain. At this stage of the research the purpose was to establish a method for comparing ontology languages.

The following sentences were derived from the natural language description of the scenario.

```
A Client may request many Travel Itineraries. An
Itinerary is for one Client.

An Itinerary comprises one or more Travel Legs. One
Travel Leg is for one client Itinerary.

An Itinerary may comprise Accommodation Detail. One
Accommodation Detail is for one client Itinerary.

A Travel Leg uses one Transport Means. One Transport
Means is required for many Travel Legs.

A Travel Leg follows one Travel Route. One Travel Route
is required for a Travel Leg.

Underground (Bus, Aircraft, Vehicle, Train, Ferry,
Ship) is a Transport Means.

A City may have many Tourist Attractions. A Tourist
Attraction is located in one City.

A City may have many Airports. An Airport is located in
one City.

A City is located in one Country.  A Country has many
Cities.

A Country is located in one Continent. A Continent
contains many Countries.

A Travel Route involves one arrival City. An arrival
City is part of many Travel Routes.

A Travel Route involves one departure City. A departure
City is part of many Travel Routes.
```

```
A Travel Route is provided through a Transport Company
Schedule. A Transport Company Schedule provides many
Travel Routes.

A Transport Company supplies one or more Transport
Company Schedules. A Transport Company Schedule is
provided by one Transport Company.

A City provides many Accommodation types. An
Accommodation type is located in many cities.

A B&B is a type of Accommodation

A Hotel is a type of Accommodation

A Hotel is either a Chain or Privately Owned

A Hotel has many rooms (at least one). A room belongs
to one Hotel.

A Room is classified as having one facility type group.
A Facility Type is provided by many Rooms.
```

4.1 UML model in the travel domain

The UML class diagram for the scenario travel domain is shown in Figure 1. We recognise the model represents our interpretation of the scenario and is not necessarily the only correct representation of the travel domain. In addition the model has been constructed using the natural language description of [1] for pedagogical purposes. The domain serves as a basis for comparing two languages for ontology modelling, one using UML and the second using OWL. Due to the size and complexity of the travel domain, property details for some classes are not illustrated at Figure 1. We made several assumptions when completing the modelling task. First, a travel leg involves only one type of transport, for example a plane, train or car. Second, a travel leg is one ‘leg’ of an entire journey. Third an itinerary is made up of at least one travel leg. The travel domain was created using a UML class diagram to describe five clusters of information within the domain. These are: information about the client and the travel plans for that client, modes of transportation, travel schedules and travel routes, cities and attractions, and accommodation options.

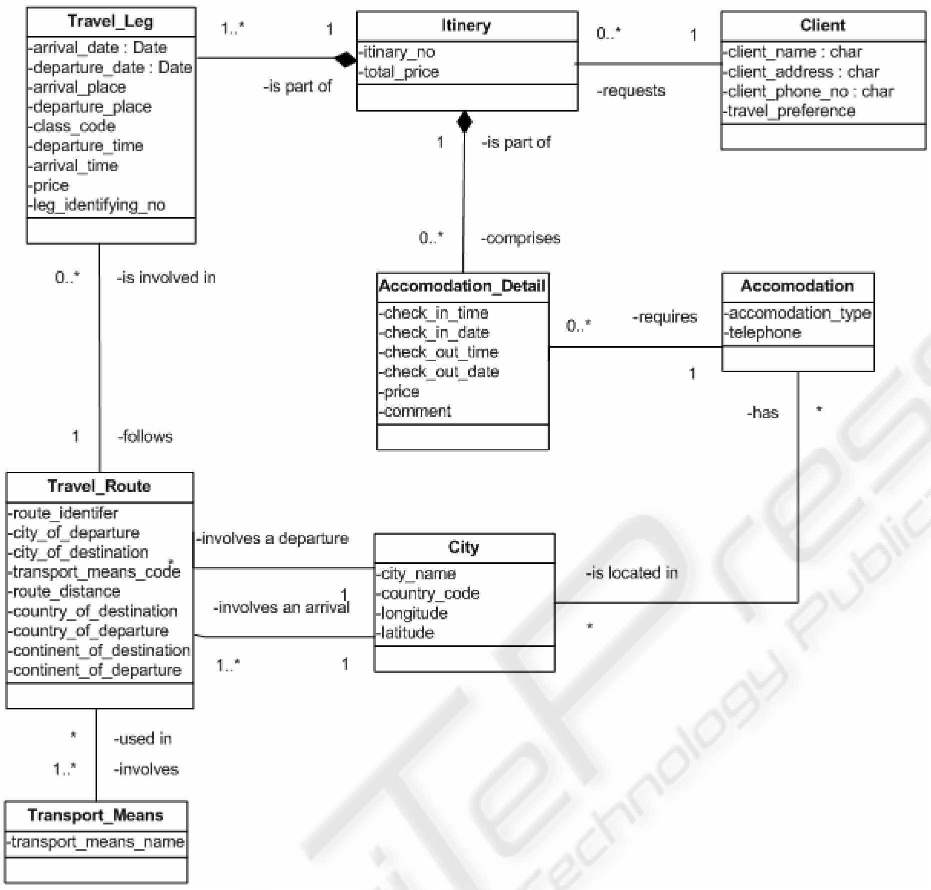


Fig. 1. One portion of the UML Class diagram

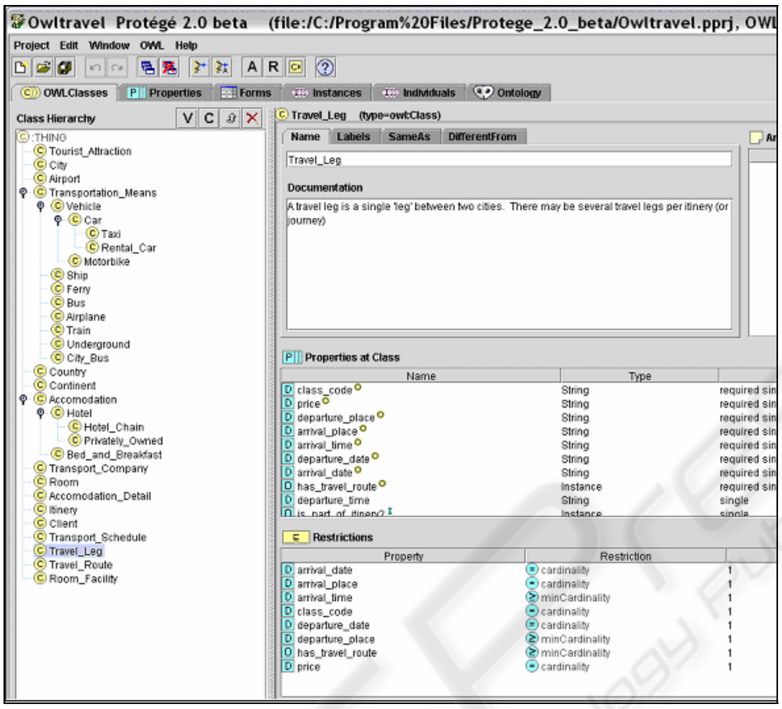
4.2 OWL Model

We used the Protégé OWL plugin [18] as a tool for developing the OWL travel model. Protégé is an integrated software tool used to develop knowledge based systems.

Example Object property and Inverse properties in OWL/RDF source code

```
  <owl:ObjectProperty rdf:ID="has_Itinerary">
    <rdfs:range rdf:resource="file:/C:/ Protege_2.0_beta /Owltravel.owl#Itinerary"/>
    <owl:inverseOf>
      <owl:ObjectProperty
             rdf:about="file:/C:/ Protege_2.0_beta Owltravel.owl#has_client"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="file:/C:/Protege_2.0_beta /Owltravel.owl#Client"/>
  </owl:ObjectProperty>
```

The class taxonomy is shown at Figure 2.


Owltravel Protégé 2.0 beta (file:/C:/Program%20Files/Protege_2.0_beta/Owltravel.pprj, OWL
Project Edit Window OWL Help
A R
OWLClasses
Properties
Forms
Instances
Individuals
Ontology
Class Hierarchy
V C
Name
Labels
SameAs
DifferentFrom
An

```
:THING
  Tourist_Attraction
  City
  Airport
  Transportation_Means
    Vehicle
      Car
        Taxi
        Rental_Car
      Motorbike
    Ship
    Ferry
    Bus
    Airplane
    Train
    Underground
    City_Bus
  Country
  Continent
  Accomodation
    Hotel
      Hotel_Chain
      Privately_Owned
    Bed_and_Breakfast
  Transport_Company
  Room
  Accomodation_Detail
  Itinery
  Client
  Transport_Schedule
  Travel_Leg
  Travel_Route
  Room_Facility
```

Travel_Leg (type=owl:Class)

Travel_Leg

Documentation

A travel leg is a single 'leg' between two cities. There may be several travel legs per itinery (or journey)

Properties at Class

Name	Type	
class_code	String	required sin
price	String	required sin
departure_place	String	required sin
arrival_place	String	required sin
arrival_time	String	required sin
departure_date	String	required sin
arrival_date	String	required sin
has_travel_route	Instance	required sin
departure_time	String	single
is_part_of_itinery?	Instance	single

Restrictions

Property	Restriction	
arrival_date	cardinality	1
arrival_place	cardinality	1
arrival_time	minCardinality	1
class_code	cardinality	1
departure_date	cardinality	1
departure_place	minCardinality	1
has_travel_route	minCardinality	1
price	cardinality	1

Fig. 2. OWL taxonomy in Protégé

5 Model comparison

This section describes the quantitative analysis which was performed when comparing the OWL and UML models. The quantitative analysis is not an exhaustive measure of the difference between UML and OWL concepts, instead the metrics provide a simple comparison between model elements in the generated UML and OWL models. Nonetheless we believe the quantitative method used may be a useful pedagogical tool for model comparison.

5.1 Concepts occurring in both models.

There are sixteen classes Client, Itinerary, Travel_Leg, Accomodation_Detail, City, Country, Continent, Airport, Travel_Route, Transport_Company_Schedule, Transport_Company, Transport_Means, Accommodation, Room, Room_Facility, Tourist_Attraction in the UML and OWL models. There are fifteen subclasses: Underground, Bus, Train, Vehicle, Airplane, Ferry, Ship, Taxi, Rental_Car, Car, Hotel_Chain, Hotel, Privately_Owned, B&B in both models. In addition, a total of eighty-eight basic properties exist within these classes. There are fifteen generalisations in the UML model and fifteen subclasses in the OWL model.

5.2 Concepts occurring in the control model not specified in the observed model.

The UML class model is obviously different from a pragmatic perspective as it is graphical representation of the natural language description, whereas the OWL model as presented in Protégé uses a tree structure. The OWL/RDF source code can also be generated within the Protégé environment. The constructs observed in the UML model which do not appear in the same form in the OWL model are the relationships (associations) between classes. The domain, range and cardinality are declared using the Association construct in UML. In addition we have added three constraints using OCL and pseudocode to the Class definitions in the UML model.

5.3 Concepts occurring in the observed model not specified in the control model.

The OWL ontology model contains slots (properties) within each class which represent the relationships between classes, for example: has_city_of_destination and has_city of arrival in the Travel_Route Class. There are twenty-six properties of this type in the OWL model, eight of these object properties are inverse properties. The domain, range and cardinality are declared using object properties in OWL. In addition, instances have been added to the OWL ontology model (Twenty-seven instances). This is not to say that the UML language does not support instances, but it is not appropriate to show instances of a travel execution in a class diagram.

5.4 Observation Instrument: Model Element Comparison

We created four initially 0-valued matrices, where observation and control model elements formed the rows and columns. The first matrix specifies all model elements in the UML model. The second matrix specifies all model elements in the OWL model. The third matrix was to measure the correctly represented observed data. To do this we calculated where for each element in the UML model there was a corresponding element in the OWL model, where the element in OWL model described the same referent (phenomena), that is, denoted with same domain term (symbol). In other words, we checked the occurrences of the UML model terms that are used in the OWL model to denote referents ("things/phenomena") as considered relevant to the domain, regardless of any language symbols that are used to specify it. This matrix concerns the correspondence between the elements both in the control and in the observed model. The results of the analysis are presented at Table 1.

Table 1. Precision, Recall and Fallacy Measures for Correctly Represented Model Elements

PrecisionRep = Correctly Represented / Represented	1
RecallRep = Correctly Represented / Existing Correct	0.919355
FallacyRep=(Represented-Correctly Represented)/Existing Correct	0

The intuition is that precisionRep measures the correctness of representation for the OWL language model (1), recallRep measures the OWL language model's capability to represent (0.92), and fallacyRep measures the failure of the OWL language model representation in the context of the travel domain description (0). These results are as we expected because the same classes and properties are defined in both models. However, these results do not show whether the model elements are specified in the same way in each model. A fourth matrix was created to measure correctly specified observed data. We calculated where for each element in the OWL model there was a corresponding element in UML where the element in UML and the OWL model describe the same referent (phenomena), and in the same way, that is, uses equal language constructs. In other words, we checked the occurrences of the OWL model terms that are used in the UML model to denote referents ("things/phenomena") as considered relevant to the domain, when expressed by same or synonymous language symbols in the UML language. Likewise, this matrix is non-trivial, since it concerns the correspondence between the elements both in the control and in the observed model. The results of the analysis are presented at Table 2.

Table 2. Precision, Recall and Fallacy for Correctly Specified Model Elements

PrecisionRep = Correctly Specified / Represented	0.608187
RecallRep = Correctly Specified / Existing Correct	0.55914
FallacyRep = (Represented - Correctly Specified) / Existing Correct	0.360215

Using the measures for correctly specified observed data, we have measured the stricter, yet also more realistic metrics for the OWL language model, such as correctness of specification (0.6), capability to specify (0.56), and failure to specify (0.36) in the context of the travel domain. These results reflect the difference in how relationships (incl. domain, range and cardinality) are established in OWL compared to UML. UML associations allow the specification of business rules between Classes, for example: A City may have many Tourist Attractions. A Tourist Attraction is located in one City. In OWL, to model a similar notion, we use properties (usually prefixed with has_) for example has_location and has_tourist_attraction. This is an example of where we are modelling the same referent using different language constructs. Only where we set-up inverse relationships in OWL are we modelling the similar two-way business rules (associations) in UML. In the UML class diagram we use Associations between classes to model relationships whereas in OWL we use functional properties, object properties and inverseOf constructs. Associations in UML also provide the domain and range for associations between objects. In OWL we use rdfs:domain and rdfs:range to specify the domain and range of owl:ObjectProperty. We use owl:DatatypeProperty for defining properties of owl:class.
The matrix results only provide a simple comparison between model elements in UML and OWL, which were useful as a pedagogical aid. The next section describes a more general framework for evaluating ontology models.

6 Quality Evaluation

Despite the considerable level of interest in the creation and use of ontologies, there have been few attempts to define a systematic framework within which their quality can be evaluated. Research has been undertaken to establish frameworks for conceptual modelling [10,11,17] because conceptual modelling is similar to ontology modelling we can extend these frameworks to be useful for determining ontology model quality. Atkins [2] addressed the need for quality in the context of conceptual data modelling. She developed a framework based on the work of [10,13]. The framework was used in her research for assessing the final products of the INTECoM development [2]. Her adapted quality framework is shown at Figure 3. The INTECoM framework is useful for assessing the quality of ontology models because it introduces the importance of measuring the procedural quality of the model as well as the quality of the language used. Introducing procedural quality is also one step towards defining rigor in ontology construction. As yet there are no formal steps for creating ontology models. However there are several guidelines which are commonly cited as a basis for ontology development [9, 14]. In the paper 'Ontology 101', Noy and McGuinness provide a useful knowledge engineering methodology to ontology construction which consists of the following steps: defining the classes, arranging the classes in a taxonomic hierarchy, defining slots and allowed values, and filling in the values for slots for instances. The first step towards measuring procedural quality would be (at a minimum) ensuring the steps and guidelines of [14] are followed.

As [10] suggests, a model is a representation of statements taken from a domain and expressed using some form of formal grammar or language. However, their framework only illustrates a static position and does not recognise the dynamics of model construction. Thus Atkins [2] includes a new concept of process. This reflects that not only are the constructs of a language used to represent the information within a domain but that some form of process (or method) is also required to build any particular ontology model. While Lindland et al [11] define the links of semantics, relating the model to the domain, and syntax, relating the model to the modelling language, the new link of procedure relates the model to the process by which it is constructed [2]. This gives rise to the procedural quality shown on the diagram at Figure 2. The overall goal of procedural quality is that the process by which model construction had occurred is explicit and has been followed appropriately.

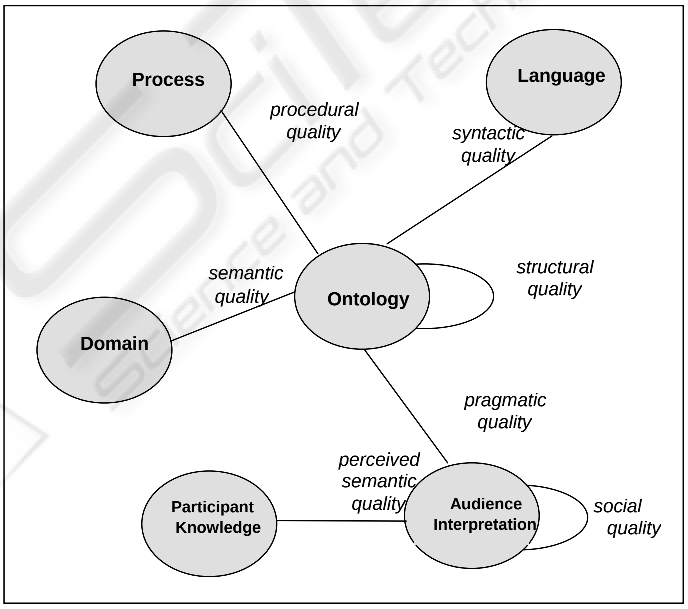


Fig. 3. Quality framework for ontology modelling (adapted from [2])

The above quality framework is useful because it not only addresses the model quality in terms of syntactic, semantic and pragmatic quality but also emphasises the need for procedural quality. Procedural correctness requires that the process by which the model is constructed is explicitly laid down and that the relevant activities have been completed satisfactorily [2]. Figure 3 presents a quality framework adapted from [2] for use in determining ontology model quality.

"Perceived Semantic Quality" can be measured through checking semantic completeness and semantic correctness. The means to achieve semantic completeness involves a subjective user confirmation that their view is complete. For example when using Owl all business rules should be represented using properties and inverse relations. In a subjective manner we can say that the OWL model is complete according to the natural language description provided. However, we have not captured all properties that would appear in the subclasses for transport means and accommodation. The means to achieve semantic correctness requires that each class has a complete set of correct and relevant examples. Each instance created in OWL should be verified by subject matter experts. Perceived semantic correctness can not be measured formally. However, we can create instances using Protégé which could then be verified by a domain expert.

Pragmatic quality can be measured through the subjective measure of pragmatic correctness, that is the model is a) understandable and b) understood. We suggest the use RML (Referent Modelling Language) developed by [19] to provide a graphical notation for OWL. The tree structure in OWL is understandable, however the RDF source code as generated by Protégé is not so clear. There is a potential for the use of RML to provide a graphical representation of the OWL ontology, because we can convert an RML model into Description Logics with respect to OWL semantics and syntax. We could then convert the description logic into statements which could be verified by domain experts.

Procedural quality is measured through procedural correctness. Procedural correctness can be facilitated through the research of [14] on ontology construction. The primary steps are: class definition, formation of the taxonomic hierarchy, slot definition and allowed values, and instance creation.

Syntactic correctness requires a syntactic check of all model constructs. OWL follows an abstract syntax for OWL Lite and OWL DL [22] and exchange syntax for transformation to RDF graphs. The Protégé ontology editor tool enforces the abstract syntax of OWL Full. In our case the travel ontology has the classification of OWL Full through the OWL syntax checker.

Semantic correctness may be achieved through consistency checking which requires the language to be a formal logical language. OWL uses model-theoretic semantics for OWL ontologies written in the abstract syntax. A model-theoretic semantics in the form of an extension to the RDF semantics provides a formal meaning for OWL ontologies as RDF graphs (OWL Full). However, because OWL is based on RDF Schema, which is not a satisfactory schema-layer-Semantic Web language [16], OWL must provide the semantics for RDFS. This is a problem according to [16] because RDFS does not distinguish the modelling information in the ontology level with that in the language.

Social Agreement may be facilitated through a subjective rating by the designer that any conflict between users of the model is based in their individual perception of the model rather than in its structure.

Structural quality goals - simplicity and flexibility are to establish that the ontology model contains the minimum number of classes and properties necessary to represent the required statements. Whether to model a specific distinction (such as train, bus, or airplane transport) as a property value or as a set of classes depends on the scope of the domain. "Subclasses of a class usually (1) have additional properties that the superclass does not have, or (2) restrictions different from those of the superclass, or (3) participate in different relationships than the superclasses" [14]. With respect to the natural language description provided the OWL travel ontology model contains the minimum number of classes and properties necessary to represent the required statements, however the 'real' complexity of such a domain is not captured in the model.

7 Conclusion

The OWL model was developed subsequent to the UML model; we need to consider the possibility of different results if the OWL model had been developed first. The way the analysis was performed also presumes there is only one 'correct' way to represent the Travel Domain. There are many implications resulting from this assumption, first just because the observed model uses different model elements to represent the same referent does not mean that one language is more 'correct' than the other. In this case the most meaningful measure for comparison is a syntactic one, because it describes only missing or different constructs of the language and not differences in meaning. However, like [16] we believe that OWL and UML have different motivations and application domains. In a sense this means we are comparing "apples with oranges". The strength of OWL is that it has a strong foundation in description logic, this means that the modelling primitives in OWL can be mapped onto description logics and hence support the provision of reasoning. Whereas, in UML, reasoning over the object constraint language (OCL) is still under research. The results from the matrices did not include the ability (or not) to add instances. Indeed, in this respect the comparison would be meaningless because UML class diagrams are not typically for representing instances.

The quality framework presented in this paper, represents a starting point for establishing measures for achieving quality in ontology modelling. Future research may involve the comparison of more than one semantic-web enabling language for modelling such a domain. In addition it would be useful to compare the results found in this study with others using different domains.

Acknowledgements

This work is partially supported by the Norwegian Research Foundation in the framework of Information and Communication Technology (IKT-2010) program – the ADIS project.

References

1. AIFB Case.: NL description of the traveling domain at URL: http://km.aifb.uni-karlsruhe.de/eon2002 (2002)
2. Atkins, C.: INTECoM: An integrated conceptual data modelling framework, Unpublished Thesis, Department of Information Systems, Massey University, New Zealand (2000)
3. Baclawski, K., Kokar, M.K., Kogut, P.A., Hart, L., Smith, J., and Letkowski, J.: Extending the Unified Modelling Language for ontology development, Software Systems Model, Special Issue UML 2002, vol 1 (2002) 1–15
4. Berners-Lee, T. Hendler. J, and Lassila, O.: The Semantic Web, Scientific American. (2001)
5. Booch. G, Rumbaugh. G, and Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley (1999)
6. Chang. W.W.: A Discussion of the Relationship Between RDF-Schema and UML. W3C Note, URL http://www.w3.org/TR/NOTE-rdf-uml/ Aug (1998)
7. Cranefield, S. and Purvis, M.: UML as an ontology modelling language. In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99). (1999)
8. Cranefield, S.: UML and the Semantic Web, Feb. 2001. ISSN 1172-6024. Discussion paper. (2001)
9. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2), (1993) 199-220.
10. Krogstie, J., Lindland O.I and Sindre, G.: Towards a deeper understanding of Quality in Requirements Engineering, Proceedings of 7th CAiSE, Jyvaskyla, Finland (1995)
11. Lindland, O.I., Sindre, G and Sølvberg, A.: Understanding Quality in Conceptual Modelling, IEEE Software, March (1994) 42-49.
12. Melnik, S.: Representing UML in RDF. URL http://www-db.stanford.edu/˜melnik/rdf/uml/ (2000)
13. Moody, D. and Shanks, G.: What Makes a Good Data Model? A Framework for Evaluating and Improving the Quality of Entity Relationship Models, The Australian Computer Journal, 30(3) (1998) 97-110
14. Noy, N. and McGuinness, D.L.: “Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March (2001)
15. OMG Inc.: OMG-Unified Modeling Language Specification, v1.5. An Adopted Formal Specification of the Object Management Group, (2003)
16. Pan J. and Horrocks, I.: Metamodeling Architecture of Web Ontology Languages. In: Cruz I. F. et al. (Eds.). The Emerging Semantic Web. IOS Press. (2002)
17. Pohl, K.: The three dimensions of requirements engineering: A Framework and its applications, Information Systems, 19(3) (1994) 243-258.
18. Protégé.: The Protege Project.http://protege.stanford.edu (2000)
19. Sølvberg, A.: Data and what they refer to. In: P.P.Chen et al.(eds.): Conceptual Modeling, Lecture Notes in Computer Science, Springer Verlag, (1999) 211-226
20. Web Ontology Working Group.: OWL Web Ontology Language Overview W3C CandidateRecommendation 18 August 2003, http://www.w3.org/TR/2003/CR-owl-features-20030818/ (2003a)
21. Web Ontology Working Group.: RDF Vocabulary Description Language 1.0: RDF Schema (2003b)
22. Web Ontology Working Group Working Draft, http://www.w3.org/TR/2003/WD-rdf-schema-20030123/ 23 January (2003)
23. Web Ontology Working Group.: OWL Web Ontology Language Reference W3C Candidate Recommendation 18 August 2003, http://www.w3.org/TR/2003/CR-owl-ref-20030818/ (2003c)
24. Web Ontology Working Group.: OWL Web Ontology Semantics and Abstract Syntax. W3C Candidate Recommendation 18 August 2003, http://www.w3.org/TR/2003/CR-owl-semantics-20030818/ (2003d)