# Application of Ontology Engineering in Satellite Network Simulation

Lin Qi
Department of Space and Command
Academy of Equipment
Beijing, China
Linqi_98@163.co

Chen Yong
Department of Space and Command
Academy of Equipment
Beijing, China
chenyong @163.com

***Abstract*- Existed simulation works are always too dependent on a kind of particular techniques or tools to perform horizontal comparison and integration. There is no formal description for the simulation objects which leads to the lack of semantic support in the simulation. In order to guarantee the reusability and interoperability of heterogeneous models and services in satellite network simulation and provide verification support, a method based on ontology engineering is proposed. Based on the concept of simulation verification with semantic support, the purpose and requirements of applying ontology engineering in satellite network simulation are identified. A test is performed as to a specific simulation task, and the results show that, besides the semantic support, the former method can provide significant improvement in efficiency, which is over 200% in comparison with the latter one. It is concluded that it is quite feasible to apply ontology in the area of satellite network simulation.**

***Keywords-component; ontology engineering; satellite network; simulation; semantic verification***

## I. INTRODUCTION

Because of the specific attributes of satellite network and particularity of space environment, research on satellite network simulation gains more and more attention recently. Currently, the achievements in the areas of constellation configuration design, network topology, inter-satellite links (ISL), and route strategies in satellite network are relatively abundant [1,2,3], but these works are always heavily dependent on a kind of particular techniques or tools so that it is difficult to perform horizontal comparison and integration. Most of all, there is no formal description for the simulation objects, which leads to the lack of semantic support in the simulation. In order to solve these problems, it is required to apply ontology engineering to provide correctness verification of simulation, i.e. to prove the consistency of simulation project itself, and furthermore to guarantee the reusability and interoperability of heterogeneous models and services in satellite network simulation.

## II. VERIFICATION WITH SEMANTIC SUPPORT

There exists disbelief in simulation verification all the time, and one of the most important reasons is the lack of correctness confirmation of simulation itself. Similar to software testing, simulation is considered as only an experimental method to prove something is wrong rather than to prove it is right. Pure simulation lacks semantic support, and it can only verify a subset in the application domain. If something wrong is discovered, we can conclude that the simulation goals or models really have problems. But even if simulation is performed as expected, there is still no guarantee on the credibility of simulation procedure itself. Only when the simulated domain is proven to be consistent and correct, can simulation provide "real verification".

Although it is impossible to prove correctness by means of software testing, several proving methodologies have been proposed long ago. For example, correctness proving based on formal semantic introduced by Hoare [4] is to perform logic deduction and reasoning in view of system semantic, which can prevent unilateral result. Similarly, as to the lack of semantic support in simulation, we can also use description logic and ontology which are based on FOL (first-order logic) to establish a provable simulation framework, so as to provide proof for semantic consistency at higher level. When the premise is guaranteed that simulation hierarchy is consistent, we can perform qualified analysis by simulation experiment on lower level to provide fine-grained verification.

Ontology can provide a bridge to build a sharable architecture for the problem domain without concept ambiguity, which is the key to support the semantic consistency in simulation. In addition, it is necessary to deduce in descriptive knowledge base according to a set of specific rules. Ontology and description logic are ideal to provide such a logic descriptive method. As a result, it is not only necessary but also feasible to apply ontology engineering to provide formal verification for simulation application.

## III. CONCEPTS OF ONTOLOGY ENGINEERING

Ontology is an explicit description of a domain, including concepts, properties and attributes of concepts, constraints on properties and attributes, and often includes individuals. With ontology, we can define a common vocabulary and a shared understanding of a domain [5]. Ontology engineering means defining concepts in the domain (classes), arranging the concepts in a hierarchy (subclass-superclass hierarchy), defining which attributes and properties classes can have and constraints on their values, and defining individuals and filling in values. Analysis and evaluation is one of main research issues in ontology engineering.

Published by Atlantis Press, Paris, France.

The aim to develop ontology is to share common understanding of the structure of information among people and among software agents, especially to enable reuse of domain knowledge and to introduce standards to allow interoperability. With ontology, it will be easier to change domain assumptions and re-use domain and operational knowledge separately.

## IV. APPLICATION OF ONTOLOGY ENGINEERING

### A. Operational View

By applying ontology engineering, the simulation framework can set up an operational view for any simulation task.

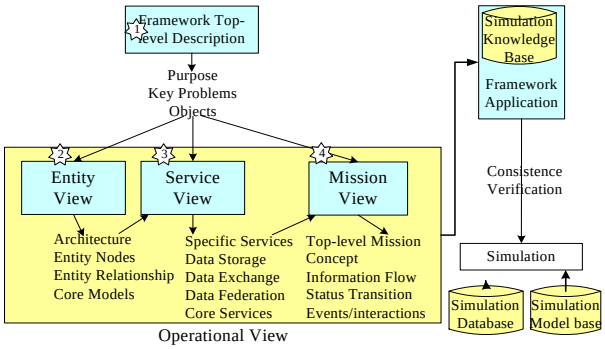


Figure 1. Operational view

The operational view is the description of the entities and services involved in a mission performed in satellite network, and can be divided into 3 sub-views. In the entity view, the entities are organized in a layered architecture, and it is also required to provide description of relationship and constraints among entity nodes. Based on the entity view, the service view put focus on the simulation-specific services and a set of supporting services. Simulation-specific services are types of services in the areas of constellation configuration, communication links, route protocols, topology, network management, and network security in satellite network, which can be invoked in simulation system. Supporting services includes data storage, data exchange, data federation, and core service to provide infrastructure for simulation system. On the basis of the entity and service views, the purpose of the mission view is to provide top-level mission concept, the description of information flow and events/interactions, and status transition array. After performing consistency verification by using reasoning engine, we can really develop a simulation application to implement simulation analysis.

### B. Ontology-development related tools

We use Protégé-3.2 as the ontology-development tool, which supports a rich knowledge model and is open-source and freely available [6]. In addition, in order to implement consistency deduction, a practical OWL-DL reasoner named Pellet is used as DIG (Description Logic Implementers Group) engine. According to the visualization requirements, visualization plug-in (graphviz) is also configured in Protégé. These tools can be used as follows:

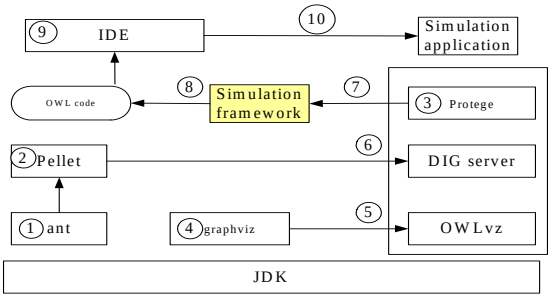


Figure 2. Ontology-development related tools

① Install and configure ant after installing JDK;
② Build Pellet with ant;
③ Install Protégé;
④ Install graphviz to provide visual presentation;
⑤Configure the OWLvz tab in Protégé with graphviz;
⑥Configure Pellet as the reasoning engine of Protégé;
⑦ Develop simulation framework in Protégé;
⑧ Provide consistency proving with simulation framework;
⑨ Map the OWL code generated by simulation framework to IDE developing environment;
⑩ Develop simulation application.

### C. Developing Ontology In Satellite Network

#### 1) Define Classes and the Class Hierarchy

Class is a concept in the domain and is a collection of elements with similar properties. Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy) and a class hierarchy is usually an IS-A hierarchy. In satellite network simulation, we can establish a class hierarchy according to the constitution and latitude of satellite entities, including satellite-network, constellation, single node (satellite and station), and single component (on-board or grounded devices) at the top level, which can be divided into lower levels furthermore (shown in Fig. 3).

#### 2) Disjoint Classes

Classes are disjoint if they cannot have common instances, and disjoint classes cannot have any common subclasses either. For example, MEO-Satellite, GEO-Satellite and LEO-Satellite are disjoint.

#### 3) Define Properties of Classes – Slots

Slots in a class definition describe attributes of instances of the class and relations to other instances. There are different types of properties, including “intrinsic” properties (orbit planes of constellation), “extrinsic” properties (name of satellite), parts (payload on satellite), and relations among objects (owner of a on-board device (satellite)). In addition, it is necessary to define property constraints (facets), which describe or limit the set of possible values for a slot. For example, IntraPlaneLinks_LEO is a sub-property of IntraPlaneLinks property. It is a datatype property, which has its own datatype, range, and allowed values besides the

domain constraints (shown in Fig. 4). As to object property, it is useful to inverse slots. Inverse slot contain redundant information, but allow acquisition of the information in either direction, enable additional verification, and allow presentation of information in both directions. In Fig. 4, it is shown that IsDeviceof and HasDevice are inverse slots.

*4) Create an instance of a class*

When we create an instance of a class, the class becomes a direct type of the instance. Any superclass of the direct type is a type of the instance. At this stage, we must assign slot values for the instance frame, and slot values should conform to the facet constraints. In fact, this work is always done when we establish the mission view, because only at that time can we figure out the real slot values corresponding to the current mission.

The service view and mission view can be established in the same way. Finally, by using Pellet as reasoning engine, we can detect the violation of property constraints, cycles in the class hierarchy, terms that are used but not defined, interval restrictions that produce empty intervals (min > max), so as to confirm the consistency of the simulation case. In addition, classes with a single subclass, classes or slots with no definitions, slots with no constraints can also spotted. Fixing these problems is helpful to improve the quality of simulation application.

## V. TEST AND RESULTS

The best test is the application for which the ontology was designed. So we set up a specific mission view to describe the simulation analysis of the coverage characteristics in satellite network, and perform consistency verification on the description. After getting the confirmed results, we develop a simulation application to test whether this ontology-based approach is applicable.

Coverage characteristics are very important for satellite network to perform well. It is not enough to simulate the coverage characteristics of a single satellite; coverage characteristics of the whole constellation or network are also needed to form a consistent system. In addition, besides coverage rate, there are many other important criteria including coverage time, coverage continuity, switching times, and coverage degree, which have constraints between each other. As a result, it is suitable to take such a task as an example to test the feasibility of the approach we proposed.

We want to build a simulation application to analyze the coverage characteristics in 4 cases: MEO (Medium Earth Orbit) constellation to ground, MEO constellation to LEO (low Earth Orbit) satellites, GEO (geostationary orbit) constellation to MEO satellites, and GEO constellation to LEO satellites. The main parameters of satellite network to be simulated are shown in Table I. And station node on the ground is selected as A(N: 39.906193，E: 116.388039).

TABLE I. CONFIGURATION OF SATELLITE NETWORK

| **Orbit type** | **GEO** | **MEO** | **LEO** |
|---|---|---|---|
| altitude(km) | 35786 | 11946 | 1000 |
| inclination(°) | 0 | 45 | 99.6 |
| period(s) | 86164.09 | 24685.75 | 6307.12 |
| Number of satellites | 3 | 12 | 4 |
| Number of orbits | 1 | 4 | 4 |
| Phase factor | — | 1 | — |

We developed the simulation in two different ways. The first method is to develop the application all at once according to the requirement description based on HLA in Visual C++ developing environment, which is called Method 1. In this way, we have spent much time on coding and debugging, especially on modifying the improper relationship between federate members. Setting simulation period as 86400s and step as 60s, we can get the simulation results shown in Table II.

TABLE II. SIMULATION RESULTS (METHOD 1)

| **Type** | **Average coverage time** | **Coverage rate (%)** | **Average switching times** |
|---|---|---|---|
| MEO-ground(A) | 8617.3 | 9.97 | 9.537 |
| MEO-LEO1 | 1237.3 | 1.43 | 43.258 |
| GEO-MEO21 | 8766.1 | 10.14 | 11.278 |
| GEO-LEO1 | 2005.0 | 2.32 | 51.944 |

The other method (Method 2) is to apply the ontology-based approach introduced above first, and then develop the application on the foundation of consistent hierarchy confirmed by the approach. Based on the entity view and service views, the mission view is established according to the task analysis. By using Pellet to compute inconsistent concepts in the operational view, we have to spend some time to fix the inconsistent problems discovered by Pellet. During this time, we can form the consistent semantic framework for the current simulation task, which is the preparing work to develop the specific simulation application. Then a corresponding simulation system is developed according to the OWL code generated by Protégé. Under the same simulation condition, the result is shown in Table III.

TABLE III. SIMULATION RESULTS (METHOD 2)

| **Type** | **Average coverage time** | **Coverage rate (%)** | **Average switching times** |
|---|---|---|---|
| MEO-ground(A) | 8178.1 | 9.47 | 10.563 |
| MEO-LEO1 | 1616.7 | 1.87 | 53.442 |
| GEO-MEO21 | 8293.0 | 9.60 | 10.418 |
| GEO-LEO1 | 2102.8 | 2.43 | 41.088 |

The difference between these two methods lies in Table IV, in which is shown the time we spent in developing the simulation application with these methods.

TABLE IV. DEVELOPING TIME (IN DAYS)

| **Method** | **Developing framework** | **Consistency verification** | **Coding** | **Debugging** | **Total time** |
|---|---|---|---|---|---|
| Method 1 | 0 | 0 | 3 | 10 | 13 |
| Method 2 | 1 | 0.5 | 1 | 2 | 4.5 |

## VI. DISCUSSION

From the Table 2 and Table 3, we can conclude that both methods can get the similar simulation results: as to the cases involve LEO satellite, the average coverage time is relatively short, but the switching times is rather high; in contrast, the average switching times in the other 2 cases is rather lower.

Although the exact data produced by the two systems are not just the same, they are both helpful to discover the difference among those four simulation cases and to analyze the reason for the difference.

From the results shown in Table 4, it is clear that Method 2 is superior to the Method 1 in respect of developing efficiency. Although 1.5 days is spent in developing the ontology-based framework and performing the consistency verification in Method 2, we can save 10 days in the period of coding and debugging. As to such a simple simulation task to analyze the coverage characteristics in satellite network, we can save 9.5 days in total, the improvement of which is over 200%. Because of the reusability and interoperability of ontology-based framework, the improvement will be more significant in face of more complex tasks, especially when involving different simulation requirements and services. Most of all, we can confirm the consistency of the simulation system by applying ontology engineering, which is unable to be provided by the pure experiments, This is the key point of this study.

Currently, more and more groups are involved in such an effort to apply ontology engineering in different field, and it shows that communication, interaction, and system building really require common shared ontologies. As to the semantic requirements of satellite network simulation, it is also required to provide consistency verification by applying ontology engineering. The objective of this study is to propose a conceptual model and analytical method to set up a formal description of satellite network, which can be easily used in other simulation tasks to support semantic consistency and to promote the developing of simulation applications. In order to meet the requirements proposed in Part 3, we hope to do more work in the following areas to make the approach more applicable, including:

- Verifying the feasibility and possible advantages of such ontology-based method in more complex simulation scenarios.
- Studying how to organize the relationship among the entity view, the service view and the mission view to set up a more integrated framework.
- Studying how to distinguish the level and granularity of logic description to prevent too much complexity.

## REFERENCES


[1] Lloyd Wood, George Pavlou, and Barry Evans, “Effects on TCP of routing strategies in satellite constellations”, IEEE Communications Magazine, 2001,39(3), pp,172-181.

[2] I.F. Akyildiz, , E. Ekici, and M.D. Bender. MLSR: a novel routing algorithm for multilayered satellite IP networks, IEEE/ACM Transactions on Networking, 2002,10(3).pp, 411-424.

[3] E, Frazzoli, G,B, Palmerini, and F, Graziani, “Debris Cloud Evolution: Mathematical Modeling And Application To Satellite Constellation Design”, Acta Astronautica, 1996,39(6),pp,439—445.

[4] Hoare, C.A.R., “An Axiomatic Basis for Computer Programming”, Communications of the ACM, 1969,12(10), pp, 576-583.

[5] Natalya F. Noy and Deborah L. McGuinness (2001) “Ontology Development 101: A Guide to Creating Your First Ontology”,http://protege.stanford.edu/publications/ontology_development/ontology101.html

[6] Matthew Horridge, Holger Knublauch, etc, A Practical Guide To Building OWL Ontologies Using The Protege, The University Of Manchester, 2004.4.


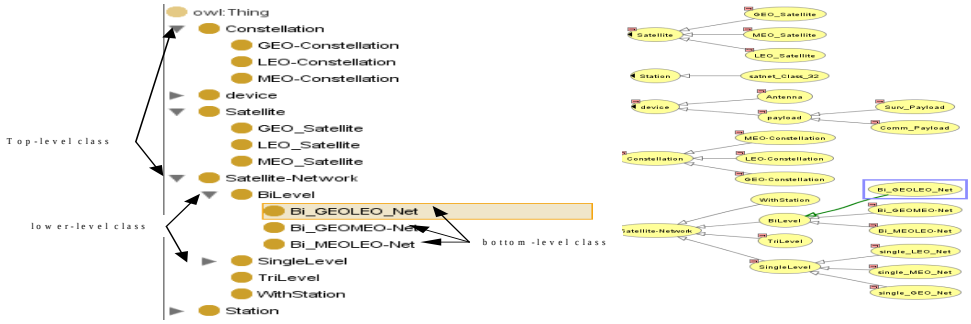


Figure 3. Class hierarchy

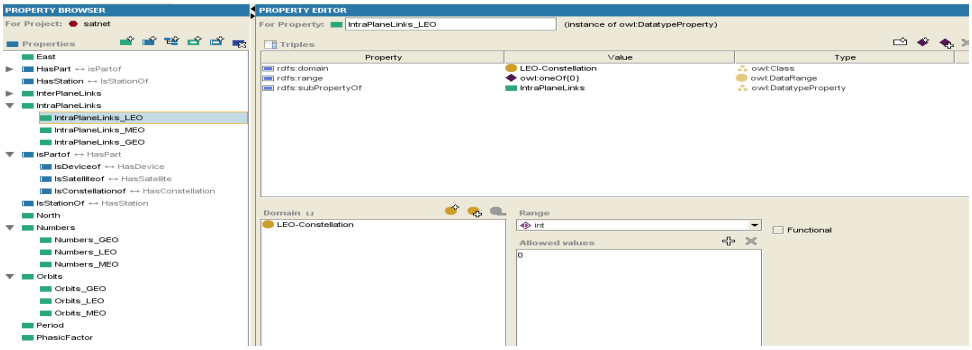


Figure 4. Properties of classes