27th Annual INCOSE International Symposium (IS 2017)
Adelaide, Australia, July 15-20, 2017

Semantically-enabled Model-based Systems Engineering of Safety-critical Network of Systems

Leonard Petnga
University of Maryland
College Park, MD 20742, USA
+1 301 405 9926
lpetnga@umd.edu

Mark Austin
University of Maryland
College Park, MD 20742, USA
+1 301 405 6627
austin@umd.edu

Mark Blackburn
Stevens Institute of Technology
Hoboken, NJ 07030, USA
+1 561 637 3452
mark.blackburn@stevens.edu

Abstract. This paper describes a novel approach to the development and integration of semantics to the model-based systems engineering and operation of safety-critical network of systems. Engineering models work directly with formal domain and meta-domain (especially time and space) knowledge that are determinate, provable (ambiguity free) and executable. Knowledge is encoded as semantic blocks, which are an integration of ontologies, rules, and communication and computation interfaces. These concepts are exercised in a collision avoidance problem involving autonomous agents at a traffic intersection.

Introduction

Problem Statement. The rapidly–growing complexity, interconnectivity, and cost of systems in civilian and military domains is driving the need for new design methodologies and tools to model and analyze the performance of systems of systems (SoSs) and complex cyber-physical systems (CPS) at the engineering level. SoSs are a *set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities* (DoD, 2004). In many cases, relationships among the participating SoS entities might only be weakly coupled. CPSs, on the other hand, are characterized by a network of computational and physical elements, seamlessly integrated and tightly coupled. Examples of the latter include self-driving vehicles, unmanned aerial vehicles (UAVs), mobile autonomous robots and unmanned ground systems (UGSs). Despite their smartness, standalone CPSs are often ill-equipped to tackle the complexity of safety-critical missions, especially in the military. Instead, their association with other systems (including humans) leads to networks of CPS, with the collective behavior emulating a SoS. We introduce the term "network of systems'' (NoSs) to designate the aggregation of SoSs and CPSs operating in a networked environment.

Ongoing projects such as the US Coast Guard Integrated Deepwater System (Peckenpaugh, 2016) and air traffic control and management (FAA, 2016) demonstrate both the need for and the transformative power of adopting NoS perspective in addressing complex critical military and civilian undertakes. However, research challenges are numerous and diverse. Nevertheless, there is a consensus on the need for novel domain-specific modeling and simulation methodologies and tools (Crossley, 2004), (Maier, 2005), (Kalawsky, 2013). Specific and unambiguous domain knowledge and their proper exchange and handling across various engineering areas of expertise are critical to appropriate components modeling and decision-making at all levels. This includes the proper elicitation of system requirements, which, along with erroneous assumptions about the operation of a control/computer system, has been shown to be the leading cause of software–related accidents in modern systems (Leveson, 2004). The ability of the system to achieve/demonstrate the expected level of capability, performance, resilience while remaining safe is at stake, especially when there are humans-in the loop. One near-term goal is to equip System Engineers (SEs) with rigorous approaches for assessment of NoS safety and resilience while carrying out complex tactical operations in challenging/hostile environments with little to no human interactions.

Scope and Objectives. This paper reports on the development of a semantically-enabled approach to the model-based systems engineering (MBSE) of safety-critical NoSs. We will look into the family of NoSs for which correct requirement elicitation and operational capability involve proper handling of spatio-temporal constraints. A framework for the development of ontology-based semantics for domains and meta-domains along with their rules and their integration with engineering models will be introduced and described. The problem of collision avoidance (i.e. safety) of autonomous mobile vehicles at a non-signalized intersection will illustrate the application of these concepts.

Model-Based Systems Engineering for Network of Systems. Figure 1 shows a simplified architecture of a NoS. Each component system has its own independent behavior and will communicate with others via an interface (Wi), if and when needed, for the purpose of the system level mission. System (Si) behavior and structure are modeled as graphs that dynamically evolve in response to events and obey to a set of domain-specific rules. As a directed NoS, the responsibility of meeting the system mission or goal is assigned to a manager (Wc). However, each component is smart and can process information and make (local) decision too.

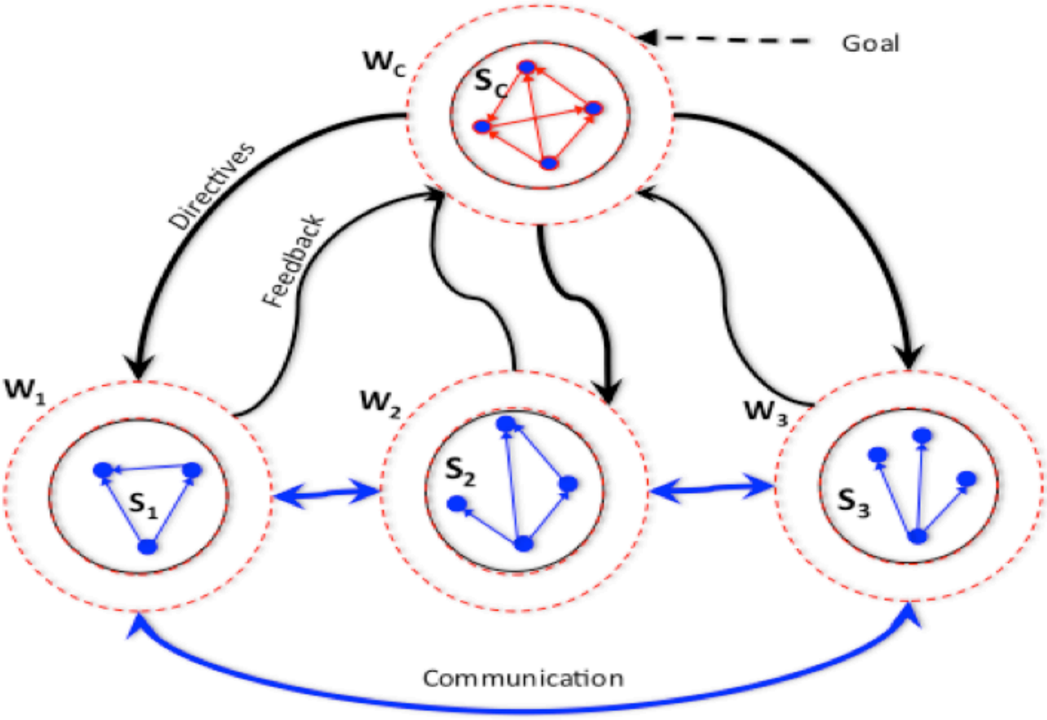


Figure 1. Simplified graph-based model of a (directed) NoS .

As an illustration, we consider a network of connected autonomous vehicles that must safely cross a non-signalized intersection managed by an intersection manager (IM) as shown in Figure 2. The IM is the manager in this directed NOS and cars have their own individual, independent trajectory and behavior (see Figure 2 (a)). However, they have to communicate and coordinate with each other and, apply traffic rules imposed by the IM to ensure the proper space-time separation needed to avoid collision while crossing the intersection (see Figure 2 (b)). The network structure and configuration are dynamic as vehicles enter and leave the intersection area.

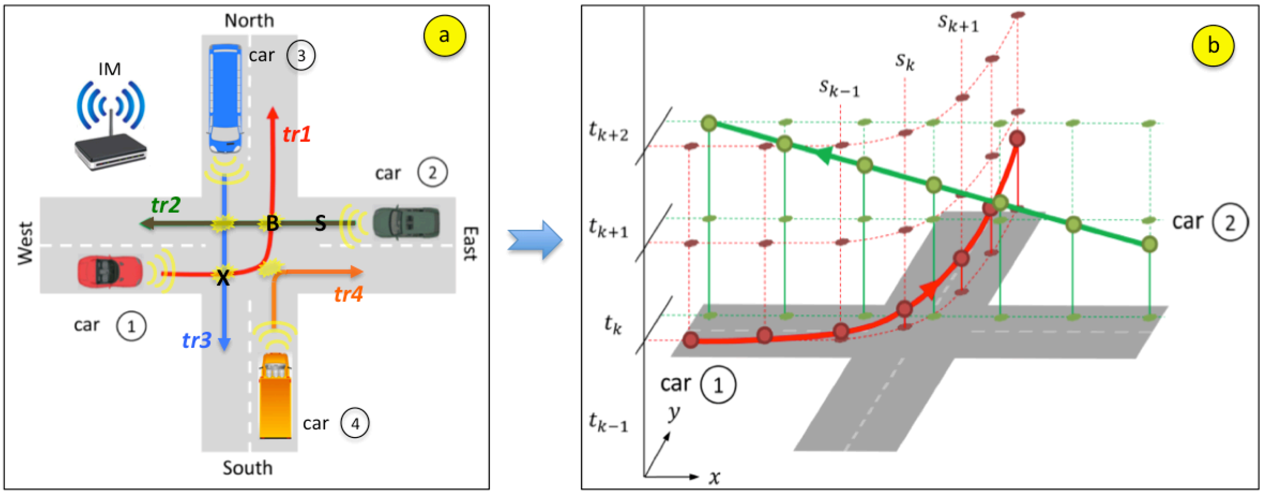


Figure 2. Schematics for network of autonomous vehicles crossing a lightless traffic intersection and example space-time trajectories that must be separated (Adapted from (Wuthishuwong, 2013))

Achieving correct-by-design NoS requires effective dealing with multiple physics, heterogeneous, (spatio-temporally) distributed, connected components, which could be tightly or loosely integrated, dynamic and/or static. Weak semantics of leading languages such as the Unified Modeling Language (UML) and System Modeling Language (SysML), coupled with difficulties in achieving separation of concerns and robust system integration make present-day MBSE procedures inadequate in dealing with these challenges. Researchers have advocated approaches using logic to strengthen UML/SysML semantics through language retrofitting (Graves, 2011) and ontological profiling (Rouquette, 2010) which creates formal language representations in system modeling environments (tools). Despite their demonstrated benefits in addressing inconsistencies and redundancies in models, they are not intuitive and require strong knowledge in logic. Moreover, they do not support effective analysis of system-level properties.

There is a strong need for modeling abstractions to be broad in scope, act across multiple stages of system development and formality. Our perspective is that, development of NoSs can benefit from ongoing work in CPS thanks to the multiple overlaps (intelligence, integration/coordination, distributed behaviors, heterogeneous components, etc.) between the fields. The underlying tenet of our research is that future methodologies for design and analysis of NoSs need to employ semantic descriptions of application domains, and use ontologies and formal reasoning to adaptively enable validation of requirements, communication (or mappings) among multiple disciplines and models, and assessment of system capability in the face of permanent uncertainty.

Semantic Web: Theory, Models, Languages and Tools

Semantic Web Vision and Network of Systems. In his conceptualization of the World Wide Web (late 1980s), Tim Berners-Lee identified two main goals: (1) To make the Web a collaborative medium and, (2) To make the Web understandable and automatically processable by machines (Berners-Lee, 2001). The first part of this vision has come to pass – in today's Web, machines retrieve and render information presented in formats that are readable and interpretable by humans. The Semantic Web (second part) aims to produce a semantic data structure, which allows machines to access and share information, thus constituting a communication of knowledge between machines, and automated discovery of new knowledge. This is an ongoing effort that will greatly benefit the development of NoS given the current trends towards machines as full system components and software/computers increasingly being responsible of more and more system functionality. There is a need for mechanisms (i.e., markup languages) enabling the introduction, coordination, and sharing of the formal semantics of data, as well as an ability to reason and draw conclusions (i.e., inference) from semantic data obtained by following hyperlinks to definitions of problem domains (i.e., so-called ontologies).

Logic-based Semantics and Ontologies for Reasoning. To be used in reasoning, data need to be backed by semantics, which provided clear, unambiguous interpretation with respect to the knowledge of the associated domain and the context of use. To address this challenge, researchers have developed several knowledge representation formalisms such as Semantic Networks, Frame Systems, Description Graphs and Logic-based infrastructure. Because of their sound mathematical foundations (well defined semantics and proven reasoning algorithms) and decidability as fragment of first order logics - all features critical for reasoning tasks - we opt for description logics (DL) formalisms (Baader, 2008). In DLs, domains are described in term of concepts and roles assigned to or among them. Terminological axioms constitute the *TBox* or $\mathcal{T}$ (definitions) and it's complemented with assertional axioms ABox or $\mathcal{A}$ (individuals). Universal ($\forall$), existential ($\exists$), intersection ($\sqcap$), Union ($\sqcup$) and negation ($\neg$) operators support the restriction needed to make the language decidable at low complexity. Interpretations $\mathcal{I}$ are functions that map concepts (C) and binary relationships (R) and they provide the required semantics to the domain being described. A summary of description logic concepts constructors can be found in (Horrocks, 2007).

DLs provide the mathematical foundations on top of which machine and human ontological languages such as the web ontology language (OWL) can be built in a systematic way (Patel-Schneider 2004). They enable the creation of ontologies, which are engineering artifacts that provide explicit specifications of the intended meaning of a vocabulary used to describe a given domain. In ontologies, DL concepts are represented as classes while roles are captured as properties and relationships. Ontological models provide semantic meanings that enrich the way models can be branched and integrated across domains of knowledge automatically. These capabilities are critical to NoS given the strong need to understand intricate relationships spanning multiple domains and components and their ultimate affects on system level functionality. However, (Patel-Schneider, 2004) points that the basic DL ($\mathcal{ALC}$) requires some extensions to efficiently support OWL, as suitable ontological language for such semantics capabilities. In (Petnga, 2016), we've identified and described the necessary extensions, leading to $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$ DLs (respectively mapped to OWL1-DL and OWL2 DL) as appropriate logic-based formalisms for our work. Moreover, $\mathcal{SROIQ}$ DL is decidable (Horrocks, 2006). Thus, OWL2 DL, which we use in this work, is computationally decidable.

Working with Semantic Web Technologies and Jena. Figure 3 illustrates the technical infrastructure that supports the Semantic Web vision, and the foundation upon which we hope to build NoS applications. Each new layer builds on the layers of technology below it. We observe the top layer position of OWL among standardized technologies. OWL provides semantic descriptions of the underlying data. Together, XML, RDF and OWL allow for the implementation of reasoning that can prove whether or not assertions are true or false in almost real-time (decidability). Briefly, the bottom layer is constructed of Internationalized Resource Identifiers (IRI) and Unicode. IRIs are a generalized mechanism for specifying a unique address for an item on the web. The eXtensible Markup Language (XML) is organized into tree hierarchies that provide the fundamental layer for representation and management of data on the Web. Semantic Web applications will gather information from a variety of sources, and in the context of NoS, merge and organize these sources for local and global decision-making. Unfortunately, there is no easy way for tree structures to be merged. The resource description framework (RDF) solves this problem by allowing for the representation of graphs of data on the web. Graphs can always be merged.

Not all technologies on the semantic web are standardized. Some are emergent ones that are used mostly for horizontal and vertical integration of multiple layers of the stack. Generally speaking, they are Application Programming Interfaces (API) used to complete integration tasks. Jena is one of such technologies (see http://www.jena.apache.org). It's an open source Java framework for building semantic web and linked data applications. It provides APIs for RDF, triple store and OWL (ontology and inference). Jena uses a rule-based reasoning approach, which is the classic technique to logic-based reasoning where the knowledge-based system is developed by deduction, induction, abduction or choices from a starting set of data and rules. A unifying logic, such as the DL, is needed for horizontal integration of top layers of stacks and provides the rigorous, formal support needed by system (NoS) models. The latter are the result of the top level, i.e., the application layer, which allows users to visualize and interact with whatever underlying semantic platform supporting the application.

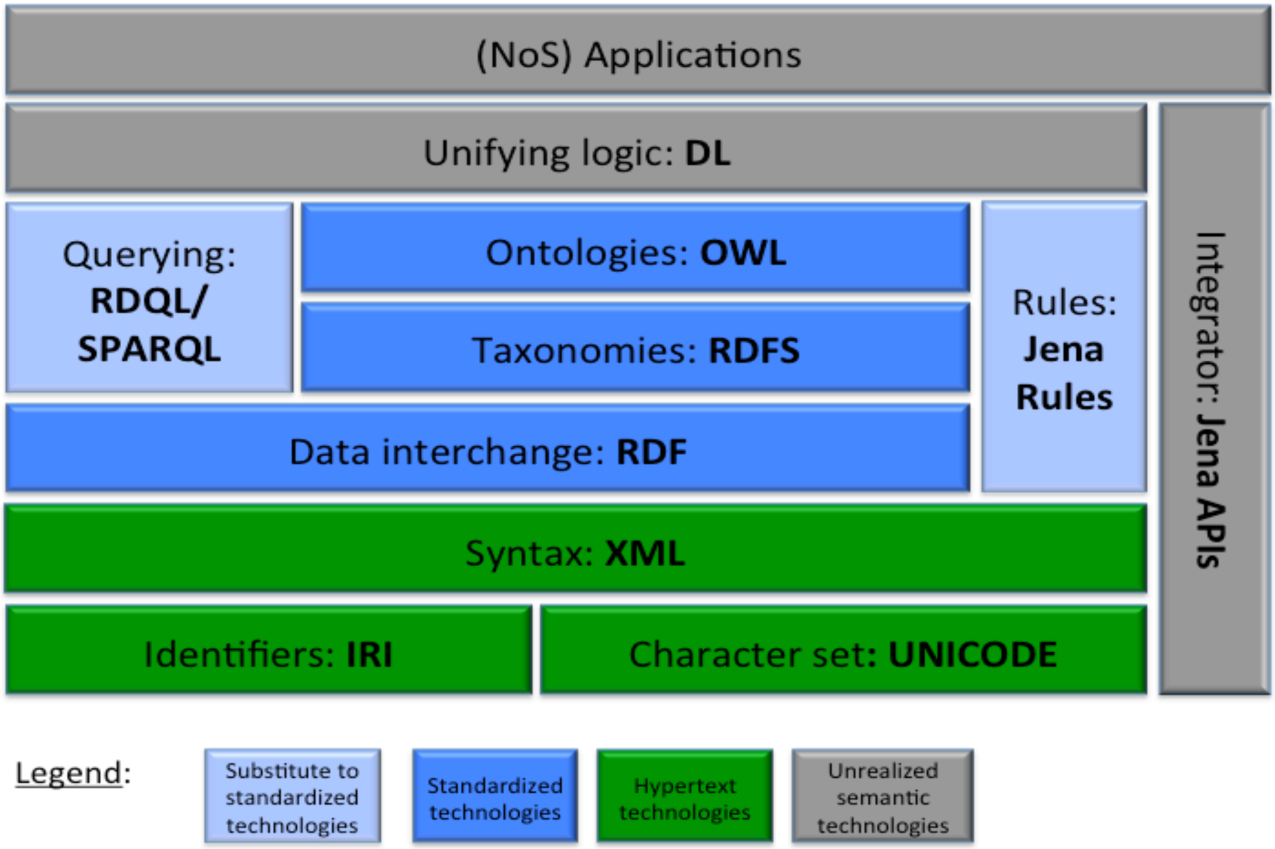


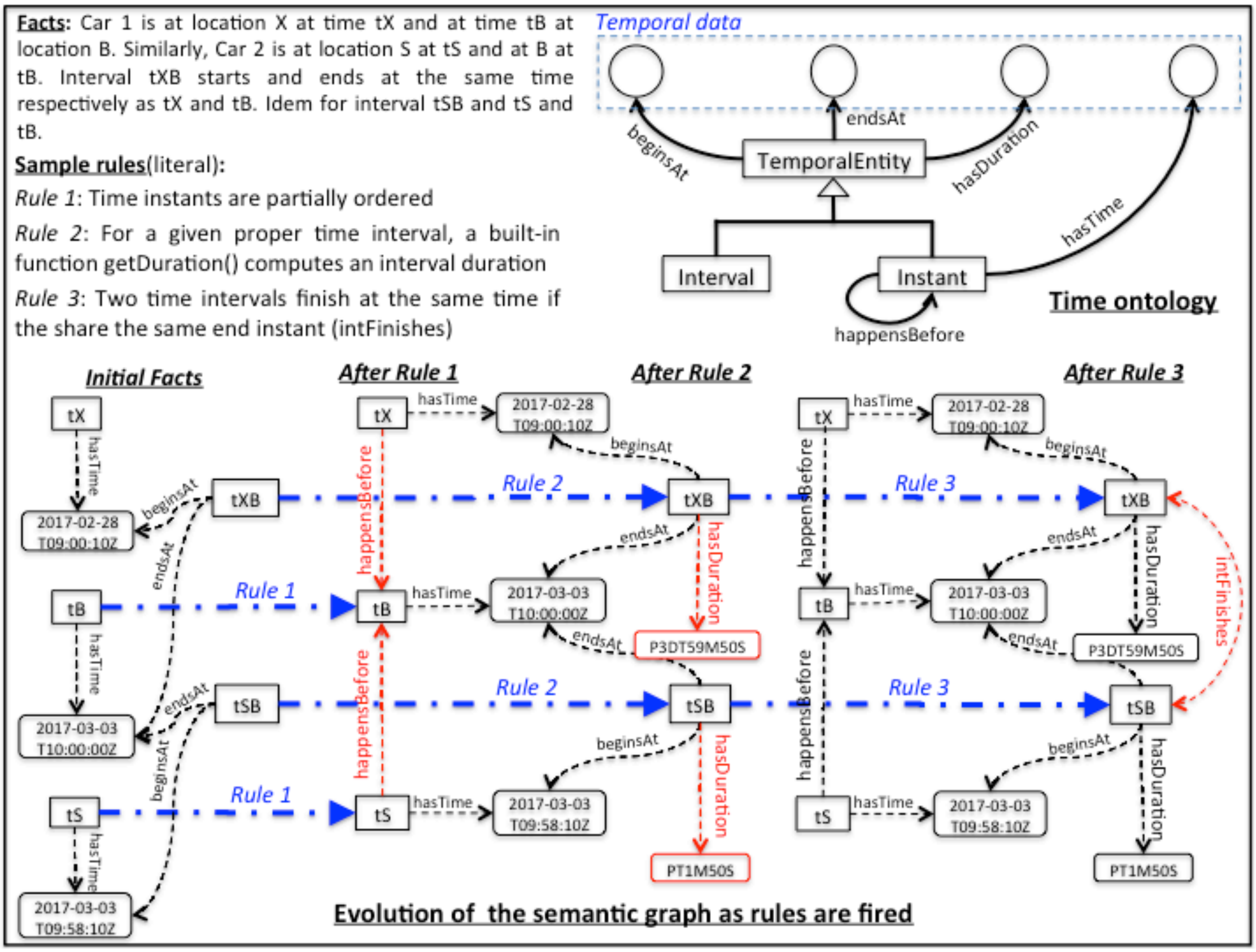
Figure 3: Semantic Web technology stack.

Semantically-enabled Modeling and Reasoning for Network of Systems

Semantic Modeling of Space and Time. Generally speaking, safety-critical NoS will be distributed in both space and time, possibly even covering multiple time zones. This makes the understanding of these domains critical to effective semantic modeling of such systems.

Formal approaches to NoS design need representations of space and time that are mathematically precise. Important modeling challenges include the multiplicity of theories (point, interval, duration, dimension), the complexity of expressing ternary relation involving time in triple graphs, the difficult choice between discrete (efficient but inaccurate) and continuous (accurate but computationally inefficient) time. Fortunately, foundation research found interval-based models as most appropriate for formal analysis having time-dependent behavior. Therefore, Allen's temporal interval calculus (ATIC) was identified as the most qualified temporal theory for modeling of this domain (Allen, 1983). Instant and proper time intervals are primitives in this theory. It enables formal expression of mereological (part-of), topological (connects) and logical (rules-based) relationships between entities. Restricted axioms ensure time reasoning decidability (in OWL DL).

Figure 4 illustrates semantic modeling and reasoning mechanisms in the temporal domain. The upper right-hand side shows a representation of the time ontology with the relationships among classes (e.g.: #Instant) and properties (e.g.: #hasTime). The upper left-hand side shows a set of facts and three rules. Locations X, S and B are on cars trajectories as indicated in Figure 2 (a). We note that Rule #3 is the literal definition of ATIC's "finishes" topological relationship between two proper time intervals. Encoding the facts in the knowledge based initializes the time semantic graph and, when the rules are applied to the ontology, the structure of the graph is transformed through inferencing as introduced in the previous section. The schematic along the bottom of the figure shows the evolution of part of the time semantic graph as the three rules (R1)-(R3) are executed as a function of time.



Facts: Car 1 is at location X at time tX and at time tB at location B. Similarly, Car 2 is at location S at tS and at B at tB. Interval tXB starts and ends at the same time respectively as tX and tB. Idem for interval tSB and tS and tB.

Sample rules(literal):

Rule 1: Time instants are partially ordered

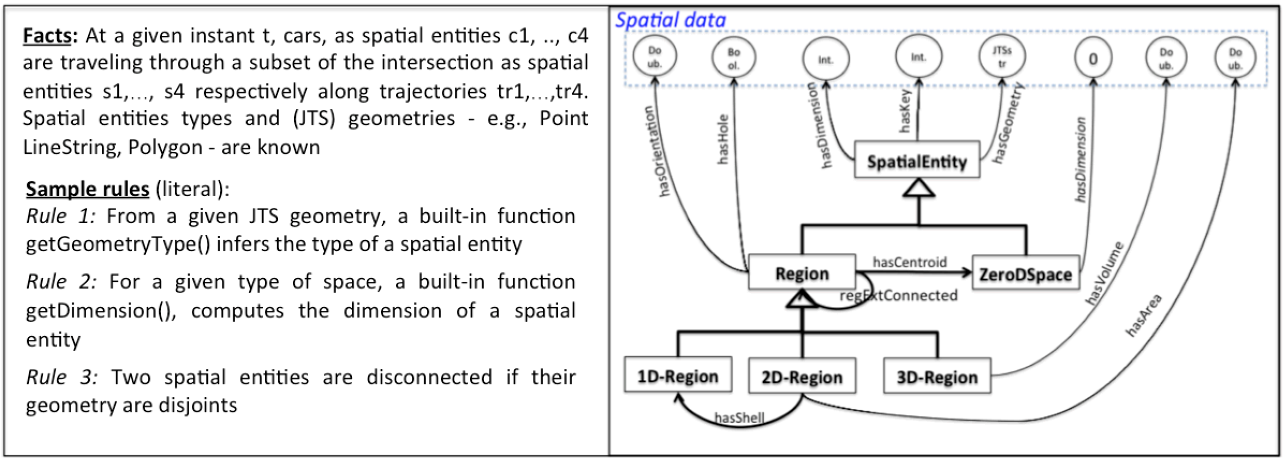
Rule 2: For a given proper time interval, a built-in function getDuration() computes an interval duration

Rule 3: Two time intervals finish at the same time if the share the same end instant (intFinishes)

Figure 4: Illustration of semantic-driven modeling and reasoning in the temporal domain.

As for time, we need ambiguity-free models of space that properly capture and infer the spatial configurations of NoS as materialized in the world. In addition to the decision between a wide range of theories (point, interval, array, region, etc.), the modeler is faced with the challenge of handling non-mereotopological aspects (orientation, shape, elevation, etc.), avoiding triadic relations and dealing with mereotopological composition complexity (multiple possible results). Moreover, space is also often under-specified in natural language and computational accuracy is a very relevant consideration especially when sensors are moving; the velocities of both the sensors and moving objects will affect timeliness of computations. Because there is no one-size-fits-all theory that tackles all these challenges, we select the one that matches our need the best i.e. the Region Connectedness Calculus (RCC-8) (Cohn, 1997). It's a space-region theory with strong mereotopological focus and enough flexibility to seamlessly integrate with "low dimension" theories and extensions to account for key relevant non-topological aspects such as distance, area, volume and other relevant features. In this theory, regions of dimension D ≥ 1 and height (8) mereotopological relations are formally defined between them; there is also a RCC-5 version. In order to achieve reasoning decidability and simplify the process, we add restrictions to the theory with (1) closed convex spatial entities (those with holes can be triangularized), (2) no transitivity deduction between spatial entities, (3) maximum dimension of spatial entities limited to three (to satisfy the hyperbolicity property for space-time interactions). Figure 5 shows an excerpt of the space ontology with a representation of the relationship among classes and properties. A spatial entity has a few core properties (e.g.: #hasGeometry, #hasDimension, etc.) that are inherited by its specializations (e.g.: #Region, #ZeroDSpace). The left-hand side of the figure shows a set of facts (capturing the state of the given system) and three rules. Cars 1-4 are represented as spatial entities s1- s4 in the space ontology along with their properties, traveling on a trajectory tr1- tr4 as shown in Figure 2 (a). The following rule is the Jena translation of the RCC-8 "DisConnect" relationship between two regions.

```
// Rule 3 (R3): Deduction of "regDisConnected" relationship between spaces ...
[ DC: (?x rdf:type se:Region) (?y rdf:type se:Region) noValue (?x se:regDisConnected ?y) (?x
se:hasGeometry ?g1) (?y se:hasGeometry ?g2) getRCCRelation(?g1,?g2,?rel)
equal(?rel,"DC"^^xsd:string) -> (?x se:regDisConnected ?y) ]
```



Facts: At a given instant t, cars, as spatial entities c1, .., c4 are traveling through a subset of the intersection as spatial entities s1,..., s4 respectively along trajectories tr1,...,tr4. Spatial entities types and (JTS) geometries - e.g., Point LineString, Polygon - are known

Sample rules (literal):

Rule 1: From a given JTS geometry, a built-in function getGeometryType() infers the type of a spatial entity

Rule 2: For a given type of space, a built-in function getDimension(), computes the dimension of a spatial entity

Rule 3: Two spatial entities are disconnected if their geometry are disjoints

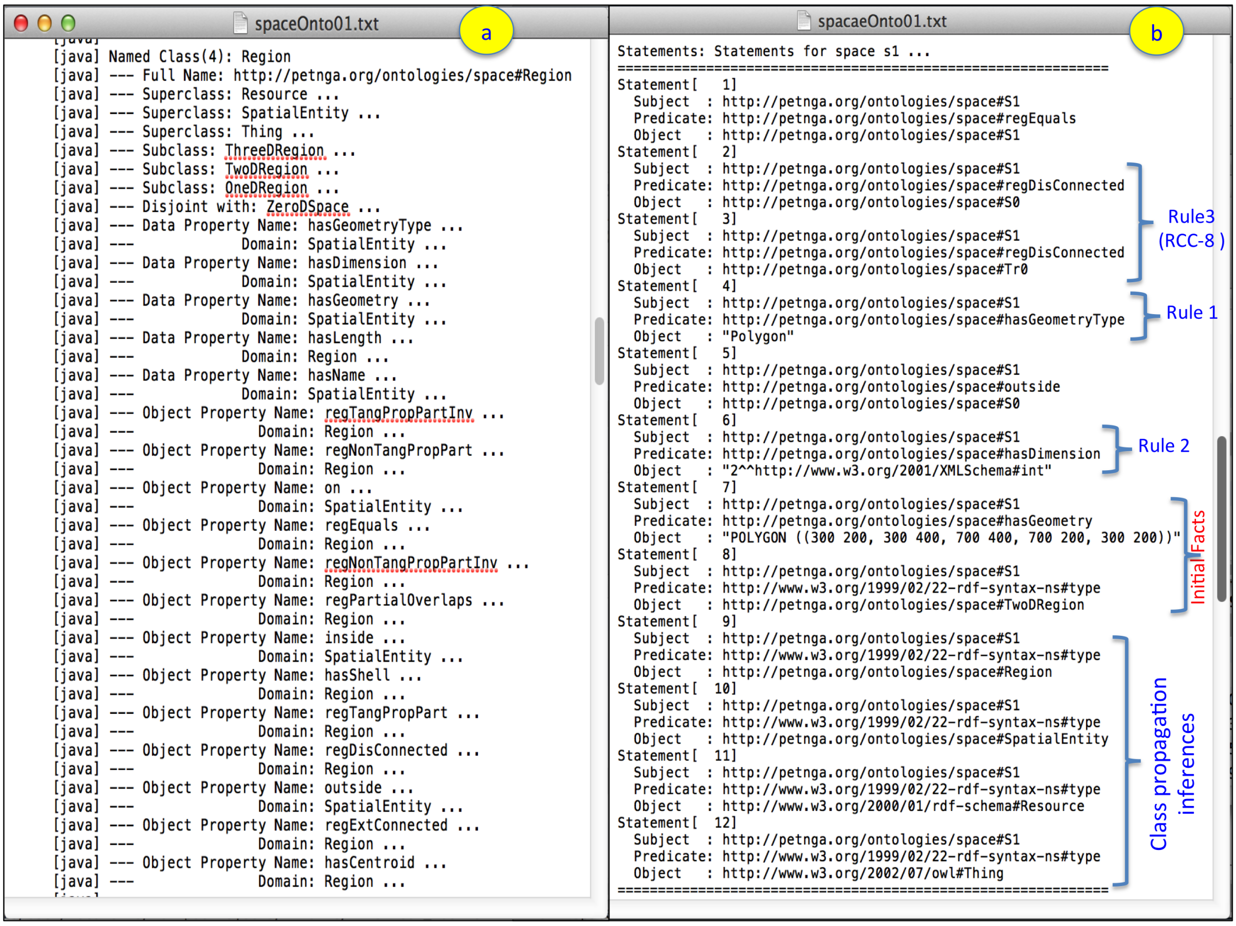
Figure 5: Illustration of a simplified ontology of space and sample literal rules.

Figure 6 (a) shows a view of the full description of a region while (b) is an excerpt of statements relative to a given spatial entity in the knowledge base. We note that the geometry of the entity (Statement 7) is encoded as a Java Topology Suite (JTS) object. JTS is an object-oriented software library providing Euclidian planar linear geometry algorithms in computational geometry (see http://tsusiatsoftware.net/jts/main.html). Statements 4, 6 and 2-3 show the respective results of the Rules 1, 2 and 3 on the set of initial facts (Statements 7-8).

Because notions of space and time cut across all domains (they are not specific to NoS), by themselves, the associated models are not sufficient to fully tackle the modeling challenges of this class of systems. We close this gap by introducing a multi-layer system architecture that connects cross-cutting theories and models to engineering models and semantic models.

System Architecture. The development of semantic architectures for NoS is complicated by the variability and multiplicity of participating domains, the conjunction of two worlds (cyber & physical) and the need to understand and formally evaluate system behaviors resulting from interactions between components and with the surrounding environment. We introduce the three-layer architecture in Figure 7 to tackle these challenges. Each bottom layer provides the one above with the necessary foundation (semantic) needed as follows.

Theories and Standards Layer. It provides the mathematical foundations needed to support the formal description of meta-domain knowledge. Meta theories (for known cross-cutting domains such as time, space, communication, etc.) are separated from plain ones, which could be well-accepted domain standards. Unlike the former, the latter might not always be available for the problem at hand. Both RCC-8 and ATIC introduced above as respectively spatial and temporal calculus supporting semantic modeling of those domain belong to this layer of the architecture.



```
[java] Named Class(4): Region
[java] --- Full Name: http://petnga.org/ontologies/space#Region
[java] --- Superclass: Resource ...
[java] --- Superclass: SpatialEntity ...
[java] --- Superclass: Thing ...
[java] --- Subclass: ThreeDRegion ...
[java] --- Subclass: TwoDRegion ...
[java] --- Subclass: OneDRegion ...
[java] --- Disjoint with: ZeroDSpace ...
[java] --- Data Property Name: hasGeometryType ...
[java] ---                Domain: SpatialEntity ...
[java] --- Data Property Name: hasDimension ...
[java] ---                Domain: SpatialEntity ...
[java] --- Data Property Name: hasGeometry ...
[java] ---                Domain: SpatialEntity ...
[java] --- Data Property Name: hasLength ...
[java] ---                Domain: Region ...
[java] --- Data Property Name: hasName ...
[java] ---                Domain: SpatialEntity ...
[java] --- Object Property Name: regTangPropPartInv ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: regNonTangPropPart ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: on ...
[java] ---                  Domain: SpatialEntity ...
[java] --- Object Property Name: regEquals ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: regNonTangPropPartInv ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: regPartialOverlaps ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: inside ...
[java] ---                  Domain: SpatialEntity ...
[java] --- Object Property Name: hasShell ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: regTangPropPart ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: regDisConnected ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: outside ...
[java] ---                  Domain: SpatialEntity ...
[java] --- Object Property Name: regExtConnected ...
[java] ---                  Domain: Region ...
[java] --- Object Property Name: hasCentroid ...
[java] ---                  Domain: Region ...
```

```
Statements: Statements for space s1 ...
==============================================================
Statement[   1]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#regEquals
  Object   : http://petnga.org/ontologies/space#S1
Statement[   2]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#regDisConnected
  Object   : http://petnga.org/ontologies/space#S0
Statement[   3]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#regDisConnected
  Object   : http://petnga.org/ontologies/space#Tr0
Statement[   4]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#hasGeometryType
  Object   : "Polygon"
Statement[   5]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#outside
  Object   : http://petnga.org/ontologies/space#S0
Statement[   6]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#hasDimension
  Object   : "2^^http://www.w3.org/2001/XMLSchema#int"
Statement[   7]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://petnga.org/ontologies/space#hasGeometry
  Object   : "POLYGON ((300 200, 300 400, 700 400, 700 200, 300 200))"
Statement[   8]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  Object   : http://petnga.org/ontologies/space#TwoDRegion
Statement[   9]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  Object   : http://petnga.org/ontologies/space#Region
Statement[  10]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  Object   : http://petnga.org/ontologies/space#SpatialEntity
Statement[  11]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  Object   : http://www.w3.org/2000/01/rdf-schema#Resource
Statement[  12]
  Subject  : http://petnga.org/ontologies/space#S1
  Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  Object   : http://www.w3.org/2002/07/owl#Thing
==============================================================
```

Figure 6: A view of a) the ontological class Region and b) statements relative to a space s1 in the knowledge base.

Cross-domain Knowledge Layer. It relies on the semantic foundations provided by the theory layer to formally describe meta-domain and generic domain knowledge. Knowledge is encoded in the form of a "semantic block" made of (1) an ontology, (2) set of rules and (3) interfaces (not shown) that enable communication between semantic blocks and linking with computation platforms via customized builtin functions. Practically, generic domain knowledge can be organizational or operational policies, standard operating procedure/process (SOP) or directives that are applicable to the system at hand in part or as a whole. In the context of this framework, they should be encoded as semantic blocks. They might refer to/or use some meta-domain in their description (especially in the absence of supportive theories). Because of the significant difference in the role they play in the framework, meta-domain and generic domain knowledge are processed differently in the layer above. Semantic models of time and space introduced above are integral part of this layer.

Application Models Layer. This is the top and most complex layer of the architecture. We distinguish semantic from engineering models representing respectively the cyber and the physical worlds. The problem domain is made of textual requirements that are processed and encoded as instance of a well-defined requirement ontology through mechanism and rules that ensure their syntactic and semantic correctness. In order to achieve their proper elicitation, requirements make use of meta-domain knowledge, which provides entities as range to properties defined in the requirement ontology. The solution domain is made of engineering models and associated (component and control) semantic models. In line with the operation of NoSs, the latter are responsible of system/component functionality and decision making while the former are in charge of sensing and actuation. Models of system structure and behavior make up the engineering models. They map (partially) to component semantic models. In order to keep the complexity of the reasoning in check, only the information required for decision-making flows. Semantic blocks are used to specify individual NoS component types and instances, mirroring entities in engineering
]

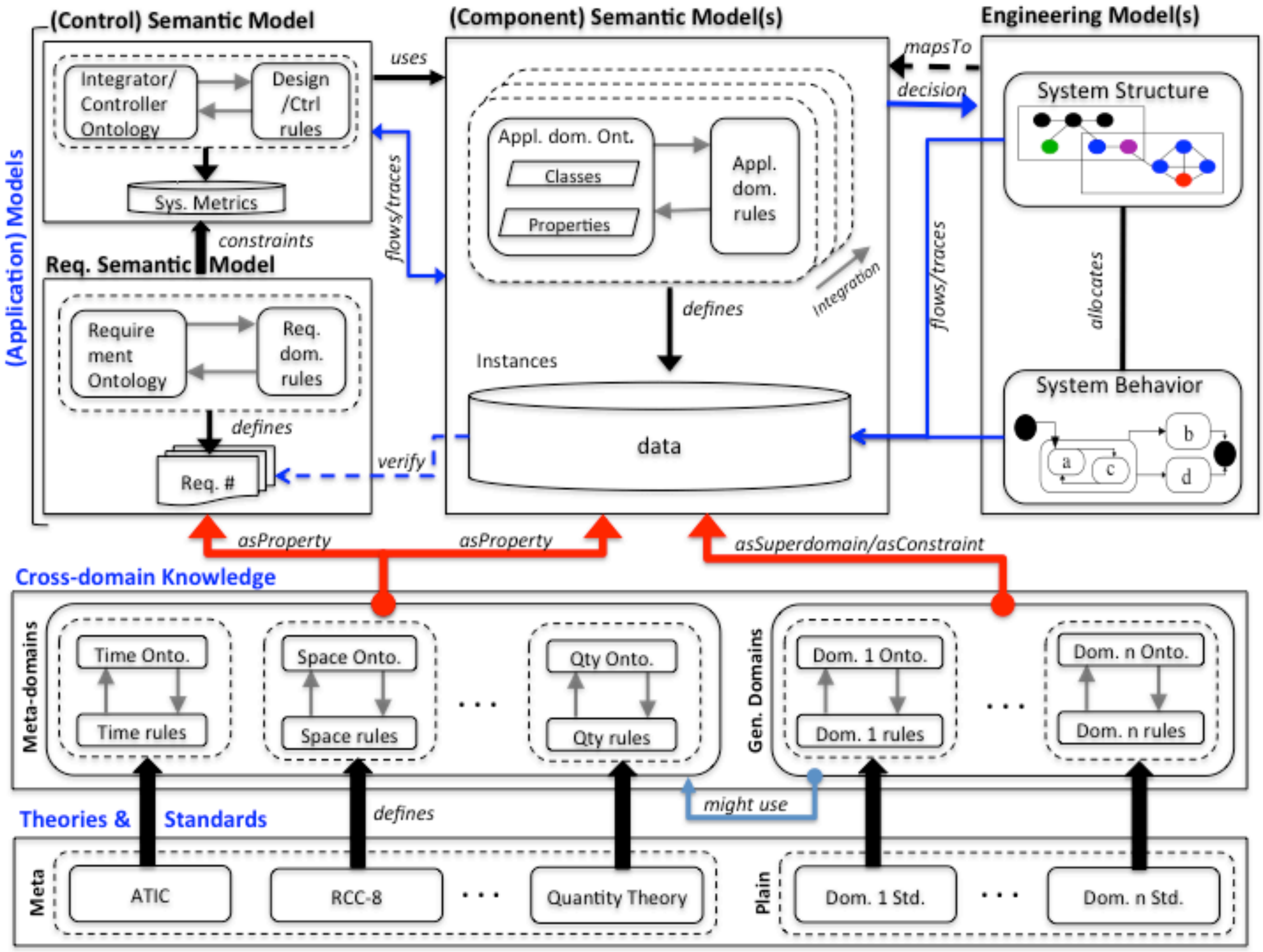


Figure 7. Framework for semantically-enabled model-based systems engineering of safety-critical network of systems

models whose data are stored in the knowledge repository. Meta-domains are added to individual components as property. The control of the system requires the integration of component knowledge and the creation of a control semantic model that comprises an integrator that can act as a semantic controller of the system. It's responsible of defining system metrics (instances) and provides guarantees of the satisfaction of system requirements (as constraints) as well as the integration of generic domain knowledge in decision making at all levels.

Development Process. The construction of the framework in Figure 7 requires a systematic and rigorous process involving ontological and knowledge modeling expertise. Figure 8 shows the proposed development process from the prospective of the system engineer. The various semantic models are assumed to exist as retrievable, modifiable and reusable components stored in a library of semantic blocks. Starting with an initial set of requirements, the first step involves analysis of the NoS design problem at hand, and uncovering the various application and cross-domain knowledge involved. Also, system level and metrics will be developed/identified at this stage. Then, system goals and scenarios are specified as done in traditional MBSE approaches. Using the library of semantic block components, textual requirements are semantically modeled, so are the system components (including its controller). Engineering models of the latter can be developed concurrently but their key attributes are to be mapped to the semantic model as per the system goals and scenarios. This will enable communication between components and further ease system integration. Rules will be used to evaluate compatibility between components. The later will be used in a system configuration only if they are shown to satisfy the required (and expected functionality) and are compatible (interface) with system components to which they will be interacting. Next, the system is simulated. Its performance will be evaluated against the requirements (model) and the design is modified accordingly. Changes (e.g.: new requirement) can be handled as new valid statement(s), property(ies), rule(s), physical component(s) added/removed to the corresponding block(s) of the appropriate semantic and engineering model(s).

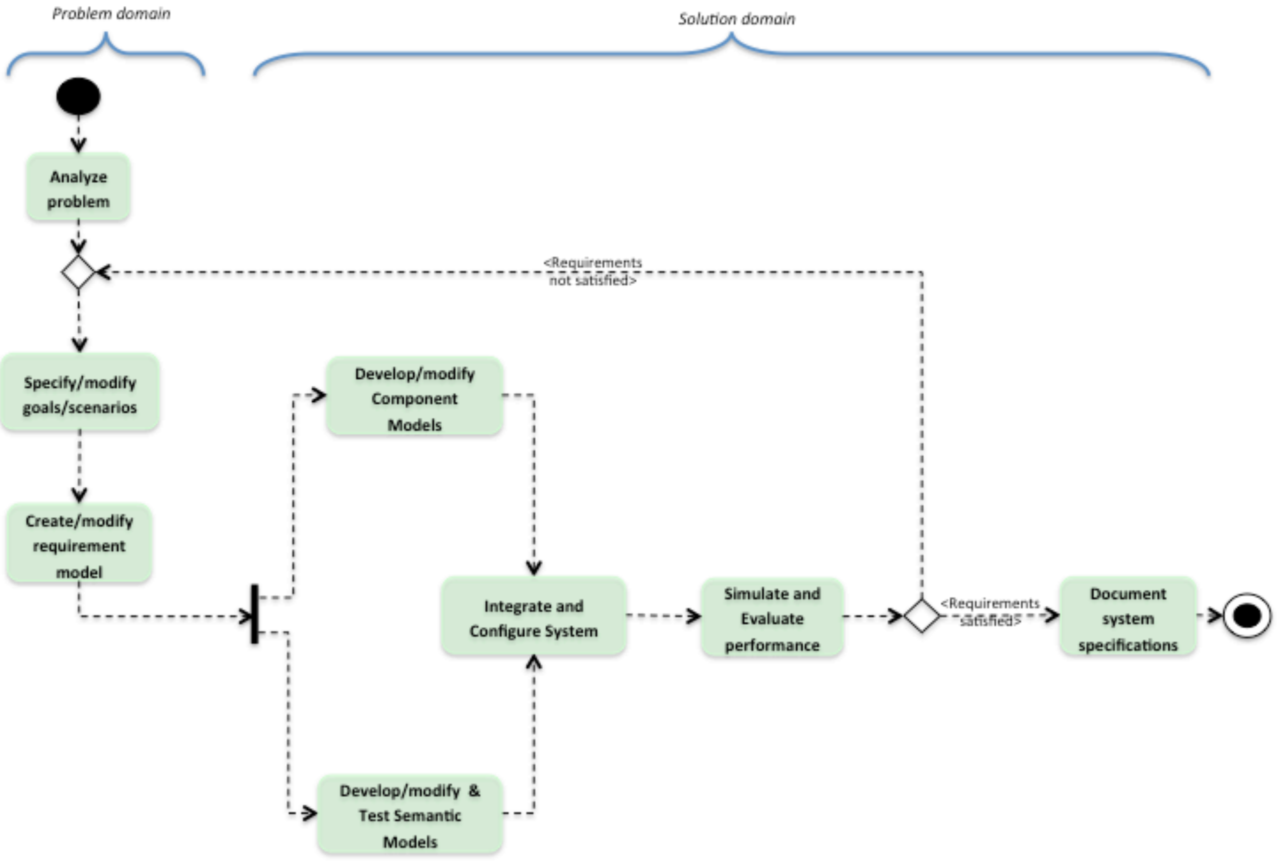

Problem domain
Solution domain
Analyze problem
<Requirements not satisfied>
Specify/modify goals/scenarios
Create/modify requirement model
Develop/modify Component Models
Integrate and Configure System
Simulate and Evaluate performance
<Requirements satisfied>
Document system specifications
Develop/modify & Test Semantic Models

Figure 8. Development process for semantically-enabled NoS.

Models Integration and Simulation with Whistle. The execution of the process described in Figure 8 requires novel software infrastructures that are able to seemingly integrate various semantic and engineering models. To that aim, we turn to Whistle, a Java-based scripting language (still under development) designed for the integration and simulation of applications that are glued together (Delgoshaei, 2014). Among the key features of the language are its ability to: (1) support the use of physical units and dimensions, multi-format (XML, Open Street Map (OSM), Java, etc.) input/output, (2) enable the use of variables, matrices, looping and branching structures to control the flow of program logic and, (3) support the integration of custom-built functions (along with their names and arguments). Together, these capabilities enable the simulation and analysis of continuous & discrete behaviors. For instance, the short fragment of Whistle code:

```
area = 0.04 m^2; // Cross section area of a pipe ...
velocity = 5 m/sec; // Fluid velocity ....
Q = area*velocity; // Discharge rate ...

print "*** Cross section area = ", area;
print "*** Fluid velocity = ", velocity;
print "*** Discharge rate = ", area*velocity;
```

shows, for example, the computation of the flow-rate computation through a pipe. Whistle makes extensive use of model-view-controller (MVC) software design pattern along with listening mechanisms to clearly separate domains from object presentations, and enables communication among multiple models and views, with the controller as mediator. Another design pattern used is the composite hierarchy, a multi-layer tree structure of arbitrary complexity, implemented in a flexible and scalable manner. It enables the systematic assembly and display of objects. This has been shown useful in creating and visualizing spatial features of engineering models with various levels of complexity.

Prototype Implementation: Collision Avoidance at Lightless Intersection

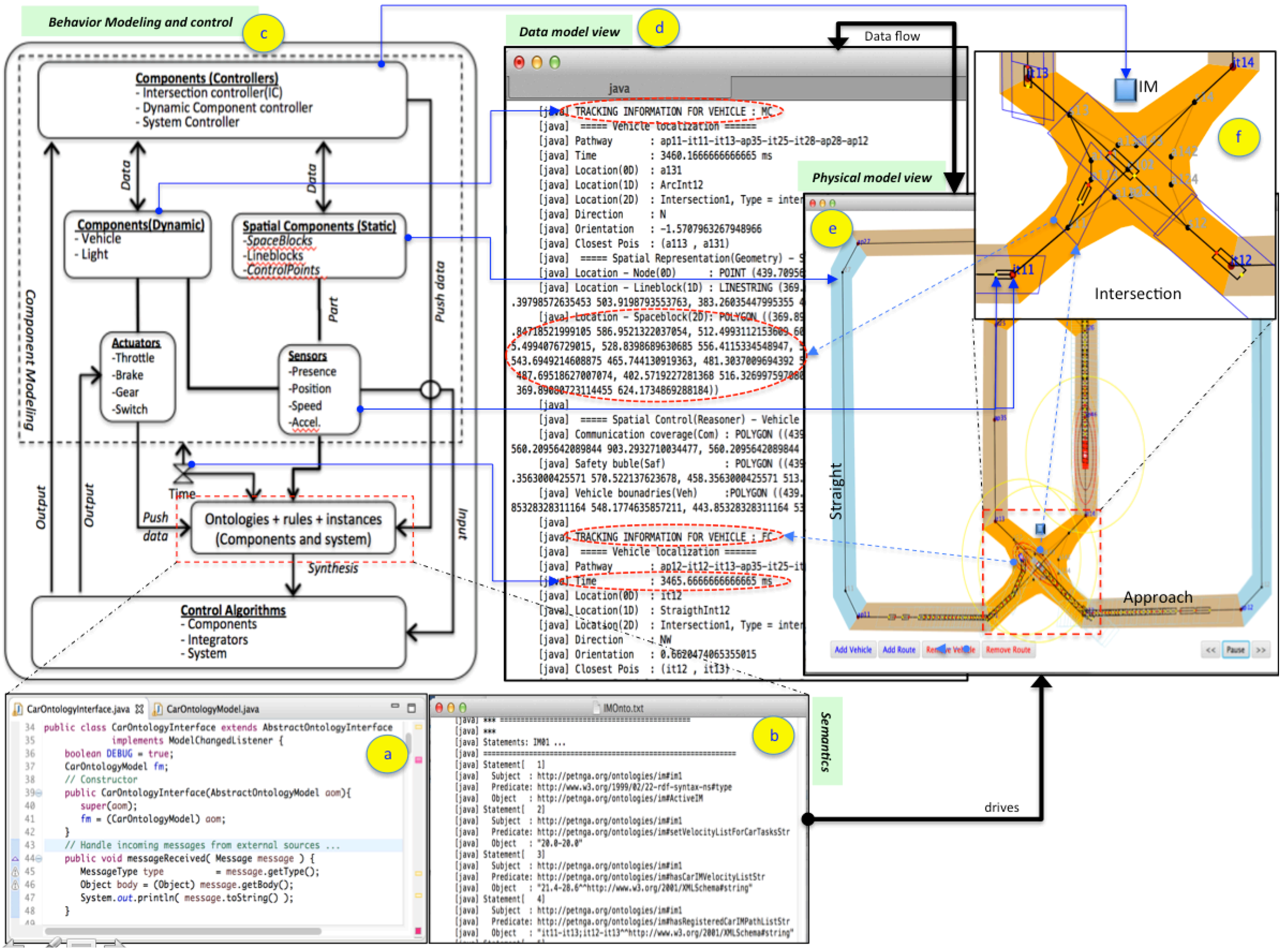
Problem Description and Assumptions. To illustrate the use and effectiveness of the semantically-enabled MBSE framework, we consider the problem of glancing collision between two smart cars at a non-signalized traffic intersection. This scenario is similar to the configuration where car 1 and 2 would collide at location B as pictured in Figure 2 (a). Stoplights and signs at the intersection are replaced by a single intersection manager(IM) that monitors traffics and intervenes to resolve spatio-temporal conflicts impeding safe crossings. The core requirement of the system is that of *"no pair of vehicle should occupy the same space of the intersection at the same time"*(Req. 1). This system is an "ideal" Directed NoS (as modeled in Figure 1) in the sense that each vehicle is an autonomous system equipped with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) capabilities. The IM is the entity responsible of the system goal of maximizing traffic throughput while ensuring safe crossing of the intersection. Thus, it provides directives to vehicles to that aim. We seek to develop a simplified implementation of the framework in Figure 7 to model and analyze this system. Our purpose here is to show how the structure and configuration of the semantic framework, the cooperation between the actors and use of cross-domain knowledge (especially space and time) in reasoning result in safe operation of this simple NoS.

For the purpose of this experiment, we assume that: (i) vehicles travel at constant but different speeds on two intersecting, known trajectories, along with the locations of checkpoints and (ii) always execute the directives of the IM as given in no delay. Also, (iii) computation and communication are performed within actuation and sensing response time margins of error with no delay. Finally, (iv) they can communicate with each other when they are within sensing range as pictured on the racetrack in Figure 9 (e) and (f) but they don't have decision capabilities to predict and resolve spatio-temporal conflicts (collisions).

Each approaching vehicle, when within communication range of the IM, must identify itself and submit its velocity and estimated arrival time at the closest checkpoint of the intersection. The IM processes the information received from all vehicles, executes appropriate predefined collision avoidance algorithms (details out of the scope of this work) and returns any required change of dynamic (e.g.: velocity) to each vehicle for execution. The series of schematics in Figure 9 show a prototype (under development) of the system implementing the proposed semantic framework for this problem.

Simplified engineering models. In order to keep the experiment simple yet explicit enough to maintain the focus on the topic of this research, vehicles are represented as "moving/dynamic" (red, gray and blue) rectangles surrounded by their respective safety bubble (a set of three red ellipses) and communication range/coverage (yellow ellipse) as shown in schematic Figure 9(e). Their semantic representations (JTS geometries) are used in the collision avoidance algorithm implemented by the IM. The latter is represented by a blue small rectangle at the top corner of the intersection but its coverage is omitted. Red and yellow dots represent sensors indicating the front and back of vehicles (Figure 9(f)). The racetrack itself is considered a "static" component partitioned into straight, approach and intersection areas with checkpoints (in blue) at the boundaries. This partition obeys a well-defined spatial modeling framework we've introduced in (Petnga, 2015). The behavior of each individual component of this NoS is illustrated by the simplified state chart next to the given element in Figure 9(c). Details of the models can be observed in real-time in the data model view (Figure 9(d)).

Semantic Modeling and Reasoning. Semantic models of the car and IM are created as standalone semantic blocks, with component ontologies extended by individual reasoner implementing corresponding rules. For more complex requirements or scenarios, requirements will be modeled as a separate domain i.e. a standalone semantic block. Ontologies communicate via their interfaces (as encoded in Figure 9(a)), which listen to (and communicate relevant) changes in the semantic graph across the domains and from external sources. These changes ultimately propagate into the knowledge base of the entity such as the one of the IM partially shown in Figure 9(b).



```
[java] TRACKING INFORMATION FOR VEHICLE : MC
[java]  ===== Vehicle localization ======
[java] Pathway       : ap11-it11-it13-ap35-it25-it28-ap28-ap12
[java] Time          : 3460.1666666666665 ms
[java] Location(0D)  : a131
[java] Location(1D)  : ArcInt12
[java] Location(2D)  : Intersection1, Type = inter
[java] Direction     : N
[java] Orientation   : -1.5707963267948966
[java] Closest Pois  : (a113 , a131)
[java]  ===== Spatial Representation(Geometry) - S
[java] Location - Node(0D)       : POINT (439.70956
[java] Location - Lineblock(1D) : LINESTRING (369.
.39798572635453 503.9198793553763, 383.26035447995355 4
[java] Location - Spaceblock(2D): POLYGON ((369.89
.84718521999105 586.9521322037054, 512.499311215360 60
5.4994076729015, 528.8398689630685 556.4115334548947, 5
543.6949214608875 465.744130919363, 481.3037009694392 5
 487.69518627007074, 402.5719227281368 516.32699759708
 369.89080723114455 624.1734869288184))
[java]
[java]  ===== Spatial Control(Reasoner) - Vehicle
[java] Communication coverage(Com) : POLYGON ((439
560.2095642089844 903.2932710034477, 560.2095642089844
[java] Safety buble(Saf)           : POLYGON ((439
.3563000425571 570.522137623678, 458.3563000425571 513.
[java] Vehicle bounadries(Veh)     :POLYGON ((439.
85328328311164 548.1774635857211, 443.85328328311164 53
[java]
[java] TRACKING INFORMATION FOR VEHICLE : FC
[java]  ===== Vehicle localization ======
[java] Pathway       : ap12-it12-it13-ap35-it25-it
[java] Time          : 3465.6666666666665 ms
[java] Location(0D)  : it12
[java] Location(1D)  : StraigthInt12
[java] Location(2D)  : Intersection1, Type = inter
[java] Direction     : NW
[java] Orientation   : 0.6620474065355015
[java] Closest Pois  : (it12 , it13)
```

```java
public class CarOntologyInterface extends AbstractOntologyInterface
            implements ModelChangedListener {
    boolean DEBUG = true;
    CarOntologyModel fm;
    // Constructor
    public CarOntologyInterface(AbstractOntologyModel aom){
        super(aom);
        fm = (CarOntologyModel) aom;
    }
    // Handle incoming messages from external sources ...
    public void messageReceived( Message message ) {
        MessageType type          = message.getType();
        Object body = (Object) message.getBody();
        System.out.println( message.toString() );
    }
```

```
[java] *** ==========================================
[java] ***
[java] Statements: IM01 ...
[java] ==========================================================
[java] Statement[   1]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
[java]   Object   : http://petnga.org/ontologies/im#ActiveIM
[java] Statement[   2]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#setVelocityListForCarTasksStr
[java]   Object   : "20.0-20.0"
[java] Statement[   3]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasCarIMVelocityListStr
[java]   Object   : "21.4-28.6^^http://www.w3.org/2001/XMLSchema#string"
[java] Statement[   4]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasRegisteredCarIMPathListStr
[java]   Object   : "it11-it13;it12-it13^^http://www.w3.org/2001/XMLSchema#string"
```

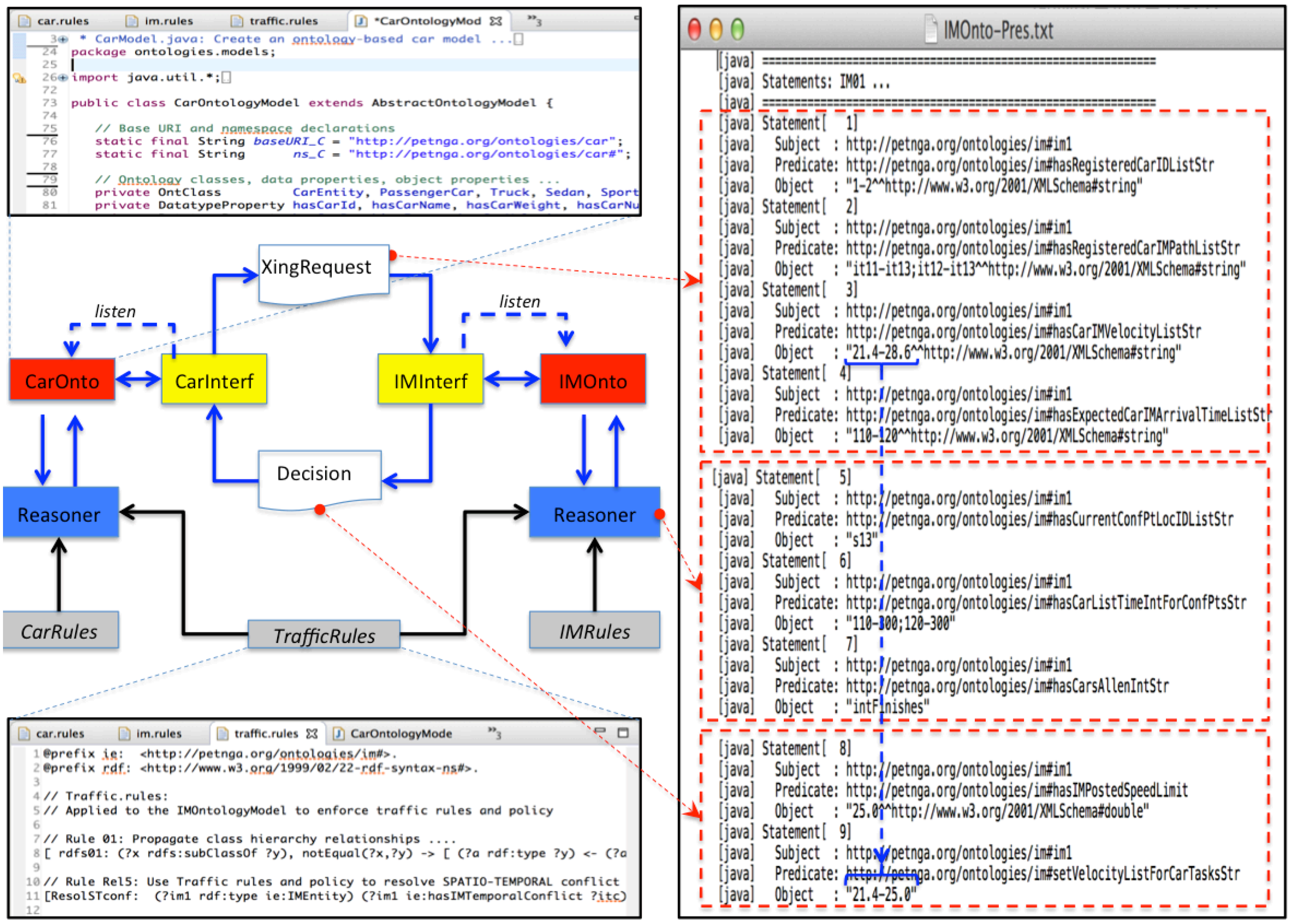
Figure 9. Schematic of the semantically-enabled implementation for a lightless traffic system.

As indicated above for time and space, decidability of reasoning tasks (for collision avoidance) involving such cross-cutting domains requires precise and accurate models. However, this can add complexity to the reasoning process and increase computation time, leading to invalid decisions and low performance of the system as shown in (Petnga, 2015). Thus, for illustration and simplification purposes, we choose the lowest possible ontological commitment i.e. 0D space + Time for our semantic models. Thus, temporal relations are computed and established as per ATIC.

Figure 10 illustrates the communication and control mechanisms in the networked traffic system for collision avoidance. Traffic policy (for instance, right of way to the vehicle coming to the right) are handled as generic domain rules applied to the entities involved (i.e.: cars) by the IM. The latter is the controller of the system but no additional design rules nor metrics are added or defined.

Vehicles approaching the intersection send a “crossing request” containing the required information (Statements 1 - 4) to the IM and it returns its decision on conflict resolution (Statements 8 - 9). The right hand side of the Figure illustrates what is happening behind the scene as the collision is predicted and resolved before it materializes in the physical world.

Executing the reasoning algorithm (which translates Req.1 as constraint), the IM computes and finds *s13* (Figure 9(f)) as the spatial intersection of the two trajectories thus, the risk of glancing collision. It needs to know each vehicle registration (Statement 1), trajectory (Statement 2), current velocity (Statement 3) and expected arrival time at the closest intersection checkpoint (Statement 4). It computes the travel duration of each vehicle to *s13* and infers the corresponding time interval of each vehicle while in the intersection control zone (Statement 6). Next, the nature of the relationship between the intervals as per ATIC is established as "intFinishes" (Statement 7). This clearly predicts a temporal conflict between the two vehicles, which completes the prediction of glancing collision at *s13*.



```
 * CarModel.java: Create an ontology-based car model ...
package ontologies.models;

import java.util.*;

public class CarOntologyModel extends AbstractOntologyModel {

    // Base URI and namespace declarations
    static final String baseURI_C = "http://petnga.org/ontologies/car";
    static final String      ns_C = "http://petnga.org/ontologies/car#";

    // Ontology classes, data properties, object properties ...
    private OntClass            CarEntity, PassengerCar, Truck, Sedan, Sport
    private DatatypeProperty hasCarId, hasCarName, hasCarWeight, hasCarNu
```

```
@prefix ie:  <http://petnga.org/ontologies/im#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Traffic.rules:
// Applied to the IMOntologyModel to enforce traffic rules and policy

// Rule 01: Propagate class hierarchy relationships ....
[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y) -> [ (?a rdf:type ?y) <- (?a

// Rule Rel5: Use Traffic rules and policy to resolve SPATIO-TEMPORAL conflict
[ResolSTconf:  (?im1 rdf:type ie:IMEntity) (?im1 ie:hasIMTemporalConflict ?itc)
```

```
[java] ============================================================
[java] Statements: IM01 ...
[java] ============================================================
[java] Statement[   1]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasRegisteredCarIDListStr
[java]   Object   : "1-2^^http://www.w3.org/2001/XMLSchema#string"
[java] Statement[   2]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasRegisteredCarIMPathListStr
[java]   Object   : "it11-it13;it12-it13^^http://www.w3.org/2001/XMLSchema#string"
[java] Statement[   3]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasCarIMVelocityListStr
[java]   Object   : "21.4-28.6^^http://www.w3.org/2001/XMLSchema#string"
[java] Statement[  4]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasExpectedCarIMArrivalTimeListStr
[java]   Object   : "110-120^^http://www.w3.org/2001/XMLSchema#string"
[java] Statement[   5]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasCurrentConfPtLocIDListStr
[java]   Object   : "s13"
[java] Statement[  6]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasCarListTimeIntForConfPtsStr
[java]   Object   : "110-300;120-300"
[java] Statement[   7]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasCarsAllenIntStr
[java]   Object   : "intFinishes"
[java] Statement[  8]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#hasIMPostedSpeedLimit
[java]   Object   : "25.0^^http://www.w3.org/2001/XMLSchema#double"
[java] Statement[  9]
[java]   Subject  : http://petnga.org/ontologies/im#im1
[java]   Predicate: http://petnga.org/ontologies/im#setVelocityListForCarTasksStr
[java]   Object   : "21.4-25.0"
```

Figure 10. Illustration of ontology-based communication and control for spatio-temporal reasoning in networked traffic system.

Finally, the collision (i.e. spatio-temporal conflict) is resolved (in the cyber world) and prevented (in the physical world). The IM does that by identifying the vehicle with the lowest traffic priority (i.e. #2) and computing the velocity required to achieve the temporal separation at *s13* in accordance with traffic rules such as the posted speed limit for that branch of the trajectory (Statement 8). The IM communicates to both vehicles the new (safe) velocities in its decision (Statement 9).

Engineering and Semantic Models Integration and Simulation. In the absence of sound integration science for NoS, software capability is an interesting alternative. Thus, we ought to use Whistle to integrate the various semantic and engineering models of our traffic system within and across domains. This is an ongoing step-by-step effort with careful bottom up assembly, simulation and testing of models. At this time, we are able to integrate physical models to Whistle. Engineering models (car, IM, racetrack) can be created in accordance with their precise JTS geometric representations. The latter are used for basic reasoning and control of the execution of the simulation as per Req. 1 (network of 2 cars and 1 IM). However, the control algorithm has to be very detailed and executed line by line without any flexibility and, in the absence of semantics, it is not scalable to a large number of entities. The integration of the semantic side is on its way with modeling of key components (car, IM) and cross-domain knowledge (space and time) completed. Collision avoidance algorithms making use of cross-domain knowledge and integrating Req. 1 as constraint are already developed and tests are on the way. Linking of cross-domain knowledge and component models as per Figure 7 will follow. Next, Whistle grammar will be updated to enable the integration and testing first of semantic domains then, with engineering models.

Discussion and Conclusions

Benefits of our Approach. The framework introduced in this paper addresses key semantic limitations of state-of-the-art MBSE approaches for NoS. It enables: (1) the development and integration of domain and cross-domain knowledge to engineering models in support of modeling and analysis of emergent behaviors of such systems, (2) mechanisms for (software) integration of cyber and physical components for emulation of smartness and decision making and, (3) support for effective analysis of system-level properties such as safety, that can be formulated in requirements and algorithms as constraints and characterized by metrics. Models are backed by sound DL semantics for which appropriate extensions and restrictions on meta-theories, guarantee stability and decidability of reasoning tasks. As shown in the case study, we go beyond state-of-the-art reservation-based approach (Wuthishuwong, 2013) and account for the geometry of vehicles and their dynamics in the reasoning process. This results in great insights into system design and operation at early-stage of NoS development.

Future Work. There is a need to investigate mechanisms for knowledge modeling of requirements and the systematic translation of textual requirements (in plain English) into syntactically and semantically valid ontological instances. The linkage of requirements (as knowledge domain) with meta-domain knowledge and its integration to the rest of the semantic framework will enable formal and automatic verification of design and smooth the design space exploration process. Also, results obtained in the case study are contingent to a certain number of assumptions and constraints that need to be removed/relaxed to emulate real-world scenarios. For instance, mechanisms for enabling the study and integration of component failure or adversary behavior (ii), account for computation, communication and execution delays and failures (iii) and various environmental changes/affects as well as attacks (both cyber and physical) are critical.

References

Allen, J., 1983, ‘Maintaining Knowledge about Temporal Intervals’, Communications of the ACM, 26(11) : 832–843 (New York, NY, US).

Baader et al., 2008, ‘Description Logics’. Van Harmelen, F., Lifschitz, V., and Porter, B., editors, Handbook of Knowledge Representation, chapter 3, pages 135–180, Elsevier (US).

Berners-Lee, T., Hendler, J., and Lassa, O., 2001, ‘The Semantic Web’, Scientific American, Volume 284, Issue 5, pages 35–43 (US).

Cohn, A., Bennett, B., Gooday, J., and Gotts, N., 1997, ‘Qualitative spatial representation and reasoning with the region connection calculus’, Geoinformatica, 1:144, Springer (US).

Crossley, A., 2004, ‘Systems of Systems: An Introduction of Purdue University Schools of Engineering’s Signature Area’, MIT Engineering Systems Symposium (Cambridge, MA, US).

Delgoshaei, P., Austin, M.A., and Pertzborn, A, 2014, ‘A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems’, International Journal On Advances in Systems and Measurements, Vol. 7, No. 3-4, pp. 223–238 (EU).

Department of Defense (DoD), 2004, ‘Defense Acquisition Guidebook Chapter 4 : System of Systems Engineering,’ US Department of Defense (Washington, DC, US).

Federal Aviation Administration (FAA), 2016, ‘NextGen is Working’, https://www.faa.gov/nextgen/, accessed June 15 (US).

Graves, H., and Bijan, Y., 2011, ‘Using formal methods with SysML in aerospace design and engineering’, Annals of Mathematics and Artificial Intelligence, Springer (US).

Kalawsky, S., 2013, ‘The Next Generation of Grand Challenges for Systems Engineering Research’, Conference on Systems Engineering Research, Georgia Institute of Technology (Atlanta, GA, US).

Horrocks et al., 2006, ‘The Even More irresistible SROIQ’, 10th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, pp. 57–67 (UK).

Horrocks, I., and Pan, Z., 2007, ‘RDFS(FA): Connecting RDF(S) and OWL DL’, IEE Transactions on Knowledge and Data Engineering, Issue 02, volume 19, pp. 192–206 (US).

Leveson, N., 2004, ‘The Role of Software in Spacecraft Accidents’, AIAA Journal of Spacecraft and Rockets, 41:564–575 (US).

Maier, W., 2005, ‘Research Challenges for Systems-of-Systems’, IEEE International Conference on Systems, Man and Cybernetics (Waikoloa, HI, US).

Patel-Schneider et al., 2004, ‘OWL Web Ontology Language semantics and abstract syntax’, W3C recommendations, from http://www.w3.org/TR/owl-semantics/ (US).

Peckenpaugh, J., 2016, ‘Coast Guard Awards Multibillion-Dollar Ship, Aircraft Deal’, Government Executive Magazine, from http://www.govexec.com/dailyfed/0602/062502p1.htm, (US).

Petnga, L., and Austin, M.A., 2015, ‘Spatial Modeling and Reasoning for Cyber-Physical Systems’, International Conference on Complex Systems Engineering (Storrs, CT, US).

Petnga, L. and Austin, M.A., 2016, ‘An Ontological Framework for Knowledge Modeling and Decision Support in Cyber-Physical Systems’, Advanced Engineering Informatics, 30(1): 77–94 (US).

Rouquette, N., and Jenkins, S., 2010, ‘Transforming OWL2 Ontologies into Profiles Extending SysML’, 12th NASA-EST Workshop on Product Data Exchange (Oslo, Norway).

Wuthishuwong, C., and Traechtler A., 2013, ‘Vehicle to Infrastructure based Safe Trajectory Planning for Autonomous Intersection Management’, 13th International Conference on ITS Telecommunications (ITST)

Biography

Dr. Leonard Petnga is a Postdoctoral Associate at the Institute for Systems Research (ISR) at the University of Maryland College Park (UMCP). His research focuses on knowledge structures and Model-Based Systems Integration for Cyber-Physical Systems (CPS) with applications in Transportation, Aeronautic and Energy systems. He has also been a Cyber-Physical Systems Scholar in the Systems Integration Division (SID) at the US National Institute of Standards and Technology (NIST).

Dr. Mark Austin is an Associate Professor of Civil and Environmental Engineering at the University of Maryland, College Park, with a joint appointment in the Institute for Systems Research (ISR). Mark is Technical Director of the Master of Science in Systems Engineering (MSSE) Program at ISR. Mark has a Bachelor of Civil Engineering (First Class Honors) from the University of Canterbury, Christchurch, New Zealand, and M.S. and Ph.D. degrees in Structural Engineering from the University of California, Berkeley.

Dr. Mark R. Blackburn is an Associate Research Professor with Stevens Institute of Technology and primarily responsible for research focused on methods, modeling, simulation, visualization, and automated tools for reasoning about computer-based systems. He is the Principal Investigator (PI) on a Systems Engineering Research Center research task sponsored by NAVAIR investigating approaches to model-centric engineering.

Acknowledgment: This work was supported by the U.S. Naval and Air Command (NAVAIR) under task number SERC RT 157 from the Systems Engineering Research Center.