Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

AN ONTOLOGY FOR TRAJECTORY SIMULATION

Umut Durak
TUBITAK-SAGE
PK.16 Mamak
06261 Ankara, TURKEY

Halit Oguztuzun
Dept. of Computer Engineering
Middle East Technical University
06530 Ankara, TURKEY

S. Kemal Ider
Dept. of Mechanical Engineering
Middle East Technical University
06530 Ankara, TURKEY

ABSTRACT

From the concept exploration for a weapon system to training simulators, from hardware-in-the-loop simulators to mission planning tools, trajectory simulations are used throughout the life cycle of a weapon system. A trajectory simulation can be defined as a computational tool to calculate the flight path and flight parameters of munitions. There is a wide span of trajectory simulations differing widely with respect to their performance and fidelity characteristics, from simple point-mass simulations to six-seven degrees of freedom hardware-in-the-loop missile simulations. From our observations, it is a common practice in the industry that developments of these simulations are carried out as isolated projects although they rely on the same body of knowledge. We envision an ontology that will capture the common knowledge in trajectory simulation domain and make domain knowledge available for reuse. Trajectory Simulation Ontology, dubbed TSONT, is being developed to realize this vision.

1 INTRODUCTION

We present a formal ontology developed for trajectory simulations, called Trajectory Simulation Ontology (TSONT). The aim of this work is the construction of a reuse infrastructure to be used in the development of a variety of trajectory simulations. The ontology is regarded as the domain model component of the reuse infrastructure. It is being developed to be a reusable knowledge library on trajectory simulations.

Trajectory simulation, in the present context, means computing the flight path and other parameters, such as orientation, and angular rates of the munition from the start to the end of its motion (U.S. Department of Defense 1995). Trajectory simulation deals with mathematical models of the behavior of munition and its subsystems during its operation. The equations of motion determine the acceleration, velocity and position of the munition resulting from forces and moments due to gravity, thrust and aerodynamics. The guidance and control models account for subsystems such as the control system.

Trajectory simulations are widely used throughout the product lifecycle of weapon systems. The objective is to improve the understanding of various aspects of the weapon system for a variety of tasks, such as specifying munition performance requirements, designing munitions, optimizing the design parameters, assessing munition performance, teaching users the correct use of weapon system, and fire control.

Early ontology development in technical domains was carried out in mid 90's; consider, for example OLMECO (mechatronic design components) and KACTUS (technical domain ontologies). Ontologies were regarded as reusable knowledge libraries (Benjamin et al. 1996) (Borst and Akkermans 1997) (Borst et al.1995) (Schreiber, Wielinga and Jansweijer 1995). The term is borrowed form philosophy, where it means the systematic exploration of existence. First, Neches defined ontology as the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary. Later, following (Gruber 1993) ontology is defined as formal and explicit specification of a shared conceptualization. See (Corcho 2003) for a survey of ontology definitions as well as methodologies and tools.

For the trajectory simulation ontology, we use OWL (Web Ontology Language) as ontology definition language and Protégé as the Ontology Development Environment.

In the rest of this paper we first discuss the objectives of this study. After discussing the basics of ontologies, we present our Trajectory Simulation Ontology.

2 OBJECTIVES

Each particular trajectory simulation requires a customized approach. The level of sophistication of simulations varies greatly depending on the application. These levels range from unsophisticated two dimensional models to very detailed six-seven degree of freedom models that include hardware-in-the-loop and seeker simulations.

The requirements of a given trajectory simulation are derived from the objectives of the intended user, who might be interested in the analysis, development, procurement and operation of some munition.

We observe, it is common practice to develop trajectory simulations for each and every application again and again. Considering the complexity of the modeled systems and requirements of the simulation application, the risk of failure in such projects is considered to be high. Besides the risk of failure, expenditure of intellectual labor to study similar problems of the same domain is a waste. Another concern is the quality of the products of each development. The verification in trajectory simulation project requires a great deal of effort due to the demand for experts' time and flight data, which are both expensive. Implementing a systematic software reuse will help make best use of past successful efforts.

Research on software reuse suggests that the success of reuse is related to the use of artifacts in the context of a domain, where a domain is defined as the area in which an organization does business (Favaro 1995). Knowledge about the problem domain is often implicit and informal, while reusable information must be represented explicitly and formally. The term reuse infrastructure refers to the information that must be made available to the software developer, together with auxiliary information needed to locate and manipulate it. Developing a reuse infrastructure for a problem domain is the essence of domain engineering. Domain engineering comprises three fundamental processes: domain analysis, infrastructure specification, and infrastructure implementation (Fablo, Guizzardi and Duarte 2002).

Domain analysis is the identification, acquisition and evolution of reusable information on a problem domain to be reused in software specification and construction (Arango 1989). Infrastructure specification is the selection and organization of reusable information in the model to fit the patterns of reuse in the environment of the user. As a result, an architecture for reusable information, such as a library of programs, or a database scheme, is specified. The infrastructure specification, together with the semantics captured by the domain model, is input to the infrastructure implementation step, which produces and tests the specified components.

The purpose of domain analysis is to construct a model of the problem domain. Then the domain model should serve as:

- An authoritative resource of reference when ambiguities arise in the analysis of the problems or later during the implementation of reusable components.
- A repository of shared knowledge for learning and communication.
- A specification of reusable components for the developer.

As the information is gathered during domain analysis, one faces the problem of representing the knowledge for ease of both human understanding and machine readability. The latter requires a formal way to represent the knowledge. The contemporary ontology development languages all have formal semantics. Thus, the approach adopted in this study is the use of ontologies for knowledge representation.

In our view the reusable assets that are produced in each activity of domain engineering are correlated with the abstraction levels of Model Driven Architecture (MDA) of Object Management Group (OMG) (Kleppe, Bast and Warmer 2003). Ontology is regarded as Computation Independent Model (CIM). The abstract software design produced in infrastructure specification is regarded as Platform Independent Model (PIM). The detailed software designs and reusable libraries developed in infrastructure implementation are regarded as Platform Specific Model (PSM) and Code. The platform can be a particular simulation environment, such as MATLAB.

3 ONTOLOGIES AND MORE

An ontology may take variety of forms, but necessarily it will include a vocabulary of terms and some specification of their meaning. This includes definitions and indication of how concepts are inter-related which collectively impose a structure on a domain and constrain possible interpretations of terms (Uschold and King 1995). Knowledge in ontologies is formalized using five kinds of components: concepts, relations, functions, axioms and instances (Gruber 1993).

The basic idea behind developing an ontology as the domain model of the trajectory simulation domain is first to create a common vocabulary that is agreed among people working on trajectory simulations. Explication and systematization are the other goals of this effort. Ontology building can also be viewed as the elucidation of the implicit assumptions that hamper the knowledge reuse. It is also expected to create a backbone for systematization of knowledge on how to build a trajectory simulation. Vocabulary and knowledge systematization has brought us more or less standardized terms/concepts. (Mizoguchi 2001).

In the last decade ontologies have been used for variety of engineering applications (Benjamin et al. 1996) (Borst and Akkermans 1997) (Borst et al.1995) (Schreiber, Wielinga and Jansweijer 1995) (Ciocoiu, Gruninger and Nau 2001) (Durak, Mahmutyazicioglu and Oguztuzun 2005) (Avci, Kayir and Oguztuzun 2005). We aim to use the ontology as a basis for specifying the requirements for trajectory simulation applications. The benefits of this ap-

proach include documentation, maintenance, reliability, knowledge reuse and also interoperability of the developed applications (Falbo, Guizzardi and Duarte 2002).

Among many ontology languages proposed since early attempts of ontology development, OWL-Web Ontology Languge is selected in this study as the working language (Antoniou and Van Harmelen 2004).

One ontology development methodology suggests after defining the scope of the ontology and considering the possible reuse opportunities of other ontologies, one should enumerate basic terms in the domain. Then after defining the classes and class hierarchy, one should define properties of the classes and the facets of these properties. At the last step one can create the instances of the classes (individuals). The process is typically executed in an iterative manner (Noy and McGuinnes 2001). We applied roughly the same sequence. In each iteration we extended the scope of the ontology. Between iterations, prototype trajectory simulations are built depending on the ontology to validate the structure and peer reviews are handled with domain experts before expanding the ontology.

There are mathematical models in the domain that account for some kind of behavior or some law. Capturing these models in a systematic way and representing them as an integrated part of the ontology is an important concern. At this juncture, the DAVE-ML effort of NASA for the benefit of flight modeling and simulation community, has been leveraged (Jackson et al. 2004).

DAVE-ML (Dynamic Aerospace Vehicle Exchange Markup Language) is a proposed standard method for the interchange of aerospace dynamic models. It is aimed to provide a programming language independent representation of aerodynamics, mass/inertia, propulsion and guidance, navigation and control laws of a vehicle. DAVE-ML, which is XML-based, relies on MathML as a means to describe mathematical relations. MathML is an XML-based language for describing mathematics for machine to machine communication. We take advantage of DAVE-ML to incorporate mathematical models into the ontology. (An example is presented in section 4.2.) Simulation code generation from DAVE-ML models is possible.

4 TRAJECTORY SIMULATION ONTOLOGY

4.1 Top Level Entities of TSONT

The top level entities of TSONT are Trajectory Simulation Attribute, Trajectory Simulation Class, Trajectory Simulation Function, Trajectory Simulation Object, Trajectory Simulation Quantity, Trajectory Simulation Record and Trajectory Simulation Sequence, as shown in Figure 1.

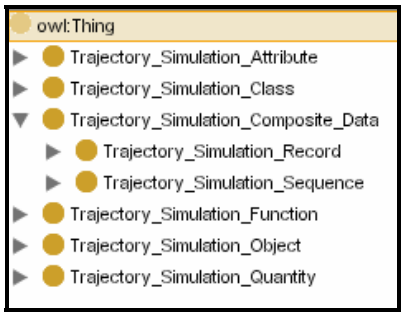

Figure 1: TSONT Top Level Entities

The top level entities of TSONT are matched with those of SUMO (Suggested Upper Merged Ontology). By reusing SUMO, we promote interoperability with other domain ontologies. SUMO is an upper level ontology proposed by the Standard Upper Ontology Working Group, an IEEE-sanctioned working group of collaborators from the fields of engineering, philosophy, and information science. The SUMO provides definitions for general-purpose terms and acts as a foundation for more specific domain ontologies (Niles and Pease 2001).

Trajectory Simulation Attribute can be regarded as the subclass of SUMO Attribute, which is defined as qualities in trajectory simulation domain which we can or prefer not to reify into subclasses of an object. Similarly, Trajectory Simulation Class is regarded as a subclass of SUMO Class and Trajectory Simulation Function as a subclass of SUMO Function. Trajectory Simulation Object, again a subclass of SUMO Object, corresponds roughly to the class of ordinary physical objects in Trajectory Simulation domain. Trajectory Simulation Quantity is defined as any specification of how many or how much of something in Trajectory Simulation domain; it is a subclass of SUMO Quantity.

Trajectory Simulation Record and Trajectory Simulation Sequence are Trajectory Simulation Composite Data types that can be used for developing trajectory simulation codes. Although these data types are well established in programming, we refer to Vienna Development Method Specification Language (VDM-SL), an ISO Standard modeling language, for the sake of definiteness (Fitzgerald and Larsen 1998).

Fig. 2 presents an excerpt from TSONT to show how these top level entities are inherited down to concepts of trajectory simulation domain. Trajectory Simulation can be a Scalar Quantity or a Vectoral Quantity. Acceleration Vector, Angular Acceleration Vector, Angular Velocity Vector, Force Vector, Moment Vector, Orientation Vector, Position Vector and Velocity Vector are all types of Vectoral Quantity. Aerodynamic Force, Gravitational Force and Thrust Force are all derived from the Force Vector.

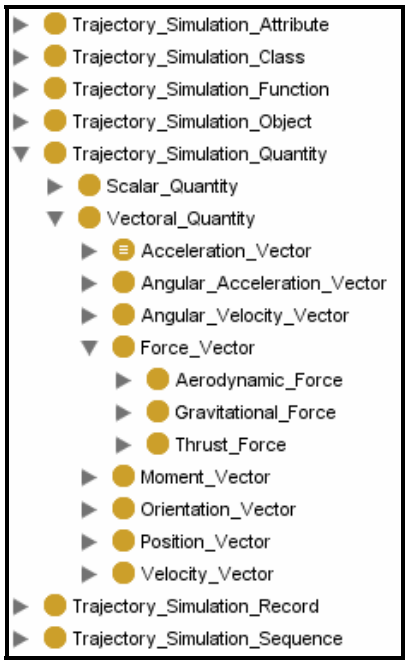

Figure 2: Excerpt from TSONT

4.2 Structure of TSONT

The structure of TSONT is devised to render concept to implementation mapping amenable to reuse by trajectory simulation developers.

Trajectory simulations, which can be composed of multiple phases, are to be executed to calculate the trajectories of munitions launched from a weapon. This fact is reflected in TSONT as depicted in Figure 3. Simulation is modeled as a sub entity of Trajectory Simulation Class. Simulation is defined by hasMunition, hasPhase, hasTrajectory, hasWeapon and servesComputeTrajectory properties. These properties formalize the definition of the trajectory simulation.

Trajectory simulation phases are defined as the segments of a munition flight whose simulation can be performed by using a set of models solved by a numeric solver. For example, computing the trajectory during boost phase and after motor is off, which is called free flight, require a particular sets of models.

Figure 4 lists the models that are used to simulate the flight of a munition as they are represented in TSONT. These models are used to solve different behaviors in different phases of simulation. Trajectory phases are generalized into three classes as shown in Figure 5. The first level Phase is nothing but the phase definition of unguided free flight of a munition. It uses aerodynamics model (to compute aerodynamic forces and in some cases moments), dynamics model (to compute accelerations), earth model, environment model and gravity model. Propelled Phase represents the segment of trajectory that starts with propulsion of a munition from a gun by a charge. So this phase has a propellant model in addition. Guided Phase stands as a class for guided munition trajectory phases. To simulate a guided phase of a munition one will need autopilot model, Canard Actuation System (CAS) model, guidance model and sensor model in addition to first level Phase models. A Thrusted Phase is where the thruster is active, so it adds a thruster model to the set. Some phases of flight might possess hybrid characteristics. In a trajectory phase both guidance and thruster might be active, for instance, in an air-to-air missile simulation. Such a phase is derived both from guided phase and thrusted phase.

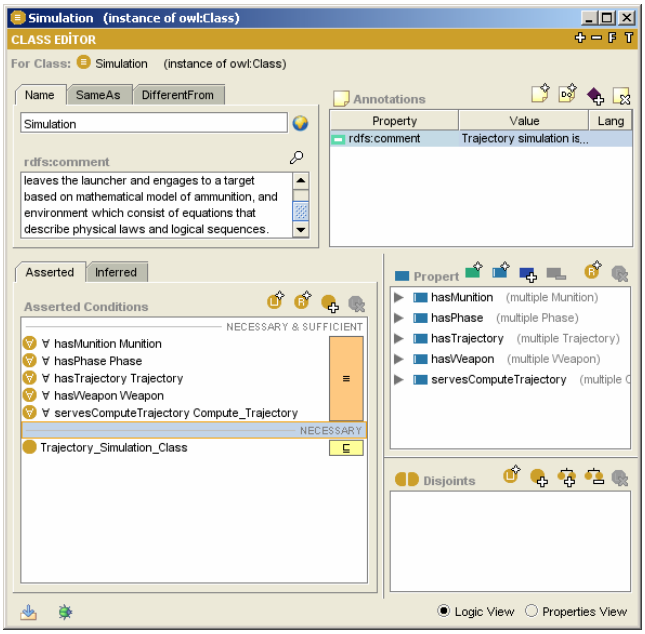

Figure 3: Simulation Definition in TSONT

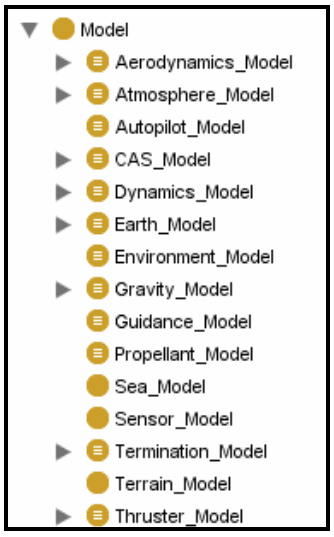

Figure 4: Trajectory Simulation Models in TSONT

Among Trajectory Simulation Models, the Aerodynamics model will be discussed in detail to present our approach to the development of TSONT. Aerodynamics models are the mathematical models we use to compute the aerodynamic force and moments at any time along the trajectory.

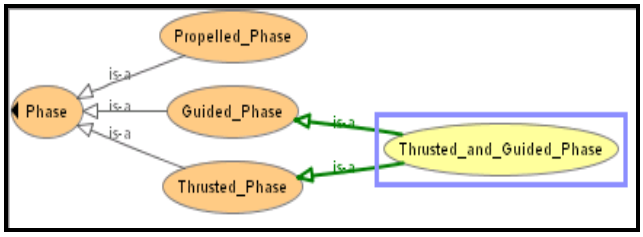

Figure 5: Phase Hierarchy in TSONT

Aerodynamics Model in trajectory simulation can be classified as point mass aerodynamics model or rigid body aerodynamics model. In the hierarchy depicted in Figure 6, body fixed six DOF aerodynamics model is used to compute the aerodynamic forces and moments.

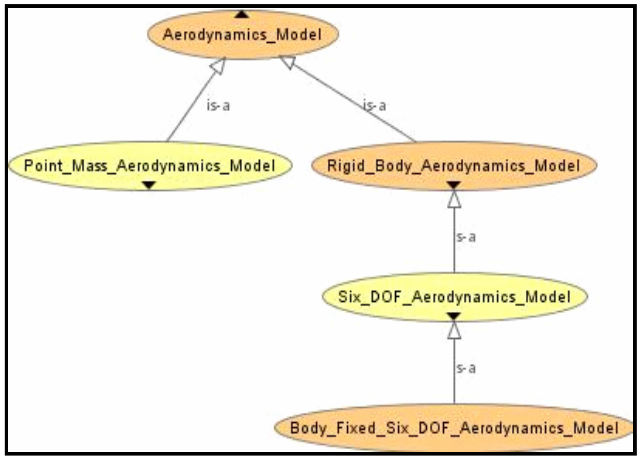

Figure 6: Aerodynamics Model Hierarchy in TSONT

Figure 7 shows the representation of Body Fixed Six DOF Aerodynamics Model in TSONT. This representation formalizes the functions offered by the model, namely, Compute Aerodynamic Forces and Compute Aerodynamic Moments.hy in TSONT.

Functions are regarded as services offered by classes in TSONT. A function either carries out some computation using its inputs and producing outputs, or supplies requested parameters. To illustrate how functions are defined in the ontology, one of the functions of Body Fixed Six DOF Aerodynamics Model, which is Compute Six DOF Aerodynamic Forces with respect to Body Fixed Coordinate System, will be discussed.

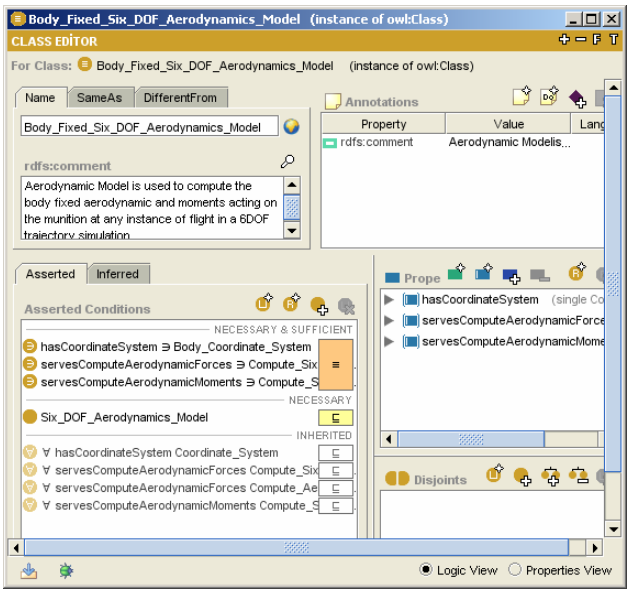

Figure 7: Definition of Body Fixed Six Aerodynamics Model in TSONT

The computation of body fixed aerodynamic forces in a six DOF trajectory simulation can be expressed as in Equation (1).

$$\begin{bmatrix} F_{ax} \\ F_{ay} \\ F_{az} \end{bmatrix} = Q_d \cdot A \cdot \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}, \qquad (1)$$

where F_{ax} , F_{ay} and F_{az} are the aerodynamic forces in body fixed axes, x, y and z respectively. Q_d is the free stream dynamic pressure, A is the reference area of the munition, and C_x, C_y, and C_z are the aerodynamic coefficients in the respective directions (Mahmutyazicioglu 1994). This function is defined in the ontology as presented in Figure 8.

The inAerodynamicsRecord, inAtmosphereRecord, inBallisticRecord and inDynamicModelState properties with outAerodynamicForce property formalizes the input output mapping of function. Implementation property refers to the DAVE-ML file that documents the mathematical mapping of inputs and outputs of the function, given Equation (1), in a human and machine readable form. The Equation (1) is expressed in DAVE-ML as:

```
<variableDef
 name=" Body Fixed Aerodynamic Force"
 varID="FA" units="N" >
   <description>
   Three dimensional body fixed aerodynamic
   force in N. It is a vector (FAX,FAY,FAZ)
   in first, second and third axes respect-
   tively.
  </description>
```

```
    <calculation>
      <math>
        <apply>
          <eq/>
          <list>
            <ci>Fax</ci>
            <ci>Fay</ci>
            <ci>Faz</ci>
          </list>
          ...
        </apply>
      </math>
    </calculation>
    <isOutput/>
  </variableDef>
```

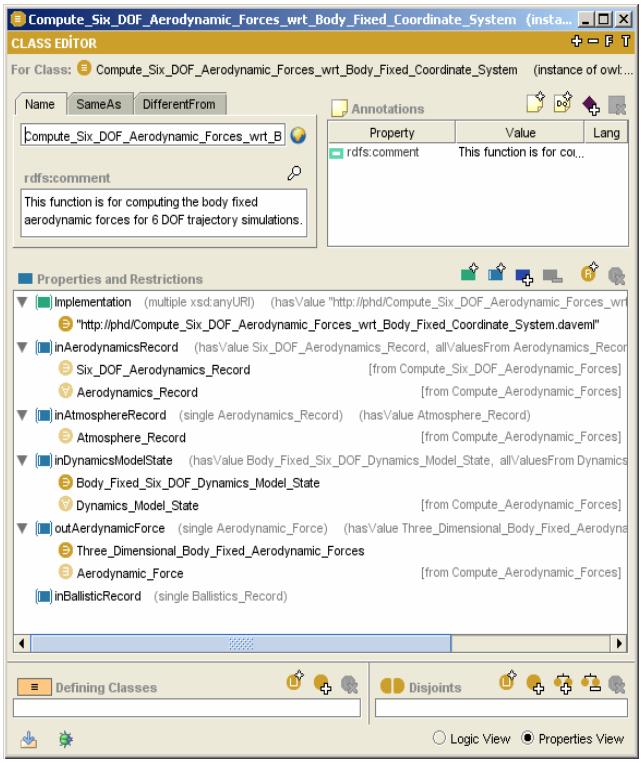

Figure 8: Definition of Compute Six DOF Aerodynamic Forces wrt Body Coordinate System in TSONT

C_x, C_y, and C_z the aerodynamic coefficients of a munition, are parameters of a trajectory simulation. The entities derived from Trajectory Simulation Object are physical objects which are the subjects of the simulation study. The physical objects are represented in the simulation by their parameters. For example, the munition whose trajectory is to be computed is defined in simulations by its Aerodynamics and Ballistics parameters as given in Figure 9.

Aerodynamics is one of the parameters used in trajectory simulation, listed below in Figure 10. Parameter classes serve functions to provide the required data to model classes, which are responsible for the generation of behavior. In this manner, Six DOF Aerodynamics serves Compute Six DOF Aerodynamics function, which is used to retrieve Aerodynamics Record. Aerodynamic Record, in turn, is required to compute the Aerodynamic Forces, as presented in Figure 8.

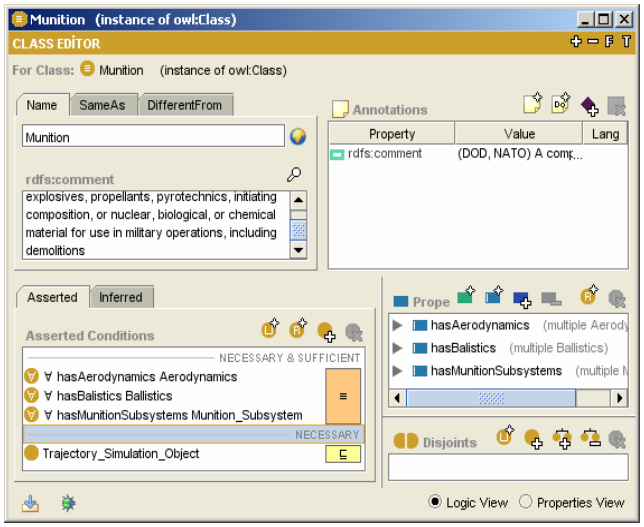

Figure 9: Definition of Munition in TSONT

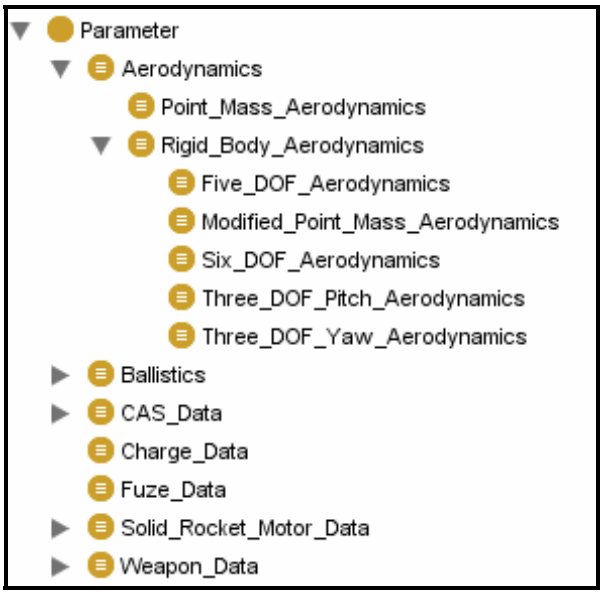

Figure 10: Trajectory Simulation Parameters in TSONT

All the records that are defined in the ontology are derived from Trajectory Simulation Record, presented in the previous section. NACA Six DOF Aerodynamics Record, a convention to represent Six DOF Aerodynamics, is an example of a Trajectory Simulation Record. C_x, C_y, and C_z are the components of NACA Six DOF Aerodynamics Record; refer to Figure 11 for its definition in TSONT.

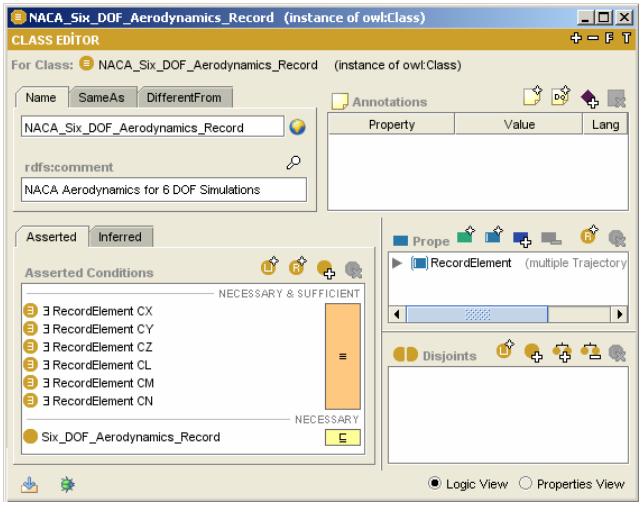

Figure 11: NACA Six DOF Aerodynamics Record in TSONT

4.3 Individuals of TSONT

While the OWL classes of the ontology provide meta-level information, the specific requirements of each particular simulation can be added to the ontology as individuals. The domain structure captured by classes and constraints represented by the conditions of the classes will constrain the relations among these individuals. A collection of individuals is sometimes termed as the knowledge base.

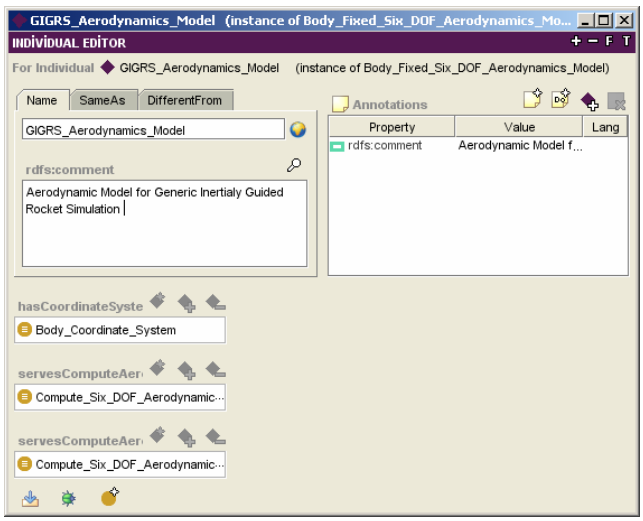

Figure 12: GIGRS Aerodynamic Model Instance in TSONT

Consider, for example, the trajectory simulation for a certain INS guided surface to surface rocket concept, designated GIGRS (Generic Inertial Guided Rocket Simulation). The TSONT user defines an instance of the Body Fixed Six DOF Aerodynamics Model, called GIGRS Aerodynamics Model, as shown in Figure 12. Then, TSONT specifies the coordinate system of the model and the functions that the GIGRS Aerodynamic Model should serve. This demonstrates the use of TSONT as a specification for a trajectory simulation.

5 CONCLUSION AND FUTURE WORK

TSONT construction effort is an attempt to capture the knowledge in the trajectory simulation domain. By matching the top level entities in the ontology with the entities of the widely accepted upper level ontology SUMO, future opportunities to interoperate with other domain ontologies are facilitated. Composite data types that are used in trajectory simulations are grouped depending on a well established software specification language, VDM-SL, in order to make these definitions clear and implementation language and platform independent. The top down structure starting from the basic concepts of trajectory simulation, goes through phases of simulation, the models that determine the behavior throughout these phases, the functions that capture the input-output relations and the DAVE-ML, which is incorporated into ontology to capture the mathematical definitions. DAVE-ML portions of the ontology provide guidance on how to structure a trajectory simulation and how to implement the mathematical relations.

TSONT is being developed based on the experience gained in the past trajectory simulation development projects of Modeling and Simulation Division of TUBITAK-SAGE. TSONT will be used in the upcoming projects as a simulation conceptual model as well as a domain model. Designs of numerous trajectory simulation projects will be based on the structure captured in TSONT. This will validate the capability of the TSONT in knowledge and design reuse.

ACKNOWLEDGMENTS

This work is supported by Defense Industries Research and Development Institute of Scientific and Technological Research Council of Turkey (TUBITAK-SAGE).

REFERENCES

Antoniou, G. and F. Van Harmelen. 2004. Web Ontology Language: OWL. Handbook on Ontologies, International Handbooks on Information Systems, Springer.

Arango, G. 1989. Domain Analysis: From Art to Engineering Discipline. In *Proceedings of 5th International Workshop on Software Specification and Design*, Pittsburgh, PA.

Avci, U., S. Kayir, and H. Oguztuzun. 2005. An OWL ontology for shell trajectories. *Journal of Defense Sciences*, vol. 4, no. 1, 123-140.

Benjamin, J., P. Borst, J. M. Akkermans, and B. J. Wielinga. 1996. Ontology Construction for Technical Domains, In *Proceedings of the 9th European Knowledge Acquisition Workshop on Advances in Knowledge Acquisition*, 98-114.

Borst, P., J. Akkermans, A. Pos, and J. Top. 1995. The PhysSys ontology for physical systems. In *B. Bredeweg, editor, Working Papers of the Ninth International Workshop on Qualitative Reasoning QR'95*, 11-21. University of Amsterdam.

Borst, W. N., J. M. Akkermans. 1997. Engineering Ontologies. *International Journal of Human-Computer Studies*, 46 (2/3):365-406.

Ciocoiu, M., M. Gruninger, and D. S. Nau. 2001. Ontologies for Integrating Engineering Applications. *Journal of Computing and Information Science in Engineering*, (1) 1, 12-22

Corcho,O., M. F. Lopez, and A. G. Perez. 2003. Methodologies, Tools and Languages for Building Ontologies. Where is Their Meeting Point? *Data & Knowledge Engineering,* Vol 46, 41-64.

Durak, U., G. Mahmutyazicioglu, and H. Oguztuzun. 2005. Domain Analysis for Reusable Trajectory Simulation. *Euro SIW'05*, 303-312, Toulouse, France.

Falbo, R.A., G. Guizzardi, and K. C. Duarte. 2002. An Ontological Approach to Domain Engineering. *International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy.

Favaro, J. 1995. Technical Report on Reuse. European Software Institute.

Fitzgerald, J., and P. G. Larsen. 1998. Modelling systems: practical tools and techniques in software development. Cambridge University Press.

Gruber, T.R. 1993. A Translational Approach to Portable Ontology Specifications. *Knowledge Acquisition*, Vol. 5, Number 2.

Jackson, E., B. Hildreth, B. York, and W. Cleveland. 2004. Evaluation of a Candidate Flight Dynamics Model Simulation Standard Exchange Format. *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Providence, Rhode Island.

Kleppe, A., W. Bast, and J. B. Warmer. 2003. *MDA Explained, the Model Driven Architecture: The Model Driven Architecture: Practice and Promise*.2nd Ed. Boston:Addison-Wesley.

Mahmutyazıcıoglu, G. 1994. Dynamics and Control Simulation of an Inertially Guided Missile. Masters Thesis, Department of Mechanical Engineering, Middle East Technical University, Turkey.

Mizoguchi, R. 2001. Ontological Engineering: Foundations of the Next Generation Knowledge Processing. *Web Intelligence 2001*, Maebashi City, Japan.

Niles, I., and A. Pease. 2001. Towards a Standard Upper Ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Chris Welty and Barry Smith, eds, Ogunquit, Maine.

Noy, N.F. and D. L. McGuinnes. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report SMI-2001-0880, School of Medical Informatics, Stanford University, USA.

Schreiber, G., B. Wielinga, and W. Jansweijer. 1995. The KACTUS View on the 'O' Word. In *Proceedings of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*. Montreal, Canada.

U.S. Department of Defense. 1995. Missile Flight Simulation, Part One Surface-to-Air Missiles. MIL-HDBK 1211.

Uschold, M., M. King. 1995. Towards a Methodology for Building Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada.

AUTHOR BIOGRAPHIES

UMUT DURAK is a Senior Researcher in Defense Industries Research and Development Institute of Scientific and Technological Research Council of Turkey (TUBITAK-SAGE). He has been involved in modeling and simulation of weapon systems. He obtained his BS and MS from Mechanical Engineering Department of Middle East Technical University (METU), Ankara, Turkey. He is currently a Ph.D. student in the same department. His e-mail address is <umut.durak@sage.tubitak.gov.tr>.

HALIT OGUZTUZUN is an associate professor in the Department of Computer Engineering at the Middle East Technical University (METU), Ankara, Turkey. He obtained his BS and MS degrees from METU in 1982 and 1984, and PhD from University of Iowa, Iowa City, IA, USA in 1991. His current research interests include distributed simulation and model-driven engineering. His e-mail address is <oguztuzn@ceng.metu.edu.tr>.

S. KEMAL İDER is a professor in the Mechanical Engineering Department at the Middle East Technical University (METU), Ankara, Turkey. He obtained his BS and MS degrees in Mechanical Engineering from METU, both in 1976. He received MS degree in Economics in 1979 and PhD degree in Mechanical Engineering in 1988 from the University of Illinois at Chicago. His research interests include multibody dynamics and control of flexible systems. His e-mail address is <kider@metu.edu.tr>.