Special Issue: Journal of Simulation
Enhancing Simulation Composability and Interoperability using Conceptual/Semantic/Ontological Models
Andreas Tolk and John A. Miller, eds.

ONTOLOGIES AND TOOLS FOR ANALYZING AND COMPOSING SIMULATION CONFEDERATIONS FOR THE TRAINING AND TESTING DOMAINS

Reginald Ford
David Martin
Daniel Elenius

SRI International
333 Ravenswood Ave
Menlo Park, CA 94025, USA

Mark Johnson

SRI International
4111 Broad Street
San Luis Obispo, CA 93401, USA

Abstract

Military training and testing events integrate a diverse set of live and simulated systems, most of which were built independently and weren't specifically designed to work together. Data interoperability and service-oriented architecture (SOA) approaches, while essential, do not provide a complete solution to ensuring that systems will be fully compatible in their interactions. We describe a complementary approach that uses Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) to capture information about the roles and capabilities required to complete a task, and the detailed attributes of candidate resources. Our toolset applies automated reasoning to determine whether each candidate resource has the requisite capabilities and is compatible with other resources. If there are multiple candidates for a role, the reasoner ranks the relative goodness of each with respect to constraints and metrics that are appropriate for the specific task needs of the exercise or deployment. We include examples illustrating the kinds of information we capture about resources and how rules and constraints are applied to provide a nuanced assessment of their compatibility in a specific context.

1 INTRODUCTION

1.1 The Problem

Much hard work and ingenuity has gone into making heterogeneous Modeling and Simulation (M&S) and live instrumentation systems work together in complex live-virtual-constructive (LVC) federations. However, the level of interaction achieved may be relatively superficial (e.g., a common operating picture), and the gains fragile. The points of interaction among systems are myriad and often involve subtle nuances. A successful federation of systems may be forged using one or more of the extant interoperability architectures, but if one changes a little here, a little there, it all comes unglued.

One of the fundamental impediments is the lack of standard, unambiguous, and accessible descriptions of those nuances and the possible consequences of a mismatch. There is a growing recognition that nascent Semantic Web technologies hold considerable promise in overcoming this kind of problem, although practical results to date are relatively sparse. Another impediment is the lack of automated tools that are capable of transcending the naturally limited human ability to process voluminous and intricate information.

This paper[1] describes a conceptual approach for expressing detailed and precise information about systems and their interoperability context, and an automated toolset that applies a reasoning engine to draw conclusions about interaction issues and opportunities. To succeed in practice, the knowledge bases (KBs) must accurately represent both technical and experiential knowledge about a broad range of domains. The paper also discusses our initial work engaging communities of subject matter experts (SMEs) to assist our efforts.

Although the Open Netcentric Interoperability Standards for Training and Testing (ONISTT) approach and the toolset were developed to facilitate planning for improvisational LVC training and testing events, they are very general in design and could be applied to a wide variety of domains.

[1]This paper is an expansion and revision of an earlier paper (Ford et al. 2009) prepared for Winter Simulation Conference 2009 (WSC 2009).

1.2 Related Work

ONISTT builds on an extensive foundation of related work. For more than two decades, activities within the M&S community have pioneered the art of connecting disparate systems in temporary lash-ups to provide a desired set of capabilities that no single system could provide. Although many successful improvisational LVC federations have been built and employed, these successes have typically required considerable event-specific effort. Despite a number of activities to define standard data models, communications mechanisms, and integration processes, routine success has been elusive. Considerable research has been devoted to finding the root causes of this shortcoming.

Dahman (1999) introduced the notion that the success of improvisational LVC confederations[2] requires two distinct kinds of interoperability, termed "technical" and "substantive". Tolk and Muguira (2003), and later Turnitsa (2005), extended Dahman's decomposition to five- and six-level models (respectively) within a hierarchical structure called the Levels of Conceptual Interoperability Model (LCIM). While many interoperability communication architectures have addressed the first two LCIM levels (i.e., technical and syntactic), the ONISTT ontology-based framework also addresses the upper four levels (i.e., semantic, pragmatic, dynamic, and conceptual). See Tolk et al. (2008) for a discussion of the relationship between LCIM and Semantic Web technologies. The NATO Semantic Interoperability Logical Framework is a recent initiative that is investigating methods for applying semantic technologies to foster interoperability at the "upper" levels (Bacchelli et al. 2009).

ONISTT has also been influenced by recent DoD initiatives to develop a framework for building systems that can be networked together to provide improvisational capabilities – that is, capabilities that were not initially defined for the constituent systems at the time of their construction. These initiatives include the Net-Centric Data Strategy (NCDS), Net-Centric Operations and Warfare (NCOW) (DoD CIO 2009a), the DoD Metadata Registry (MDR) (DoD CIO 2009b), and the NATO Net Enabled Capability (NNEC) (Director, IS & NNEC ICT 2009). These initiatives are based largely on the creation of online accessible metadata, and employing the tenets of Service Oriented Architecture (SOA) (OASIS 2006).

Kasputis, Oswalt, McKay, and Barber (2004) discuss the use of semantic descriptors for models and simulations. The thesis is that a well-structured system of semantic descriptors can be used to assess the validity of a simulation federation — and can also define the context under which that validity would hold: "If the same language was used for both simulation requirements and a description of model capability, the groundwork would be in place for development of automated user-defined composable simulations." Vick, Murphy, Cost, and Bethea (2006) introduce the same basic idea. This is essentially the notion upon which the ONISTT project is based, except that ONISTT is more broadly concerned with composable resources of all types (including live and virtual systems in addition to constructive systems).

A related set of problems has been the focus of inquiry in the context of research on Semantic Web services. This field, which aims to enable the automation of the development and use of Web services, takes as its first challenge the enrichment of Web service descriptions. For example, Web Ontology Language for Services (OWL-S) (Martin et al. 2007), the pioneering effort in this field, introduces the expression of preconditions and effects in a Semantic Web-compatible manner, and also relies on the ability to use the Web Ontology Language (OWL) (McGuinness and van Harmelen 2004) to construct class hierarchies of services (OWL and other Semantic Web technologies are also used in ONISTT, as discussed in Section 2.2). Based on such descriptions of services, a variety of approaches have been devised for the selection and composition of services. However, service-based selection and composition rely on descriptions of interfaces and behavior, whereas ONISTT is principally concerned with descriptions of capabilities, which we view as a higher level of abstraction in portraying systems.

Several papers describe tool suites that apply OWL's combination of machine-understandability and rich semantic expressiveness in ways that resemble aspects of the ONISTT framework described herein. Preece et al. (2007) apply ontologies and prototype software to determine which Intelligence, Surveillance, and Reconnaissance (ISR) assets (e.g., type of UAV and sensors) are best suited to a particular mission, which may include constraints such as current weather conditions. Alternative solutions are ranked. Silver et al. (2007) describe an Ontology Driven Simulation (ODS) design tool that maps concepts from domain ontologies to the Discrete Event Modeling Ontology (DeMO), then translates the resulting ontology individuals to an intermediate XML representation, and finally generates an executable simulation model (see also Miller and Baramidze 2005). The tool suites described in Preece et al. (2007) and Silver et al. (2007) share with ONISTT the ability to draw on heterogeneous domain ontologies that may have significant differences in level of abstraction, inconsistent taxonomic classification, and the like.

[2]We intentionally use the term *confederation* here to emphasize the heterogeneous architectures of the large LVC simulations that our work has focused on; *confederation* is intended to convey the absence of a central governing body, as is typically found in a *federation*.

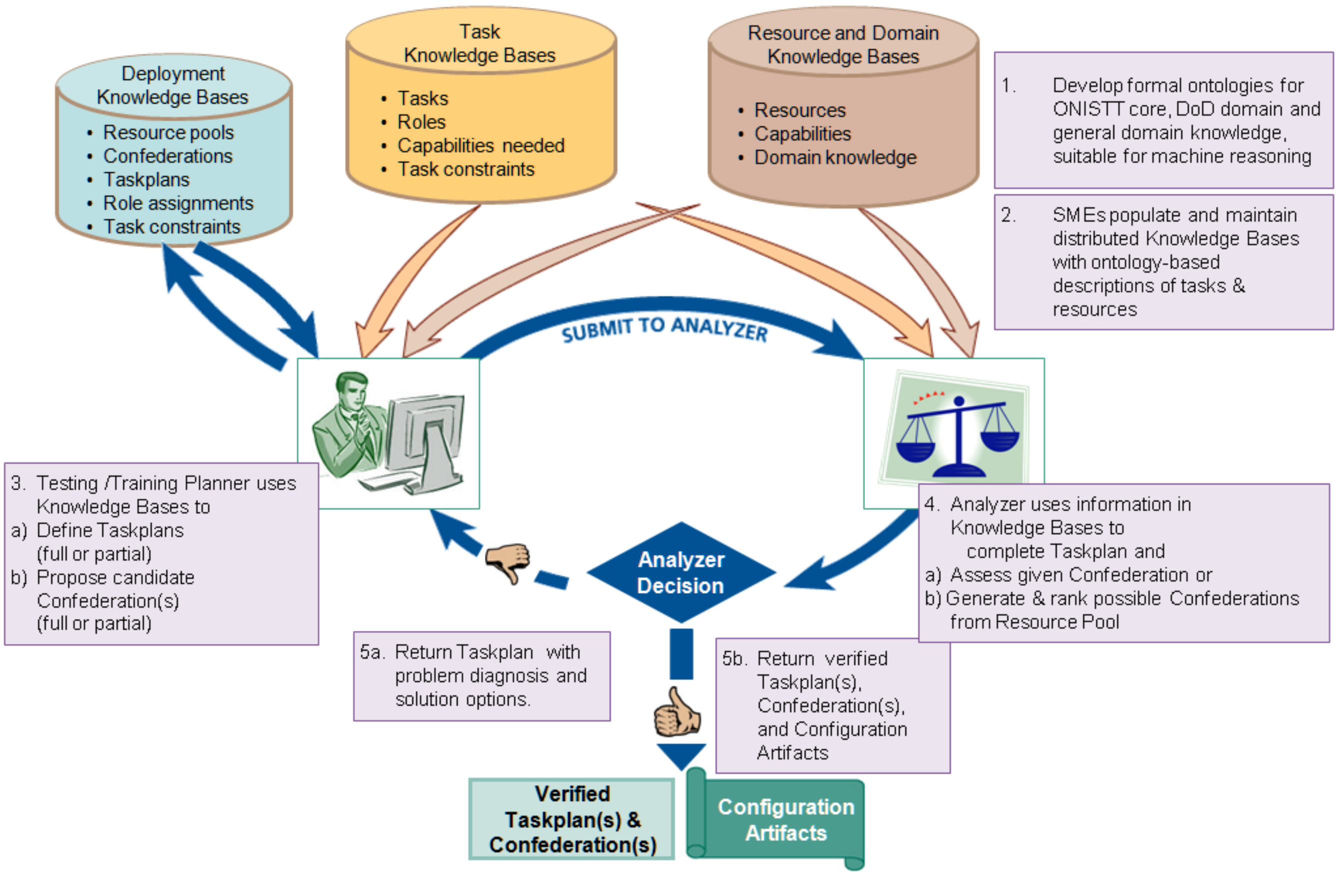

Deployment Knowledge Bases
• Resource pools
• Confederations
• Taskplans
• Role assignments
• Task constraints
Task Knowledge Bases
• Tasks
• Roles
• Capabilities needed
• Task constraints
Resource and Domain Knowledge Bases
• Resources
• Capabilities
• Domain knowledge
1. Develop formal ontologies for ONISTT core, DoD domain and general domain knowledge, suitable for machine reasoning
2. SMEs populate and maintain distributed Knowledge Bases with ontology-based descriptions of tasks & resources
SUBMIT TO ANALYZER
3. Testing /Training Planner uses Knowledge Bases to
a) Define Taskplans (full or partial)
b) Propose candidate Confederation(s) (full or partial)
4. Analyzer uses information in Knowledge Bases to complete Taskplan and
a) Assess given Confederation or
b) Generate & rank possible Confederations from Resource Pool
Analyzer Decision
5a. Return Taskplan with problem diagnosis and solution options.
5b. Return verified Taskplan(s), Confederation(s), and Configuration Artifacts
Verified Taskplan(s) & Confederation(s)
Configuration Artifacts

Figure 1: ONISTT Knowledge capture and Analyzer employment phases.

2 ONISTT OVERVIEW

2.1 The ONISTT Methodology

At the heart of the ONISTT approach is a collection of descriptions of the capabilities of *resources*, which can include all kinds of LVC systems, and the activities, or *tasks*, in which the resources will be used in test and training events. These descriptions are based on ontologies and are stored in knowledge bases (KBs). Given these descriptions, the Analyzer can evaluate the suitability of the resources for those tasks, analyze the ability of the resources to interoperate successfully to carry out those tasks, and perform automated planning of some aspects of test and training events.

The ONISTT methodology is shown in Figure 1. The essential products of the knowledge capture phase are the resource and task KBs shown at the top of the figure. First, formal ontologies are developed to express ONISTT core, DoD domain, and general domain concepts in a way that is suitable for machine reasoning (1). The KBs express information about real-world environments, tasks, infrastructures, and systems in terms of the ontology concepts (2).

The rest of the figure shows the ONISTT employment phase. The planner defines the objectives and constraints of an exercise and proposes a partial or full confederation of participants, systems, and infrastructure (3). To determine whether a proposed confederation satisfies the interoperability needs of the specified event, the Analyzer applies domain-specific interoperability rules, general reasoning technology, and facts captured in the KBs (4a). If the planner leaves some of the confederation assignments blank, the Analyzer selects and ranks candidate systems (4b).

The Analyzer either warns the planner about potential interoperability problems (5a), or returns a verified confederation and configuration artifacts (5b). The Analyzer assigns a severity level to warnings. At this point the planner can submit a modified proposal, or decide that the level of interoperability is good enough for the purposes of the exercise. Since designing and verifying a complex confederation is normally an iterative process, the Analyzer allows the planner to focus on particular parts of the exercise and save results for use in building up the larger confederation. Analysis results can also include artifacts, such as specification of settings for configurable resources, that are appropriate for the proposed deployment context. The long-term goal for ONISTT envisions hosting the KBs on the DoD MDR and implementing the analysis/synthesis function as a Net-centric Enterprise Service.

2.2 Languages and Tools

To perform these kinds of analyses and syntheses, a firm foundation is needed for representing and reasoning about knowledge. In particular, one needs a representation language with support for defining concepts in ontologies, and with a well-defined semantics for reasoning about descriptions based on those ontologies. The semantics of such a language provides a mathematically precise foundation in which the conclusions that are reached are guaranteed to be valid (known as *soundness* of reasoning), and all valid conclusions are guaranteed to be reached (known as *completeness* of reasoning).

To meet these requirements, ONISTT employs OWL, which in turn is layered atop the Resource Description Framework (RDF) (Klyne and Carroll 2004). Both of these knowledge representation technologies have been standardized at the World Wide Web Consortium (W3C) in recent years, as part of the Semantic Web initiative. The Semantic Web initiative arose in 2000 as the result of a technology development partnership between DARPA and the W3C. OWL, a description logic language, is well suited to ONISTT's objectives of describing, classifying, and performing comparisons among categories of resources, capabilities, and tasks, based on their properties.

In addition to the functionality provided by OWL, the ability to express *constraints* and *rules* is needed. For this purpose, we make use of the Semantic Web Rule Language (SWRL). SWRL was designed to be used with OWL and allows for the expression of if–then rules similar in character to those used in the Prolog programming language. Although SWRL has not been standardized, it has a thorough, stable specification that has been submitted to the W3C (Horrocks et al. 2004).

A number of commercial and open-source tools and software libraries are available for use in working with OWL and SWRL. For editing and maintaining ontologies and knowledge bases expressed in these languages, we have primarily relied on the open source tool Protégé (Knublauch et al. 2004). Our use of Protégé, and our Prolog-based reasoning engine, are discussed further in Section 4.

In terms of Figure 1, Protégé is used in developing and maintaining ontologies (2) and knowledge bases (3), both of which are represented using OWL and SWRL. Resource and event referents (1) are expressed in a less formal manner, typically involving text documents and/or design documents expressed in the Unified Modeling Language (UML) (Booch 1993), or UML-based frameworks such as the DoD Architecture Framework (DoDAF) (DoD CIO 2007). The most relevant UML diagram types include class and activity diagrams. In developing ontologies based on these referents, ONISTT leverages the Ontology Definition Metamodel (ODM) (Object Management Group 2009). ODM, recently standardized by the Object Management Group (OMG), supports ontology development and conceptual modeling in several standard representation languages, including OWL, and provides a coherent framework for visual ontology creation based on OMG's Meta Object Facility (MOF) and UML.

3 ONTOLOGIES

The ability of entities, systems, and infrastructure resources to interoperate often rests in very subtle interactions among their individual capabilities and the operational context. ONISTT interoperability analysis must be grounded in detailed, authoritative, and up-to-date KBs about simulators, training instrumentation, vehicles, communication architectures, terrain, training and testing events, and so forth. It is neither feasible nor desirable for most of these to be designed by or under the control of ONISTT developers. Section 3.1 describes some domain ontologies that are used by ONISTT but may also be employed for other purposes. Section 3.2 describes the very small set of core ontologies that make it possible for the domain ontologies to be marshalled for use in ONISTT interoperability analysis.

3.1 Domain Ontologies

A *domain ontology* formally defines the essential concepts, relationships, and terminology that belong to a subject area. Domain ontologies range from fundamental scientific or engineering concepts to specific types of systems or resources. Ideally, domain ontologies will be developed and maintained by a recognized standards body or an organization that is responsible for a resource or is authoritative about a subject area. However, in most cases the ontologies needed to meet ONISTT project objectives do not yet exist. Consequently, the ONISTT team has found it necessary to create our own implementations; we will transition to more official ontologies as they become available. Whenever possible, our domain ontologies are based on the content of available standards documents.

Our engineering_value.owl for physical quantities is an example of a fundamental domain ontology. EngineeringValue subclasses are quantity types like Mass or Length. EngineeringValue individuals contain a magnitude and a mandatory unit. Among the objectives of the ONISTT project is to reduce ambiguity to the greatest practical extent; most readers have probably encountered problems with unstated units of measure, or cases where units were not collocated with the relevant data. Every interesting simulation problem requires the use of quantities.

Also encoded in the engineering value ontology are conversion factors to the unit of measure in the International System (SI) standard for each particular quantity type. (For example, meter is the SI standard unit of length; the conversion factor from foot to meter is 0.3048.) One or at most two applications of these conversion factors allow conversion between any supported pair of units. This practice is consistent with the approach taken by the National Institute of Standards and Technology (NIST) in Thompson and Taylor (2008). This ontology has been enhanced with SWRL rules that support the definition and usage of quantity *intervals* (e.g., “0 to 100 meters”) and the comparison of quantities and quantity intervals.

The notion of a quantity (the pairing of a magnitude and a unit of measure) is so general that Quantity is a top-level abstraction in our ontology. Quantities are certainly not limited to SI concepts; and we have also used this concept to develop a domain ontology for binary data concepts addressd in IEC 80000-13 (ISO 2008). Do *you* know the difference between a mebibyte and a megabyte, or why your 700 MB file (actually MiB) won’t fit on a 700 MB CD?

Related to the engineering value ontology is our engineering_measurement.owl. This ontology is based on the practices documented in ISO (1995). EngineeringMeasurement individuals are built from an EngineeringValue representing the measurand and another EngineeringValue of the same quantity type representing the uncertainty in the measurement (typically one standard deviation). Acknowledging, documenting, and sharing measurement uncertainties can be critical in LVC confederations where live simulations (such as instrumented players and platforms) are included.

It should be noted that these ontologies are used to document, analyze, and synthesize simulation systems and systems-of-systems and are not transmitted as part of the information that is exchanged repeatedly. Contents of simulation messages *on the wire* do not need to be as complete or as verbose. A simplified example is the case where OWL-based, machine-processable metadata declares that all Lengths will be exchanged as meters; the unit indicator can then be omitted from the wire message to reduce bandwidth consumption. This approach can also be used to support heterogeneous exchange, for example, one system publishes metadata saying lengths of one class of data it is sending are to be interpreted as meters and lengths of another type as centimeters, while another application declares centimeters and millimeters, respectively.

The domain ontologies and KBs developed by the ONISTT team are combined to provide standard, unambiguous, and ontologically “deep” descriptions of their concepts. For example, the quantitative concepts described in the engineering value and engineering measurement ontologies are referenced by a group of related ontologies based on the ISO 18026 Spatial Reference Model (SRM) standard (ISO 2006), including spatial reference frame (SRF), abstract coordinate system, object reference model, and reference datum. The SRM ontologies are used, in turn, by domain KBs describing various LVC interoperability communication architectures to identify the spatial reference frames that are associated with their time-space-position information (TSPI) object models or messages. Domain KBs descriptive of specific resource types (e.g., an EA-6B simulator) use the SRM ontologies to describe their native spatial information representation, and also use communication architecture ontologies to describe how this information is published.

Figure 2 shows the main concepts and a few details of the SRF ontology, which is a direct encoding of the SEDRIS SRM ISO/IEC 18026 standard (available online at `http://standards.sedris.org/`). SRFs describe how the terrain interprets coordinates. Some types of SRF are ThreeDSpatialRefFrame, and some of these are Celestiodetic. Two instances of Celestiodetic have the name GeodeticWGS1984. But they come from different ontologies and represent slightly different ways of specifying height above the surface of the earth. Celestiodetic SRFs are also classified as types that need a VerticalComponentIdentification (VCI). A VCI indicates the surface from which the vertical component coordinates are measured. A LittleH VCI means that the vertical component is measured from the ellipsoid, a mathematical idealization of the earth’s shape. A BigH VCI means that the vertical component is measured from the geoid, which describes the (uneven) gravitationally equipotential surface of the earth. There are some places on the earth where the two measurements are the same, but in other places they could differ by as much as a couple of hundred feet. An example of how the ONISTT Analyzer applies this information is given in Section 5.2.

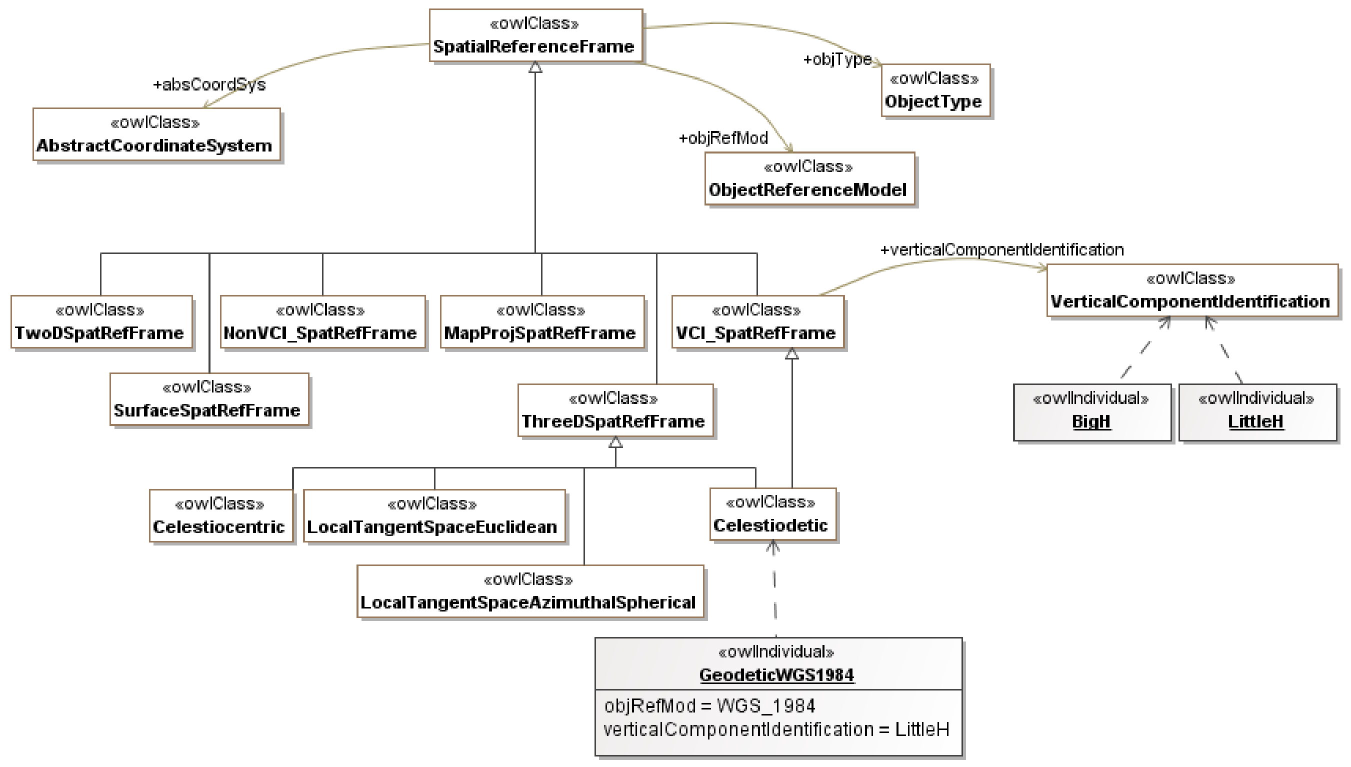

«owlClass» SpatialReferenceFrame
+absCoordSys
«owlClass» AbstractCoordinateSystem
+objType
«owlClass» ObjectType
+objRefMod
«owlClass» ObjectReferenceModel
+verticalComponentIdentification
«owlClass» VerticalComponentIdentification
«owlClass» TwoDSpatRefFrame
«owlClass» NonVCI_SpatRefFrame
«owlClass» MapProjSpatRefFrame
«owlClass» VCI_SpatRefFrame
«owlClass» SurfaceSpatRefFrame
«owlClass» ThreeDSpatRefFrame
«owlIndividual» BigH
«owlIndividual» LittleH
«owlClass» Celestiocentric
«owlClass» LocalTangentSpaceEuclidean
«owlClass» Celestiodetic
«owlClass» LocalTangentSpaceAzimuthalSpherical
«owlIndividual» GeodeticWGS1984
objRefMod = WGS_1984
verticalComponentIdentification = LittleH

Figure 2: Spatial Reference Frame ontology

3.2 Core Ontologies

The ONISTT core ontologies (see Figure 3) are designed to support automated reasoning about the ability of a collection of resources to act in concert for a specific purpose, that is, to perform a specific set of tasks in a specific context. The root concept for that context is Deployment.

The key concepts describing purpose are on the left side of Figure 3. A Task is an intended action that requires some capabilitiesNeeded, and typically has some additional constraints associated with it such as performance and compatibility rules. Tasks can be particularized by arbitrary TaskParameters. A Role parameter is a "slot" of a task that needs to be filled with some resource. A TaskPlan is a plan for how to perform a task, including assignment of resources to roles and values for other parameters. The task and task plan ontologies and the use of SWRL to specify rules and constraints are described in more detail in Elenius, Martin, Ford, and Denker (2009).

The key top-level concepts for capturing precise details about resources are located on the right side of the figure. A Resource is a thing that can *do* something, through the use of capabilities. Examples of resources are an aircraft, a human, a simulator, or training instrumentation. We also allow for intangible resources such as software systems. A resource can have subresources. A Capability is a discrete piece of functionality that belongs to a resource. Examples include the capability to send a "location" message using some message format, the capability to physically move (at some maximum speed), or the capability to detect things in the infrared spectrum. Capability subtypes have properties that specify the relevant functional and performance characteristics, both for resource description and for characterization of the capabilities needed by tasks. A Confederation is a set of candidate resources. The Analyzer looks at the capabilities of candidate resources to determine if they are the right type and if their properties have values appropriate for roles to which they have been assigned, or for any unassigned roles.

Examples of extending these core ontologies are the ONISTT ontologies of capability types for military training and testing resources containing capability subclasses, such as MovementCapability, DetectabilityCapability, and DirectFireCapability. It is important to have a common capability ontology, because both resource and task descriptions refer to these capability types. Task ontologies include variants of military Task types such as TST (time-sensitive targeting) and JCAS (joint close air support).

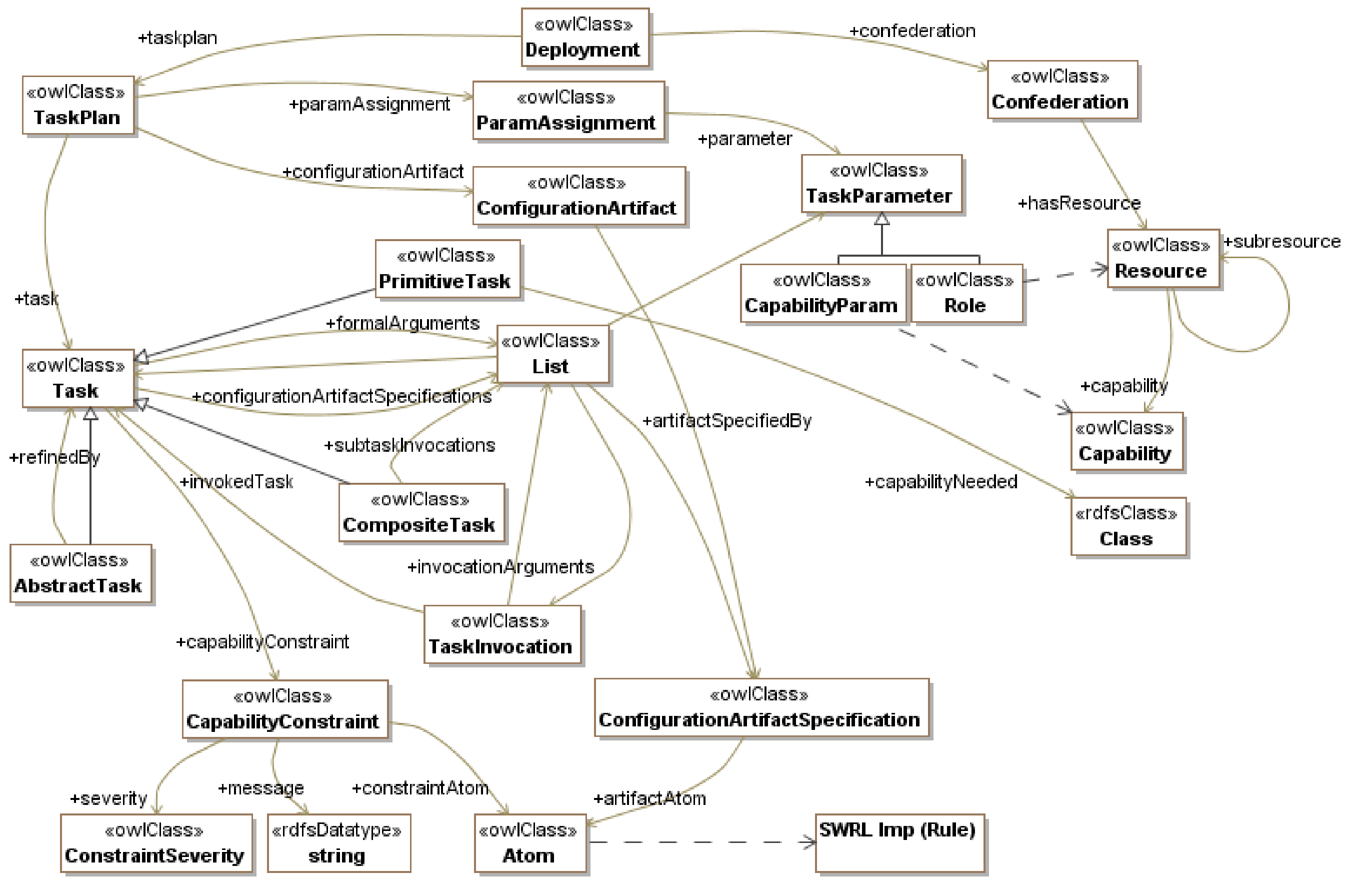


Figure 3: ONISTT core ontology.

4 ANALYZER

This section provides a brief overview of our Analyzer tool. Although a synthesis capability has been added to the tool since its original naming, we still refer to it as the “Analyzer”. For additional technical details, see Elenius et al. (2009). For application examples, see Section 5.

While the ontologies provide the declarative knowledge of the problem domain, the Analyzer provides the computational side of the automation. The job of the Analyzer is to look at the information provided and draw conclusions according to a set of rules. More specifically, the Analyzer takes a (usually partial) task plan and returns a set of completed task plans that are compatible with the input task plan. In doing so, the Analyzer performs several functions:

- Analysis or synthesis of assignments of resources to roles in the task plan. If several resources fulfill the requirements of a role, different solutions are returned.
- Evaluation of constraints. Success or failure of constraints, along with the severity value associated with each failed constraint, determines the relative “goodness” score of a solution.[3]
- Evaluation of configuration artifacts. These are used to return additional information to the task planner, regarding the configuration of the assigned resources, or explanations of why certain constraints failed, as applicable.
- Selection of abstract task refinement. Our task ontology includes a notion of abstract tasks, which can be performed in several different ways. (For example, a task requiring a *tank* resource might have refinements suitable for a live tank, a virtual tank, or a constructive tank). The Analyzer assesses each option, and can identify multiple solutions, if several task refinements are usable given the available resources. The user can limit the solution to one or a few alternatives by constraining the task plan.

The components of the Analyzer tool are shown in Figure 4. The ontologies are stored in OWL files and loaded into the ontology development framework, Protégé. A plug-in to Protégé connects the ontological knowledge with our reasoning tools. The plug-in contains a GUI that helps users create and navigate tasks and task plans, invoke the task engine (described below), and navigate and utilize results from the task engine.

The OWL KB is translated to a native Prolog representation. The translation uses the well-known correspondence of a large subset of OWL, called DLP (Description Logic Programs) (Grosof et al. 2003), to Horn clauses (the translation is described in detail in our previous work Elenius et al. 2007). This means that not all of OWL's semantics are covered (i.e., the query answering is not complete), but in practice we have not found this to be a limitation for the ontologies with which we work, since we do not tend to rely on the more complex OWL axioms and the inferences that they would enable. The Interprolog library connects the Java and Prolog environments.

The Task Engine is the heart of the reasoning process. It is a custom Prolog program that analyzes and synthesizes task plans, evaluates constraints, and produces configuration artifacts. The engine is implemented in XSB Prolog `<xsb.sourceforge.net>`. Prolog was a natural choice because it provides built-in backtracking, which we use to generate all solutions during task plan synthesis.

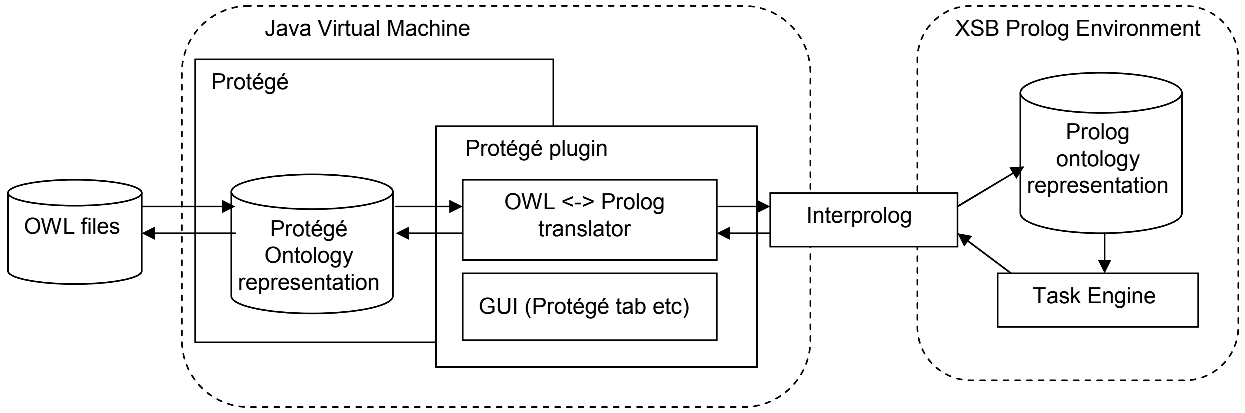


Figure 4: Implementation architecture. The new components are the Protégé plug-in and task engine. As shown by arrows, results from the task engine can be translated all the way back to OWL files.

Figure 9 shows one of several views of the results of running task synthesis for a given deployment. Results can be fairly complex. Our plug-in helps the user explore the result in terms of which resources were used, which warnings were generated, and the structure of the generated task plan. For the expert user, we provide a means to trace the execution of the SWRL rules to understand why a particular conclusion was reached, or in some cases why no solution was returned. In the future, we plan to provide trace capabilities that do not require SWRL expertise to understand.

While our main focus is on military training and testing, it should be noted that the Analyzer tool is fully general. No domain assumptions are built into the tool — all knowledge is provided through ontologies. The tool can thus be used for widely differing problems, as long as they can be expressed in terms of assignment of resources to roles and generation of task plans according to our task and task plan ontologies. One non-training-related example that we have encoded to demonstrate the flexibility and generality of the Analyzer is blood transfusion. In this example, the rules of blood type compatibility are applied to a pool of candidate donors and a recipient to identify and rank matches.

5 EXAMPLES

This section illustrates our use of Semantic Web technology in the simulation domain by describing two self-contained examples in some detail. These examples were part of our 2009 demo for the ANSC project. The demo KBs captured information about actual LVC systems, test infrastructure, and test sites. In this article, we substitute generic resources and hypothetical performance characteristics.

[3]Currently, our approach to scoring is a simple "penalty points" system: for each failed constraint, penalty points are assigned, and lower scores are better solutions; solutions with a score of zero indicate that no constraints were violated.

In a real event, these examples would be only smaller parts in a larger context. Part of the power of Semantic Web technology is the ability to easily gather together results based on widely disparate parts of the combined knowledge base.

5.1 JSTARS detects ground targets

In our first example, we want to simulate a JSTARS aircraft which is trying to detect a simulated T-80 tank. A CSIM-1 constructive system, located at Site-4, is simulating the tank. For the JSTARS, we have two candidate virtual systems (Virtual JSTARS). One is located at Site-1, and one at Site-2 (see Figure 5). We want to find out which one is a better choice.

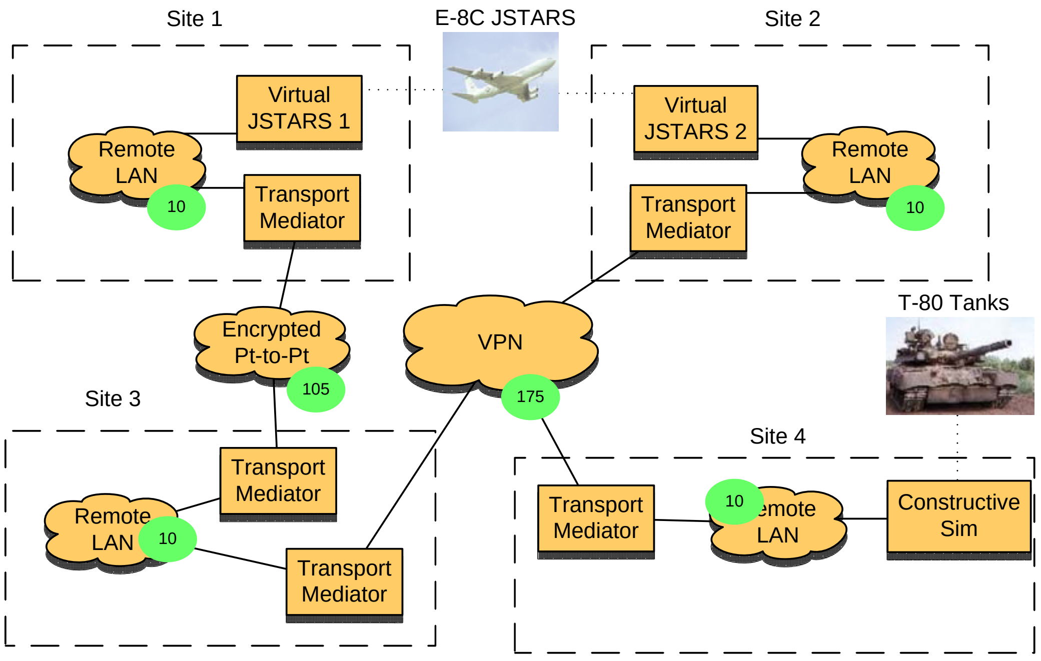
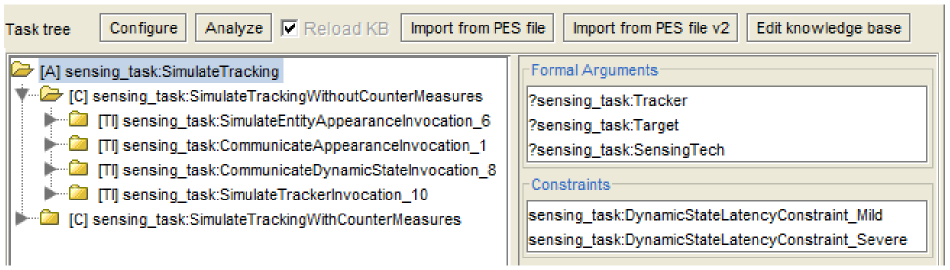


Figure 5: Participants and network topology for JSTARS example. Latency for each transport resource shown.



[A] sensing_task:SimulateTracking
[C] sensing_task:SimulateTrackingWithoutCounterMeasures
[TI] sensing_task:SimulateEntityAppearanceInvocation_6
[TI] sensing_task:CommunicateAppearanceInvocation_1
[TI] sensing_task:CommunicateDynamicStateInvocation_8
[TI] sensing_task:SimulateTrackerInvocation_10
[C] sensing_task:SimulateTrackingWithCounterMeasures

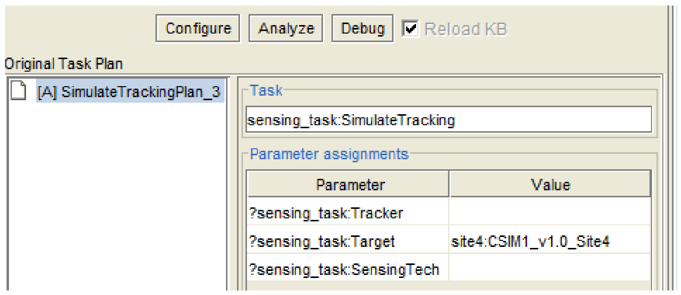
Formal Arguments
?sensing_task:Tracker
?sensing_task:Target
?sensing_task:SensingTech

Constraints
sensing_task:DynamicStateLatencyConstraint_Mild
sensing_task:DynamicStateLatencyConstraint_Severe

Figure 6: SimulateTracking task.

First, we note that the task we want to perform is SimulateTracking, from the sensing_task.owl ontology. This is an abstract task, with two refining tasks: SimulateTrackingWithCountermeasures and SimulateTrackingWithoutCountermeasures. We're interested in the latter case, i.e. with no sensing countermeasures. This in turn is a composite task, with four sub-tasks: SimulateEntityAppearance, Communication (twice), and SimulateTracker (see Figure 6). The communication sub-task is invoked once to communicate the appearance, and once to communicate the dynamic state of the entity.

The composite task also has two constraints on the latency of the dynamic state communication. The first constraint has “mild” severity, and is triggered if the latency is over 100ms. The second constraint has “severe” severity, and is triggered if the latency is over 300ms.



[A] SimulateTrackingPlan_3

Task: sensing_task:SimulateTracking

Parameter assignments

Parameter	Value
?sensing_task:Tracker	
?sensing_task:Target	site4:CSIM1_v1.0_Site4
?sensing_task:SensingTech	

Figure 7: A task plan for the SimulateTracking task.

Next, we set up a task plan for the SimulateTrackingWithoutCountermeasures task. The task has three formal arguments: Tracker, Target, and SensingTech (see Figure 7). We assign the OWL individual CSIM-1_v1.0_site4 to the Target argument. Note that we have different individuals for different installations of the CSIM-1 system. This allows us to describe the network topology, i.e., which networks this particular CSIM-1 installation is connected to. We do not assign a resource to the Tracker role, because we do not yet know which resource we want (we want the Virtual JSTARS installation at either Site-1 or Site-2).

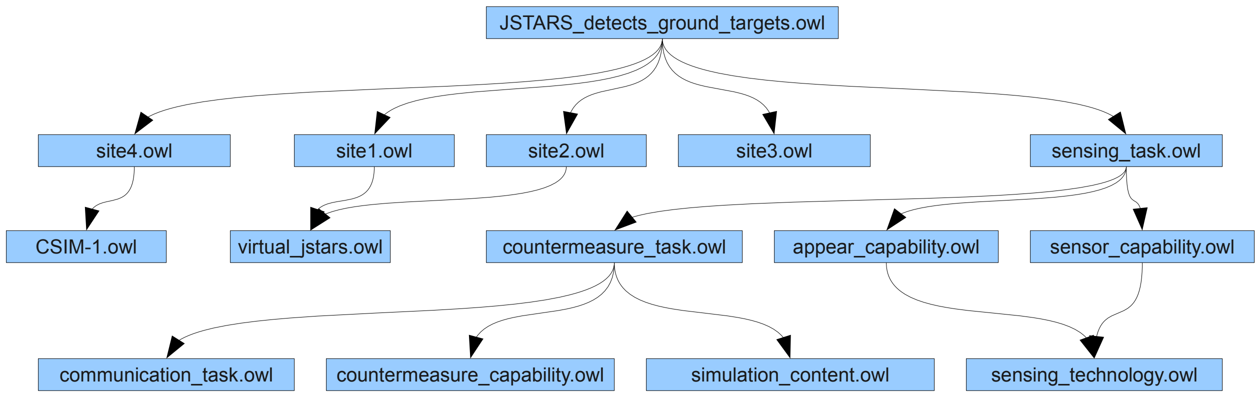
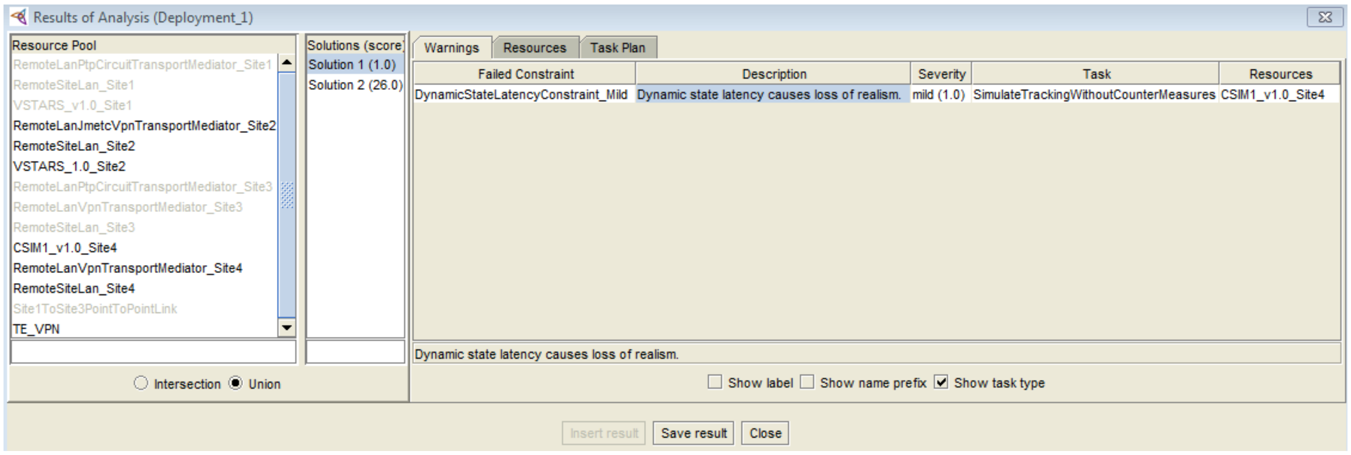


Figure 8: Partial import tree for the JSTARS example, showing task plan, task, capability, resource, and domain ontologies.

The next step is to create a confederation (resource pool) containing all the candidate resources. The confederation includes all the simulation and network resources shown in Figure 5. Note that the resource individuals come from site-specific KBs, for example, site-4.owl and site-1.owl. These, in turn, import generic ontologies of the *types* of systems instantiated there, for example, CSIM_1.owl and virtual_jstars.owl. Figure 8 shows part of the import tree of ontologies used in this example. Note that all of these ontologies except the top-level task plan ontology are reusable in other contexts, because they describe reusable tasks, resources, domain concepts, etc.

The last step before we run the Analyzer is to create a deployment, to bundle up the confederation and the task plan that we created. The result of running the Analyzer is shown in Figure 9. We get two solutions: one with Site-1 Virtual JSTARS as the tracker, and one with Site-2 Virtual JSTARS. The solutions have different scores. The solutions dialog allows us to discover that the Site-1 Virtual JSTARS has the worse (higher) score because it produced the severe dynamic latency warning, whereas the Site-2 solution produced only the mild warning. These results were based on the following considerations.

The knowledge base contains information about a) the network topology connecting different systems, and b) the latency of each transport resource, i.e. each part of the network (see Figure 5). The Communication task involves finding a connected path in the network topology between the two systems that need to communicate. Note that this is done using nothing but the task ontology; there is no special code for finding paths through networks. Once the network path is known, the total latency of the path can be computed. This is just a matter of adding up all the component latencies. Again, this is done using the standard task machinery (in particular, using configuration artifacts specified in SWRL).


Results of Analysis (Deployment_1)

Resource Pool
RemoteLanPtpCircuitTransportMediator_Site1
RemoteSiteLan_Site1
VSTARS_v1.0_Site1
RemoteLanJmetcVpnTransportMediator_Site2
RemoteSiteLan_Site2
VSTARS_1.0_Site2
RemoteLanPtpCircuitTransportMediator_Site3
RemoteLanVpnTransportMediator_Site3
RemoteSiteLan_Site3
CSIM1_v1.0_Site4
RemoteLanVpnTransportMediator_Site4
RemoteSiteLan_Site4
Site1ToSite3PointToPointLink
TE_VPN

Solutions (score)
Solution 1 (1.0)
Solution 2 (26.0)

Failed Constraint	Description	Severity	Task	Resources
DynamicStateLatencyConstraint_Mild	Dynamic state latency causes loss of realism.	mild (1.0)	SimulateTrackingWithoutCounterMeasures	CSIM1_v1.0_Site4

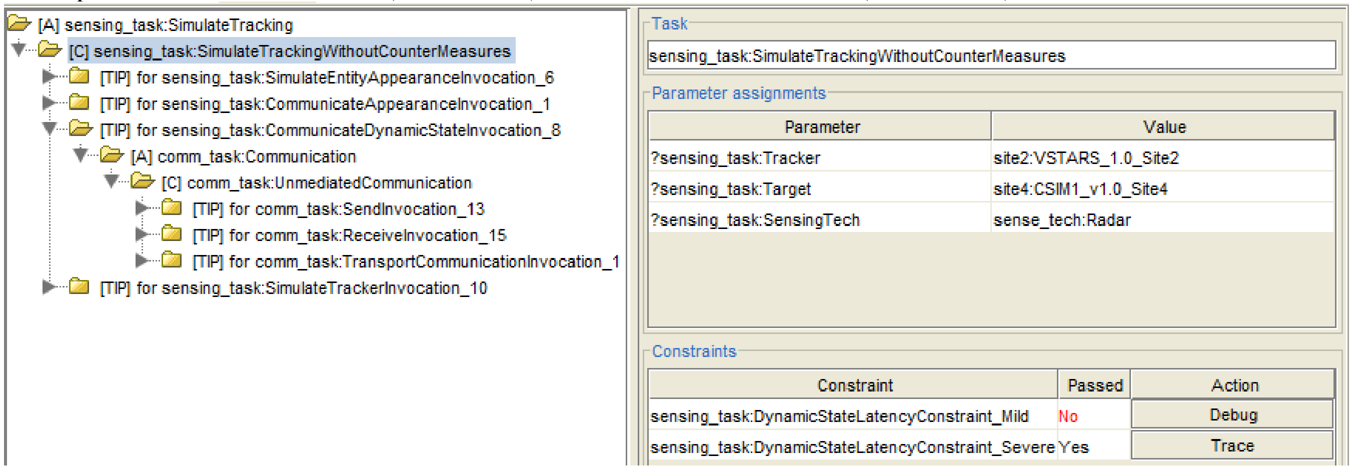
Dynamic state latency causes loss of realism.

Intersection [✓ Union]

Show label, Show name prefix, [✓ Show task type]

Figure 9: Result dialog showing solutions for the JSTARS deployment. Solution for Site-2 Virtual JSTARS selected. Site-1-specific resources are grayed out. A mild dynamic state latency warning is shown.

Figure 10 shows the synthesized taskplan for one of the solutions, including assignment of the Site-2 VSTARS to the Tracker role, a partial expansion of the task tree, and constraint results. The 195 ms latency of the path between Site-2 and Site-4 passes the severe constraint (max 300 ms) but fails the mild constraint (max 100 ms).


```
[A] sensing_task:SimulateTracking
  [C] sensing_task:SimulateTrackingWithoutCounterMeasures
    [TIP] for sensing_task:SimulateEntityAppearanceInvocation_6
    [TIP] for sensing_task:CommunicateAppearanceInvocation_1
    [TIP] for sensing_task:CommunicateDynamicStateInvocation_8
      [A] comm_task:Communication
        [C] comm_task:UnmediatedCommunication
          [TIP] for comm_task:SendInvocation_13
          [TIP] for comm_task:ReceiveInvocation_15
          [TIP] for comm_task:TransportCommunicationInvocation_1
    [TIP] for sensing_task:SimulateTrackerInvocation_10
```

Task
sensing_task:SimulateTrackingWithoutCounterMeasures

Parameter assignments

Parameter	Value
?sensing_task:Tracker	site2:VSTARS_1.0_Site2
?sensing_task:Target	site4:CSIM1_v1.0_Site4
?sensing_task:SensingTech	sense_tech:Radar

Constraints

Constraint	Passed	Action
sensing_task:DynamicStateLatencyConstraint_Mild	No	Debug
sensing_task:DynamicStateLatencyConstraint_Severe	Yes	Trace

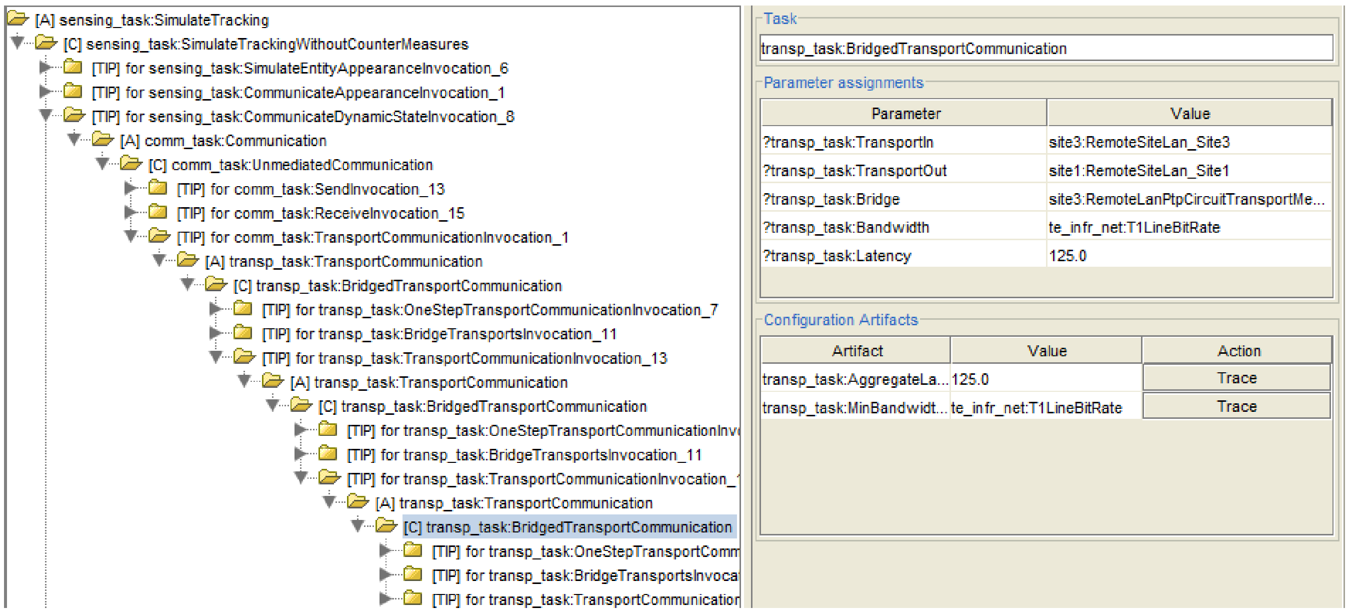
Figure 10: Result dialog showing constraint results for path between Site-2 and Site-4.

Figure 11 shows an output taskplan in which the Site-1 VSTARS is assigned to the Tracker role. The path between the Site-1 VSTARS and the Site-4 CSIM has a total latency of 310 ms, which fails the severe constraint (max 300 ms). The expansion of the task tree shows that the communication path is built up from several constituent subpaths, each of which adds to the total latency. The highlighted path between Site-1 and Site-3 has 125 ms latency.

In summary, this example shows how the task ontology framework can be used to perform rather intricate domain-specific checks, and how the Analyzer can pull together knowledge from a variety of different ontologies to find the information that it needs.

5.2 Tomahawk launch vs IADS

In our second example, we represent the simulation of a Tomahawk cruise missile launched against an SA-10 Grumble IADS air defense system. A CSIM-1 system at Site-1 simulates the Tomahawk missile, and a CSIM-2 system at Site-2 simulates the IADS.



```
[A] sensing_task:SimulateTracking
  [C] sensing_task:SimulateTrackingWithoutCounterMeasures
    [TIP] for sensing_task:SimulateEntityAppearanceInvocation_6
    [TIP] for sensing_task:CommunicateAppearanceInvocation_1
    [TIP] for sensing_task:CommunicateDynamicStateInvocation_8
      [A] comm_task:Communication
        [C] comm_task:UnmediatedCommunication
          [TIP] for comm_task:SendInvocation_13
          [TIP] for comm_task:ReceiveInvocation_15
          [TIP] for comm_task:TransportCommunicationInvocation_1
            [A] transp_task:TransportCommunication
              [C] transp_task:BridgedTransportCommunication
                [TIP] for transp_task:OneStepTransportCommunicationInvocation_7
                [TIP] for transp_task:BridgeTransportsInvocation_11
                [TIP] for transp_task:TransportCommunicationInvocation_13
                  [A] transp_task:TransportCommunication
                    [C] transp_task:BridgedTransportCommunication
                      [TIP] for transp_task:OneStepTransportCommunicationInv
                      [TIP] for transp_task:BridgeTransportsInvocation_11
                      [TIP] for transp_task:TransportCommunicationInvocation_
                        [A] transp_task:TransportCommunication
                          [C] transp_task:BridgedTransportCommunication
                            [TIP] for transp_task:OneStepTransportComm
                            [TIP] for transp_task:BridgeTransportsInvoca
                            [TIP] for transp_task:TransportCommunicatior
```

Task

transp_task:BridgedTransportCommunication

Parameter assignments

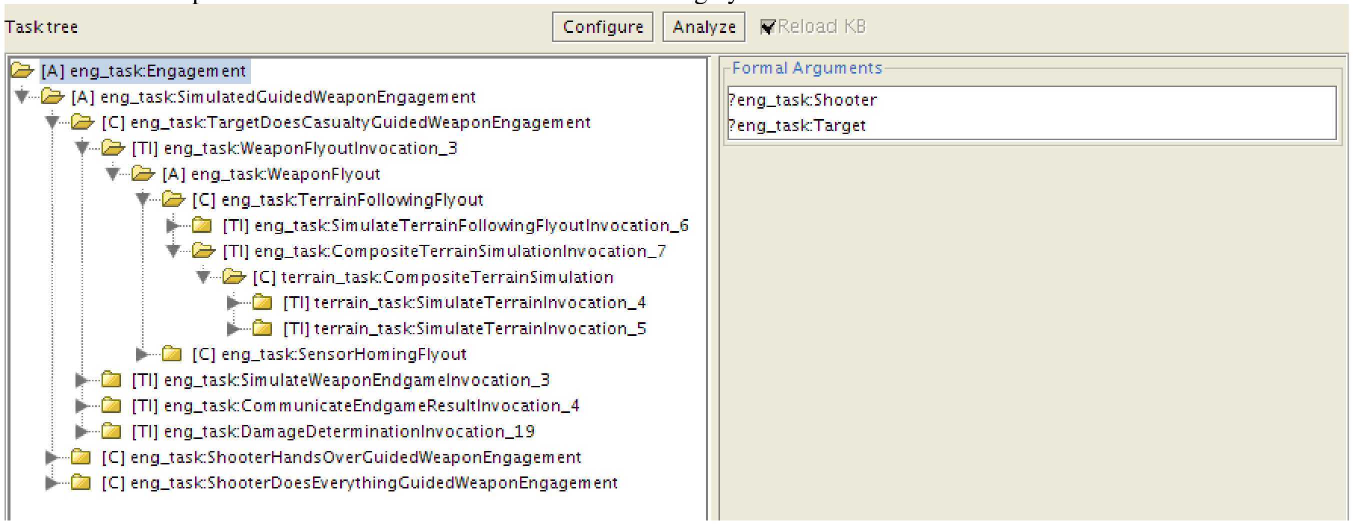
Parameter	Value
?transp_task:TransportIn	site3:RemoteSiteLan_Site3
?transp_task:TransportOut	site1:RemoteSiteLan_Site1
?transp_task:Bridge	site3:RemoteLanPtpCircuitTransportMe...
?transp_task:Bandwidth	te_infr_net:T1LineBitRate
?transp_task:Latency	125.0

Configuration Artifacts

Artifact	Value	Action
transp_task:AggregateLa...	125.0	Trace
transp_task:MinBandwidt...	te_infr_net:T1LineBitRate	Trace

Figure 11: Result dialog showing 125 ms artifact for the partial path between Site-1 and Site-3.

The task to be performed is Engagement. This is a rather comprehensive abstract task (see Figure 12). Our intent is to represent any kind of engagement as a task that refines the Engagement task. In this case, we are going to do a SimulatedGuidedWeaponEngagement. This is again an abstract task, because there are three different ways of dividing up the different component simulations between the two interacting systems.



Task tree   Configure   Analyze   ✓ Reload KB

```
[A] eng_task:Engagement
  [A] eng_task:SimulatedGuidedWeaponEngagement
    [C] eng_task:TargetDoesCasualtyGuidedWeaponEngagement
      [TI] eng_task:WeaponFlyoutInvocation_3
        [A] eng_task:WeaponFlyout
          [C] eng_task:TerrainFollowingFlyout
            [TI] eng_task:SimulateTerrainFollowingFlyoutInvocation_6
            [TI] eng_task:CompositeTerrainSimulationInvocation_7
              [C] terrain_task:CompositeTerrainSimulation
                [TI] terrain_task:SimulateTerrainInvocation_4
                [TI] terrain_task:SimulateTerrainInvocation_5
          [C] eng_task:SensorHomingFlyout
      [TI] eng_task:SimulateWeaponEndgameInvocation_3
      [TI] eng_task:CommunicateEndgameResultInvocation_4
      [TI] eng_task:DamageDeterminationInvocation_19
    [C] eng_task:ShooterHandsOverGuidedWeaponEngagement
    [C] eng_task:ShooterDoesEverythingGuidedWeaponEngagement
```

Formal Arguments

?eng_task:Shooter
?eng_task:Target

Figure 12: Engagement task, with the TargetDoesCasualtyGuidedWeaponEngagement branch expanded to some depth. Terrain following flyout requires a composite terrain simulation.

We create a taskplan, confederation, and deployment as in the JSTARS example above, and run the Analyzer. We get one solution, which shows several warnings, one of which is BigH_LittleH_Constraint, with the description "There may be a vertical error due to mismatch in vertical coordinate components (Big H/Little H)". To understand what this warning means and what caused it, we need to look more deeply into the task that is being performed. Looking back at Figure 12, we see that one of the refining tasks of SimulatedGuidedWeaponEngagement is TargetDoesCasualtyGuidedWeaponEngagement. This task involves a WeaponFlyout sub-task. WeaponFlyout is an abstract task. One of its refining tasks is TerrainFollowingFlyout, which involves a CompositeTerrainSimulation sub-task. This is a composite task with two invocations of a primitive task

SimulateTerrain. Both shooter and target must simulate terrain for the distributed engagement simulation to work. The BigH_LittleH_Constraint, the constraint that failed in this particular case, belongs in CompositeTerrainSimulation, and has the following SWRL expression:

```
vci_ok(VirtualTerrain1, VirtualTerrain2)
```

The intuitive meaning is that the VCIs of the two virtual terrains have to be compatible (i.e., the same, if required at all). This constraint is associated with ("calls") the following SWRL rule:

```
vt:spatialReferenceFrame(?vt1, ?srf1) ∧
vt:spatialReferenceFrame(?vt2, ?srf2) ∧
srf:VCI_SpatRefFrame(?srf1) ∧
srf:VCI_SpatRefFrame(?srf2) ∧
srf:verticalComponentIdentification(?srf1, ?vci) ∧
srf:verticalComponentIdentification(?srf2, ?vci)
⇒
vci_ok(?vt1, ?vt2)
```

The last line, following the implication arrow, is the rule head. The `vci_ok` predicate has two arguments: ?*vt*1 and ?*vt*2, corresponding to the two virtual terrains that are compared. Note that SWRL requires all variables to be prefixed with a question mark. The first two lines of the rule body find the SRFs asserted for each terrain. The next two lines check whether the SRFs are both types that have a VCI. If so, the final two lines check whether they have the same VCI. The Prolog rule engine tries out all possible bindings for free variables from the KB and returns "true" if there is a combination that satisfies all statements in the rule body. If false, the results window displays the warning text that is asserted for the message property of the constraint (i.e., "There may be a vertical error due to mismatch in vertical coordinate components (Big H/Little H)").

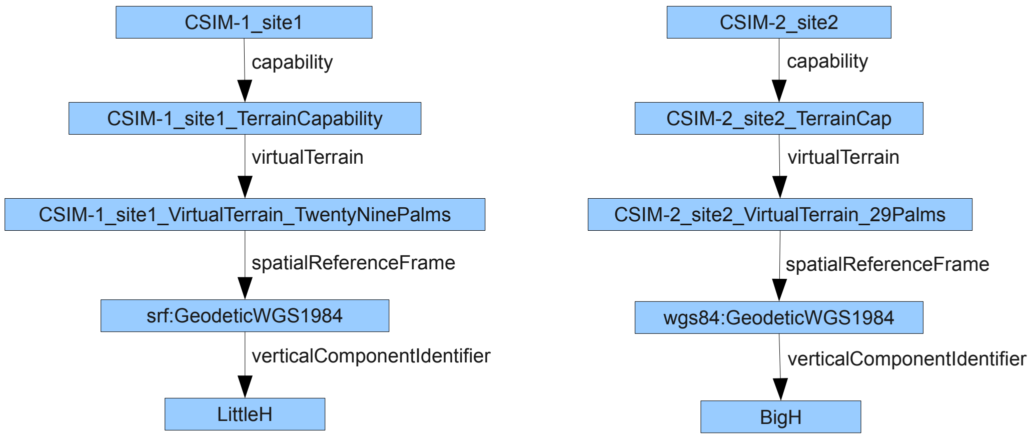
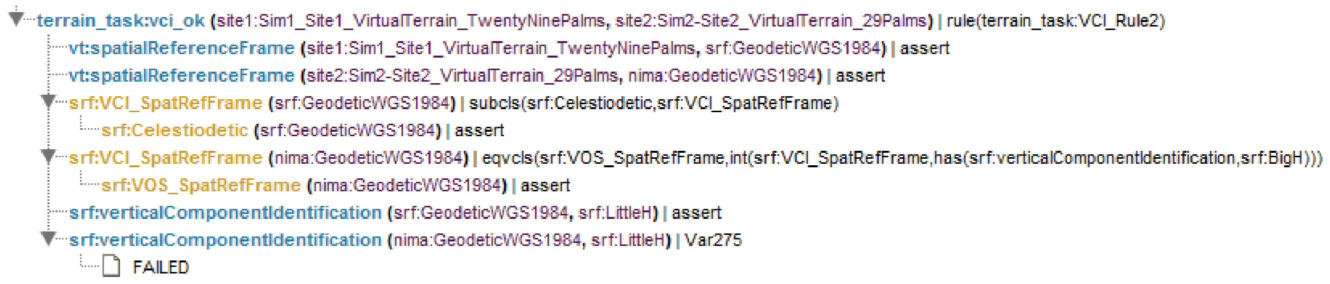


Figure 13: Virtual terrains of CSIM-1 and CSIM-2; vertical component identifier mismatch.

We now consider the facts in the knowledge base about the two simulation systems that caused the constraint to fail. The SimulateTerrain task requires a TerrainSimulationCapability. Both of the participating systems (CSIM-1 and CSIM-2) have such a capability (see Figure 13). One of the properties of TerrainSimulationCapability is virtualTerrain. In this case, both CSIM-1 and CSIM-2 have their own virtual terrains for the 29 Palms location, which is the simulated location for the engagement. One of the properties of VirtualTerrain is spatialReferenceFrame. The SRF ontology was described in Section 3.1. As we can see in Figure 13, the CSIM-1 terrain KB identifies LittleH as its VCI, and the CSIM-2 terrain KB has BigH. The difference between the two can be significant, and if a mismatch is not handled (by translating the coordinates between the two systems), the effects might include ordnance exploding underground or tanks flying in the air, as seen by one of the systems. Interestingly, both the spatial reference frames used in this case call themselves "WGS1984". However, notice (in Figure 13) that the individuals are different: One is from the srf.owl ontology, and the other from the wgs84.owl ontology. This mirrors the case in the real world: the original NIMA WGS1984 specified a "Big H" VCI, whereas the version of WGS1984 specified in the SEDRIS standard has a "Little H" VCI.



```
terrain_task:vci_ok (site1:Sim1_Site1_VirtualTerrain_TwentyNinePalms, site2:Sim2-Site2_VirtualTerrain_29Palms) | rule(terrain_task:VCI_Rule2)
    vt:spatialReferenceFrame (site1:Sim1_Site1_VirtualTerrain_TwentyNinePalms, srf:GeodeticWGS1984) | assert
    vt:spatialReferenceFrame (site2:Sim2-Site2_VirtualTerrain_29Palms, nima:GeodeticWGS1984) | assert
    srf:VCI_SpatRefFrame (srf:GeodeticWGS1984) | subcls(srf:Celestiodetic,srf:VCI_SpatRefFrame)
        srf:Celestiodetic (srf:GeodeticWGS1984) | assert
    srf:VCI_SpatRefFrame (nima:GeodeticWGS1984) | eqvcls(srf:VOS_SpatRefFrame,int(srf:VCI_SpatRefFrame,has(srf:verticalComponentIdentification,srf:BigH)))
        srf:VOS_SpatRefFrame (nima:GeodeticWGS1984) | assert
    srf:verticalComponentIdentification (srf:GeodeticWGS1984, srf:LittleH) | assert
    srf:verticalComponentIdentification (nima:GeodeticWGS1984, srf:LittleH) | Var275
        FAILED
```

Figure 14: Debugging trace that shows why the Big H/Little h constraint failed.

Our system includes a debugger that can automatically produce explanations of constraint failures, along similar lines as in the previous paragraph. Figure 14 shows the debugger's output for the Big H/Little h constraint. Each line in the trace shows a formula that the reasoner is attempting to prove. The top-level formula is the constraint itself, instantiated with the particular virtual terrains used in this example. The indented entries below any entry show formulas which, if proven, prove the formula above them. The end of each line shows, after a `|` character, a justification for deriving a parent node from its child nodes. In this example, the top-level justification is the VCI rule shown above. Justifications for the child nodes are assertions (i.e. facts in the KB), a subclass axiom, and an equivalent class axiom. We can see from the trace that the reasoner was able to prove all the sub-formulas of the VCI rule except the last one, i.e. it cannot prove that `nima:GeodeticWGS1984` has a little h SRF, which would required in order to match the VCI of the other SRF.

This is a good example of the kind of check the Analyzer can perform. It is rather intricate and goes to some depth into the knowledge base, using expert knowledge about spatial reference frames and precise descriptions of the systems involved. Without a tool like the Analyzer, such a check would be tedious, time-consuming, and easy to overlook.

To recap what happened: we specified only that we wanted a simulated guided weapon engagement from CSIM-1 to CSIM-2. The Analyzer decided, by looking at the task and resource descriptions, that

1. The engagement could be performed only as a TargetDoesCasualtyGuidedWeaponEngagement,
2. which required a WeaponFlyout task, and in this case it could only be done as a TerrainFollowingFlyout,
3. which required terrain simulations on both shooter and target systems.
4. Those terrain simulations needed to have virtual terrains with SRFs with compatible VCIs.
5. The actual terrain simulations did not have virtual terrains with SRFs with compatible VCIs.

.

6 COMMUNITY INVOLVEMENT

Ontologies, such as those used in the work described here, are best conceived as knowledge resources not for a single project or system, but for a Community of Interest (CoI) or Community of Practice (CoP). With the ongoing involvement of an appropriate CoI or CoP, an ontology can be evolved and maintained in such a way as to effectively enable the sharing of information across the community, support a variety of related uses, and contribute to the implementation of a range of applications. Further, ontology content needs to be informed by the experience and expertise of SMEs. Because of the great value of, and limited access to, such expertise, it makes sense to capture and share it across a community. For these reasons, the ONISTT and ANSC projects have endeavored to draw on appropriate SMEs and engage the relevant communities. Here, we describe three settings in which these outreach activities have occurred.

6.1 InterTEC

The Joint Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (JC4ISR) Interoperability Test and Evaluation Capability (InterTEC) is a distributed, integrated test environment for testing interoperability between C4ISR systems. InterTEC is designed to provide the capability to construct, control, instrument, capture data from, and analyze tests of interoperability in operationally relevant scenarios. InterTEC supports developmental testing, operational testing, and interoperability certification, and employs live, virtual, and constructive resources. (Additional information about InterTEC can be found at http://jitc.fhu.disa.mil/intertec/index.html.)

The ANSC project has partnered with InterTEC to identify important use cases for the Analyzer, and to explore the requirements for a successful transition of this technology into ongoing usage. InterTEC provides an invaluable source of information about tasks and resources, and how they are employed in large-scale test events. In many cases, InterTEC planning documents contain task and resource descriptions that are suitable for formalization in Analyzer KBs. In addition, conversations with InterTEC SMEs can bring to light interoperability problems that have arisen in actual testing practice, such as communications mismatches and fair fight issues. These conversations enable us to capture the knowledge that is needed to identify and rectify these kinds of problems. Whereas in current practice this knowledge is recorded in SMEs' memories, notebooks, and textual documents, ANSC methodology results in its formalization in knowledge bases.

As noted in Section 7, one of our priorities in the current phase of our work is to craft a GUI that is readily usable by InterTEC SMEs, without requiring knowledge representation expertise.

The examples presented in Section 5 are based on InterTEC resources, events, and experiences.

6.2 Joint Training Enterprise Ontology Workshop

The Joint Training Enterprise Ontology Workshop (JTEOW) in August 2009 brought together SMEs in several technical domains important to LVC interoperability. JTEOW was designed to support several initiatives that are addressing the persistent challenges of composing net-centric LVC training, testing, and operational environments. These initiatives include ONISTT, the Department of Defense (DoD) Net-Centric Data Strategy (NCDS), and the Joint Composable Object Model (JCOM).

NCDS provides a key enabler for DoD transformation by establishing the foundation for managing DoD information in a net-centric environment: ensuring data is visible, accessible, and understandable via enterprise and/or community-defined vocabularies and services. Implementation of the DoD Net-Centric Data Strategy (NCDS) has reached a point where a wider work force appreciation of the concepts, principles, and enabling products is desirable to further expand training and testing applications. (More information about NCDS can be found at https://metadata.dod.mil/mdr/ns/ces/techguide/net_centric_data_strategy_ncds_goals.html.) JCOM is discussed in Subsection 6.3 below.

6.2.1 JTEOW Objectives

The objective of the JTEOW was to identify the key concepts for each domain and their relationships, and to define rules constraining interoperability in particular scenario contexts. A central goal of JTEOW was to capture key elements of the specialized technical knowledge of participants to serve as the basis for building prototype domain ontologies that can be used to describe the capabilities of LVC resources and the capabilities that are needed to perform operational, training, and testing tasks.

The JTEOW was designed for mutual benefit of the presenters, who are experts in ontology development and application, and the participants, each of whom is a subject matter expert (SME) in one or more of the workshop topic areas. The presentation team included members of the ONISTT and JCOM projects.

During the JTEOW planning phase, members of the ONISTT and JCOM teams explored the potential for the JCOM project to leverage the ONISTT design and and application of ontologies related to LVC integrating architectures. As a result, the objectives of the Workshop were augmented to solicit information from SMEs that would help construct and refine ontologies that are common to ONISTT and JCOM.

In return for their help in providing information needed to advance ONISTT and JCOM ontologies, the JTEOW was designed to offer participants an introduction to Semantic Web technologies, with examples of how they are being applied to LVC technical domains and interoperability issues. It is expected that as these technologies mature, they will become increasingly valuable and widespread in application.

6.2.2 Approach and Domain Areas

The following broad domains were identified as candidate working group topics: communication architectures, sensors, environment, network infrastructure, munitions behaviors and effects, and logistics. Based on SME level of interest and availability, working groups were formed on the topics of communication architectures, sensors, environment, and logistics. Participants were expected to have expert knowledge in some relevant LVC technical or interoperability area, but no prior experience with knowledge representation technologies was assumed.

In each domain area, interoperability use cases were developed prior to the workshop, and were used to guide the discussions in the domain working group. The use cases were used to elicit an understanding of the concepts that are needed to formalize the knowledge the Analyzer can use to solve interoperability problems. Each working group produced an initial high-level OWL ontology (or a set of extensions to an existing ontology). Selected central concepts were explored and formalized in greater depth, and domain areas were prioritized for further development. After the workshop, the JTEOW working group findings were incorporated into OWL ontologies and SWRL rules in the context of the ONISTT and JCOM projects.

The content of the candidate domains (including the two domains that were not selected) included the following:

Communication architectures: This topic included the ONISTT communication ontology as a possible starting point, including proposals for its modification or extension. This topic also investigated metamodels for TENA, HLA, CTIA and DIS. It considered the expression of metamodels in ontological terms, and the definition of mappings between them, and the relations between metamodels and the general communications ontology. This topic also considered the semantics underlying related elements of selected DIS/HLA/TENA/CTIA object models.
Sensors: This topic addressed the capabilities of various types of sensors (radar, IR, etc.) and sensor platforms (e.g., AWACS aircraft). Electronic observability characteristics, such as radar cross section or electronic signature simulations and stimulations, were also part of this topic. Another area of interest was the fidelity of RF emission sensing systems and their ability to distinguish between multiple signals such as low probability of intercept (LPI), frequency hopping, or intermittent transmissions.
Environment: This topic included terrain, natural and cultural features, and/or atmospheric and environmental conditions. Formal models to capture correlation between simulations, or between live and simulated terrain, were of interest, as well as specifics about terrain modeling, supported terrain servers, services and formats, and spatial reference frames used in representations. Environmental condition topics included weather models, oceanographic conditions (e.g., wave heights or currents), and luminance conditions (e.g., smoke, day/night).
Network Infrastructure: Domain areas of interest in this topic included models of infrastructure elements such as routers and their configurations, supported communication types, Tactical Local Area Network Encryption (TACLANE) configuration, VPN and firewall configuration, and network parameters such as latency, delay, and loss rate.
Munitions: This topic included weapons and countermeasures behavior and effects. Concepts considered for formalized description included location of simulation (e.g., shooter or target), fidelity of fly-out and fuzing simulations (e.g., missile projector and system operation), and fidelity of conditions affecting explosion lethality at target location. Countermeasures under consideration included Radar Warning Receiver (RWR),

radar jamming simulations and stimulations, and anti-missile countermeasure simulations (e.g., fuze jammer, tow or free-fall RF decoy, chaff dispensation).
Logistics: This topic included systems interoperability issues that are specific to logistics and logistics training, as well as the formalization of the more general knowledge needed to support procurement, supply chain management, inventory maintenance, and other traditional logistics concerns.

6.2.3 Future Workshops and Working Groups

Future workshops and working group activities are under consideration, with the following identified as possible objectives:

1. Introduce semantic technologies and their application to joint training and testing environments.
2. Provide hands-on training in methods for creating and extending ontologies.
3. Provide hands-on training for populating KBs with factual information.
4. Provide hands-on training for querying and exploring KBs to find stored information.
5. Decompose a broad domain, capability, or task topic area into more narrowly defined topics that will be the subsequent focus of specific ontology development efforts.
6. Develop specific new ontologies.
7. Add information to specific KBs.
8. Facilitate technical exchange with the principals of related initiatives to explore potential sharing or synergy.

6.3 Joint Composable Object Model

The Joint Composable Object Model (JCOM) project (Lutz et al. 2009) is developing an Architecture Neutral Data Exchange Model (ANDEM) as a superset of model elements for the four major LVC interoperability communication architectures, that is, DIS, HLA, Test and Training Enabling Architecture (TENA), and Common Training Instrumentation Architecture (CTIA). One of the goals of JTEOW has been to develop ontologies that will provide a semantic foundation for ANDEM. ANDEM, in turn, can help construct and validate translators between content models in these architectures. As with natural languages, such translation can be imperfect because the communication architectures do not have identical expressivity. The ontologies underlying ANDEM will also provide a basis for understanding what is lost or subtly transmuted in translation.

JCOM project leaders report that the JTEOW tutorials and WG sessions succeeded in demonstrating the value of grounding JCOM in an ontological approach, and that significant progress has been made in developing ontologies based on the ANDEM metamodel.

7 CURRENT AND FUTURE WORK

7.1 Requirements Analysis

ONISTT was designed to assess interoperability among existing systems and services, particularly when used in new combinations for purposes that differ in some respects from those that were pre-envisioned in the system specifications. Recently we have adapted the core ontologies and the Analyzer to precisely and unambiguously define requirements for new systems, and to assess whether a particular system meets its specifications.

We are applying this approach to requirements for Joint training events within specific land, air, and maritime environments. General requirements are parameterized for application to particular areas, for example, interactions among live air training systems and ground-based simulations. The requirements ontology includes a mechanism for specifying the precise operating conditions under which a system must perform, with traceability to the operational missions that motivate these performance levels. We intend to apply the Analyzer to KBs documenting a system's design or test performance to determine whether, and to what degree, it meets the specified requirements. If a system falls short in some particulars, the Analyzer will identify missions suited to its projected or demonstrated performance.

7.2 Knowledge Base, Rules and Results Visualization

In recent phases of ANSC, we have emphasized fair fight issues – for example, terrain correlation, sensor and engagement simulations, and time synchronization. We are also improving GUI presentation methods so that SMEs who are not ontology experts can readily understand and navigate KBs and explore the results of interoperability and test coverage analysis. It is particularly difficult to explain why something failed, because there are usually an extremely large number of ways to fail, but only a handful of ways to succeed. Other technical challenges addressed by current ANSC tasking are:

- Ensuring that the time and space complexity of the ONISTT automated reasoning toolset scales to handle the full complexity of large training and testing events.
- Extending assessment of test coverage to include interdependent multi-dimensional test parameters. Alternative sets of resource assignments, each of which gives partial test coverage, will be ranked.
- As part of the NR-KPP Solutions Architecture Toolkit (NSAT) project, mapping Joint Mission Threads expressed as DoDAF Metamodel (DM-2) models to the ONISTT Task ontology, and automatically extracting threads represented in accordance with the DM-2 Physical Exchange Specification (PES).
- Developing a general ontology for specifying the properties of simulations and their level of fidelity relative to the capabilities of real world entities and systems.

The number and diversity of ontologies required for an effort like ours makes broad community participation in creating and maintaining them essential. The JTEOW and our work with the InterTEC SMEs is a start. However, the promise of semantic harmonization will not be realized if a multitude of ontologies are developed in isolation by small specialized communities. Although is not realistic to expect that universal ontologies can be designed to suit all purposes, it is reasonable to minimize redundant proliferation. As well-designed fundamental domain ontologies become more abundant, some based on existing paper standards, we anticipate using and extending them instead of creating our own.

8 CONCLUSIONS

Military training and testing events are highly complex affairs, involving numerous live instrumentation and M&S systems as well as infrastructure resources. Our approach to facilitating such events is ambitious: describe the systems and requirements in great detail using ontologies, and use automated reasoning to automatically find problems and identify potential solutions or mitigating options.

Developing standardized, authoritative ontologies can reduce the high degree of ambiguity and diversity we find in the description of systems, the tasks they need to perform, the environments in which they operate, and the architectures designed to facilitate their working together. However, it is neither possible nor desirable to retire everything that is old, or to constrain everything that is new to co-evolve in lockstep. Heterogeneity is here to stay. In this paper, we describe an approach that makes it easier to cope.

The long-term goal is to provide a complete system that is usable by military training and testing experts who are not necessarily knowledgeable in Semantic Web technologies. For such a transition to be successful, several different Semantic Web technologies and research areas need to progress further. The scale and distributed nature of the necessary ontology development will require significant improvement in ontology engineering approaches and tools.

ACKNOWLEDGMENTS

ONISTT is sponsored by ODUSD/R/RTPP (Training Transformation). ANSC is sponsored by USD/AT&L-TRMC (S&T Portfolio). JTEOW is jointly sponsored by USJFCOM JWFC and ODUSD/R/RTPP.

REFERENCES

Bacchelli, F., A.-C. Boury-Brisset, A. Isenor, S. Kuehne, B. M. Reif, J. Miles, V. Mojtahedzadeh, R. Poell, R. Rasmussen, A. Uzanali, and M. Wunder. 2009. Semantic interoperability. Technical Report RTO IST-075 - RTG-034, NATO Research and Technology Organization.

Booch, G. 1993. *Object-oriented analysis with design and applications (2nd ed)*. Addison-Wesley, Boston, MA.

Dahman, J. 1999, October. High level architecture interoperability challenges. In *Proc. NATO Modeling & Simulation Conference*. Norfolk, VA: NATO RTA Publications.

Director, IS & NNEC ICT 2009. NNEC Information Portal. <transnet.act.nato.int/WISE/Informatio> [accessed September 30, 2009].

DoD CIO 2007, April. DoD Architecture Framework, Version 1.5, Volume I. <www.defenselink.mil/cio-nii/docs/DoDAF> [accessed September 30, 2009].

DoD CIO 2009a. DoD CIO/OASD(NII) Homepage. <www.defenselink.mil/cio-nii> [accessed September 30, 2009].

DoD CIO 2009b. DoD Metadata Registry and Clearinghouse. <metadata.dod.mil> [accessed September 30, 2009].

Elenius, D., R. Ford, G. Denker, D. Martin, and M. Johnson. 2007. Purpose-aware reasoning about interoperability of heterogeneous training systems. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, Volume 4825 of *Lecture Notes in Computer Science*, 750–763: Springer.

Elenius, D., D. Martin, R. Ford, and G. Denker. 2009. Reasoning about Resources and Hierarchical Tasks Using OWL and SWRL. In *The Semantic Web, 8th International Semantic Web Conference, ISWC 2009, Washington, DC, October 25-29, 2009*.

Ford, R., D. Martin, M. Johnson, and D. Elenius. 2009. Ontologies and tools for analyzing and synthesizing LVC confederations. In *Winter Simulation Conference*, 1387–1398. Austin, TX.

Grosof, B. N., I. Horrocks, R. Volz, and S. Decker. 2003. Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, 48–57. New York, NY, USA: ACM.

Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, at <www.w3.org/Submission/2004/SUBM-SWRL-20040521> [accessed September 30, 2009].

ISO 1995. *Guide to the expression of uncertainty in measurement*. Geneva: ISO.

ISO 2006. *Information technology Spatial Reference Model (SRM)*. Geneva: ISO.

ISO 2008. *Quantities and units part 13: Information science and technology*. Geneva: ISO.

Kasputis, S., I. Oswalt, R. McKay, and S. Barber. 2004. Semantic descriptors of models and simulations. In *Simulation Interoperability Workshop, September 19-24, 2004*. Orlando, FL. 04F-SIW-070.

Klyne, G., and J. J. Carroll. 2004, February. Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C. <www.w3.org/TR/2004/REC-rdf-concepts-20040210> [accessed September 30, 2009].

Knublauch, H., R. Fergerson, N. Noy, and M. Musen. 2004. The Protégé OWL plugin: An open development environment for Semantic Web applications. In *Proc. 3rd Intern. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 2004*, ed. S. McIlraith, D. Plexousakis, and F. van Harmelen, 229–243: Springer. LNCS 3298.

Lutz, R., J. Wallace, A. Bowers, D. Cutts, P. Gustavson, and W. Bizub. 2009. Common Object Model Components: A First Step Toward LVC Interoperability. In *Simulation Interoperability Workshop, March 23-27, 2009, 09S-SIW-031*. San Diego-Mission Valley, CA: Simulation Interoperability Standards Organization.

Martin, D., M. Burstein, D. McDermott, D. McGuinness, S. McIlraith, M. Paolucci, E. Sirin, N. Srinivasan, and K. Sycara. 2007, September. Bringing semantics to web services with OWL-S. *World Wide Web Journal* 10 (3): 243–277.

McGuinness, D. L., and F. van Harmelen. 2004. OWL Web Ontology Language Overview. World Wide Web Consortium (W3C) Recommendation, at <www.w3.org/TR/owl-features> [accessed September 30, 2009].

Miller, J. A., and G. Baramidze. 2005. Simulation and the semantic web. In *WSC '05: Proceedings of the 37th Conference on Winter Simulation, December 4-7, 2005*, 2371–2377. Orlando, Florida: Winter Simulation Conference.

OASIS 2006, October. *Reference Model for Service Oriented Architecture*. OASIS. <docs.oasis-open.org/soa-rm/v1.0> [accessed July 17, 2011].

Object Management Group 2009, May. Ontology Definition Metamodel, Version 1.0. <www.omg.org/spec/ODM/1.0> [accessed September 30, 2009].

Preece, A., M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, and T. L. Porta. 2007. An Ontology-Based Approach to Sensor-Mission Assignment. In *1st Annual Conference of the International Technology Alliance (ACITA), Maryland, USA*.

Silver, G. A., O. A.-H. Hassan, and J. A. Miller. 2007. From domain ontologies to modeling ontologies to executable simulation models. In *WSC '07: Proceedings of the 39th Conference on Winter Simulation, Washington D.C.*, 1108–1117. Piscataway, NJ, USA: IEEE Press.

Thompson, A., and B. N. Taylor. 2008. Guide for the use of the international system of units (SI). Technical Report Special Publication 811, National Institute of Standards and Technology, Gaithersburg, MD.

Tolk, A., and J. Muguira. 2003. The Levels of Conceptual Interoperability Model (LCIM). In *Proceedings IEEE Fall Simulation Interoperability Workshop, 03F-SIW-007*: IEEE CS Press.

Tolk, A., C. D. Turnitsa, and S. Y. Diallo. 2008. Implied ontological representation within the levels of conceptual interoperability model. *Intelligent Decision Technologies* 2 (1): 3–19.

Turnitsa, C. D. 2005. Extending the Levels of Conceptual Interoperability Model. In *Proc. IEEE Summer Computer Simulation Conference*: IEEE CS Press.

Vick, S., S. Murphy, R. Cost, and W. Bethea. 2006, March. An agent based approach for model composition. In *Simulation Interoperability Workshop, 07S-SIW-089*. Simulation Interoperability Standards Organization.