2018 Conference on Systems Engineering Research

The Data-Ontology-Rule Footing: A Building Block for Knowledge-based Development and Event-driven Execution of Multi-Domain Systems

Maria Coelho[a], Mark A. Austin[a]*, and Mark R. Blackburn[b]

[a]*Department of Civil and Environmental Engineering, University of Maryland, College Park, MD 20742, USA*

[b]*Stevens Institute of Technology, Hoboken, NJ 07030, USA*

Abstract. State-of-the-art approaches to semantic modeling of engineering systems focus on the capture and representation of knowledge within one or more domains. A common objective is development of ontologies for the comprehensive representation of knowledge within a domain; usually, far less effort is dedicated to the development of rules for the validation, use, and interaction of the domain with other domains. This paper proposes, in contrast, a framework for knowledge-based development and event-driven execution of multi-domain systems where data, ontologies, and rules within a domain have equal importance and are co-developed. We call this domain-level building block element the data-ontology-rule footing. Our preliminary work indicates that footings can be designed to be modular, and can support: (1) systems integration of domains through the use of rules that operate across multiple names spaces, and (2) synchronization of behavior transitions in multiple domains. Event-driven execution of semantic graphs is achieved through the use of event listeners attached to semantic graphs, which, in turn, can push updates to graphical views of domain behavior. We exercise these capabilities in a case study problem that examines fault tolerance of drone operations in a military mission.

Keywords: model-based systems engineering; ontology, rules, data-driven development; unmanned aircraft operations.

1. Introduction

1.1 Problem Statement

Our work is concerned with the comprehensive development of knowledge-based representations and event-driven behavior models for engineering systems spanning a multiplicity (e.g., physical, cyber, human, natural environment) of domains. We simply refer to such entities as multi-domain systems. Multi-domain systems in both the civilian and military sectors are of interest, as are those that have benefited from remarkable advances in technology -- particularly, computing, communications, and materials technology – over the past three decades. From a model-based systems engineering (MBSE) perspective, the latter opens doors to the development of systems having: (1) new forms of functionality, (2) superior levels of performance and agility in the face of pre- planned and unforeseen disruptions, and (3) economical operations over extended time horizons. While end-users applaud the benefits that these technological advances afford, systems engineers are faced with a multitude of new design challenges that can be traced to the presence of heterogeneous content, system-level behaviors that are distributed and concurrent, network structures that are spatial, interwoven and dynamic, and design conflicts that inevitably occur when a multiplicity of stakeholders from separate domains have competing objectives and concerns.

In a decentralized system structure no decision maker knows all of the information known to all of the other decision makers, yet as a group, they must cooperate to achieve system-wide objectives. Communication and information exchange are important

* Corresponding author. Tel.: +1-301-405-6627; fax: +1-301-405-6707.
E-mail address: austin@isr.umd.edu

to the decision makers because communication establishes common knowledge among the decision makers, which, in turn, enhances the ability of decision makers to make decisions appropriate to their understanding, or situational awareness, of the system state, and its goals and objectives. And even if the resulting cross-domain relationships are only weakly linked, they are nonetheless, still linked. When part of a system fails, there exists a possibility that the failure will cascade across interdisciplinary boundaries, thus making connected systems more vulnerable to various kinds of disturbances than their independent counterparts.[1,2] The introduction of automation into a system's operation, perhaps as a replacement for human-centered control, expands the range of design concerns that need to be addressed. For example, a new fundamental question is: How do we know that an automated management system will always do the right thing? A second important question is: If part of the system fails unexpectedly, what assurances do we have that the system will handle and recover from disruptions in a manner that is both sensible and timely? Our research is motivated by the belief that as systems become progressively complex, good answers to these questions will be unobtainable unless we have an ability to model and formally reason with the semantics and data in multi-domain systems.

1.2 Objectives and Scope

State-of-the-art approaches[3,4,5] to semantic modeling of engineering systems focus on the capture and representation of knowledge within one or more domains. A common objective is development of ontologies for the comprehensive representation of knowledge within a domain (e.g., sensors and sensor networks, satellites), with far less effort going to the development of rules for the validation, use, and interaction of the ontology with other ontologies. Two further problems include: (1) a lack of discipline in the development of ontologies for system development, and (2) a lack of computational support for evolution of semantic graphs in response to events. The first factor is one of the reasons why formal representations of ontologies have a reputation of being difficult to develop and use. As a case in point, the integrated model-centric engineering ontologies (IMCE) developed at JPL (Jet Propulsion Laboratory) during the 2000-2010 era[5], the electrical engineering ontology (i.e., electrical.owl) imports the mechanical engineering ontology (i.e., mechanical.owl). Both the electrical and mechanical engineering ontologies import a multitude of foundation ontologies (e.g., analysis.owl, mission.owl, base.owl, project.owl, time.owl) and make extensive use of multiple inheritance mechanisms in the development of new classes. The result is ontologies containing more than several hundred classes, with some classes containing three or four dozen data and object properties. Notions of simplicity in system design through modularity of semantic models (e.g., bundling of ontologies and rules) do not seem to exist. SysML[6] steps around these difficulties by providing a variety of diagram types (e.g., use case, requirements, block, activity, sequence) that a engineer can use to visually describe what a proposed system will do and how it will do it.

This paper proposes a framework for knowledge-based development and event-driven execution of multi-domain systems where the complementary rolls of data, ontologies, and rules are highlighted and have equal importance. We call the building block for this framework the data-ontology-rule footing. The remainder of this paper proceeds as follows: Section 2 introduces our approach to semantic modeling and rule-based decision making. Section 3 describes the data-ontology-rule footing, its use in the construction of multi-domain semantic models, and formalisms for the modeling and visualization of domain ontologies. The case study (see Section 4) exercises these ideas through an examination of fault tolerance of drone operations in a military mission.

2. Semantic Modelling and Rule-Based Decision Making

2.1 Framework for Semantic Modelling

Model-based systems engineering development is an approach to system-level development in which the focus and primary artefact of work is models, as opposed to documents. A tenet of our work, which is motivated by the steady trend toward engineering systems becoming increasingly complex, is that methodologies for strategic approaches to design will employ semantic descriptions of application domains, and use ontologies and rule-based reasoning to enable validation of requirements, automated synthesis of potentially good design solutions, and real-time management of interactions among participating domains.

Fig 1. is adapted from our recent work[3], and shows the essential elements of a framework for semantic modelling of multi-domain systems. On the left-hand side, textual requirements are defined in terms of mathematical and logical rule expressions for design rule checking. Engineering models will correspond to a multitude of network and hierarchy graph structures for the system structure and behaviour. Behaviours will be associated with components, with discrete behaviour being modelled with finite state machines (FSM) and statecharts. System level behaviour will correspond to networks of FSM / statechart behaviours. Ontology models and rules glue the requirements to the engineering models and provide a platform for the development of system structures, adjustments to system structure over time, and system behaviour. Collections of ontologies and rules will be developed for specific domains and well as meta-domains (e.g., space, time, physical units) that are an integral part of all engineering domains.

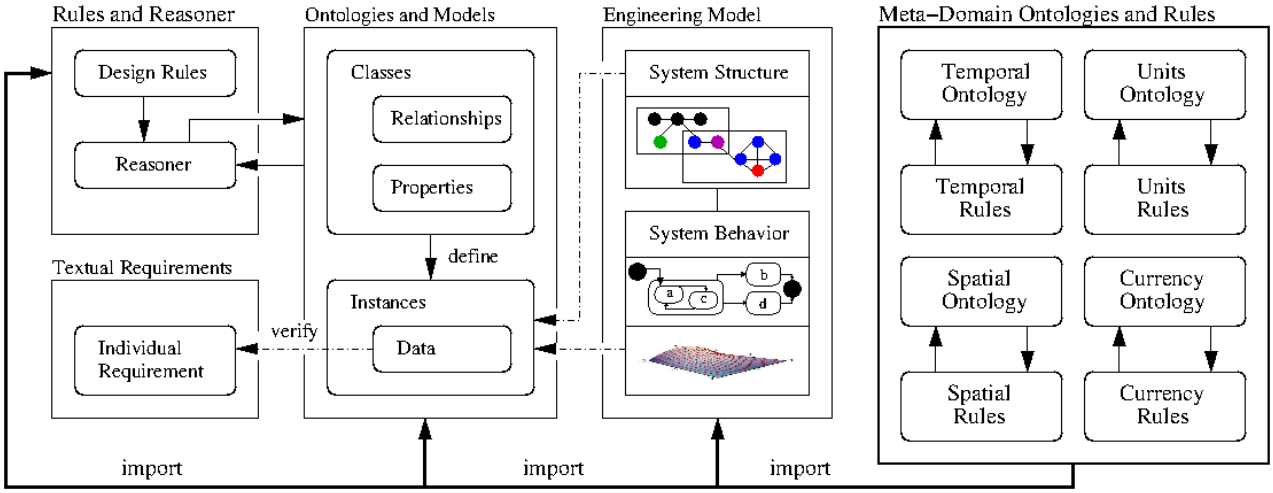

Fig 1. Framework for semantic modelling of multi-domain systems.

2.2 Use of Semantic Web Technologies for Rule Checking

Semantic models (see Fig. 1) consist of ontologies, graphs of individuals (specific instances of classes), and inference-based rules. An ontology is a formal and explicit representation of the concepts, referred to as classes, and data and object properties. Support for relationships among classes is provided by object properties. Individuals are instances of ontology concepts, and their purpose is to represent the data in a domain. Inference rules and their associated reasoning mechanisms provide a way to derive new information based on the existing data stored in the ontology in the form of*: if <condition> then <consequences>*. A key benefit of this formalism is that ontologies and rules are human readable, yet they can also be compiled into code that is executable on machines.

2.3 Working with Jena and Jena Rules

Our experimental software prototypes employ Apache Jena and Jena Rules. Apache Jena[7] is an open source Java framework for building Semantic Web and linked data applications. Jena provides the interfaces for code development and the construction of resource description framework (RDF) graphs and semantic descriptions of domains in OWL (Web Ontology Language). The Jena rule-based inference subsystem allows for a range of inference engines to be plugged into Jena, and Jena Rules is one such engine. Jena Rules uses facts and assertions described in OWL to infer additional facts from instance data and class (ontology) descriptions. As we will soon see in the case study prototype, domain specific ontologies can import and use multi-domain (or cross-cutting) ontologies. And rules can be distributed among domains and can be programmed to respond to events that involve (or affect) reasoning among a multiplicity of domains. These inferences result in event-driven structural transformations of the semantic graph model.

3. Proposed Methodology

In state-of-the-art development of semantic models, two common strategies are: (1) provide classes and data properties for all possible configurations within a domain, as well as linkages to related domains, and (2) maximize generality of the knowledge representation through extensive use of multiple inheritance mechanisms. In practical engineering settings the result can be ontologies containing hundreds of classes, each having dozens of data and object properties defined through inheritance mechanisms. In our view, notions of “simplicity of system design” through modularity of semantic models (e.g., binding of ontologies and rules) do not seem to exist. This practice makes MBSE of multi-domain systems more complicated than it needs to be and misses opportunities for adding value to the system development process.

3.1 The Data-Ontology-Rule Footing

In a step toward mitigating these complexities, we propose a semantic modelling framework (see Fig. 2) that supports: (1) concurrent data-driven development of domain data models, ontologies and rules, and (2) executable processing of incoming events.

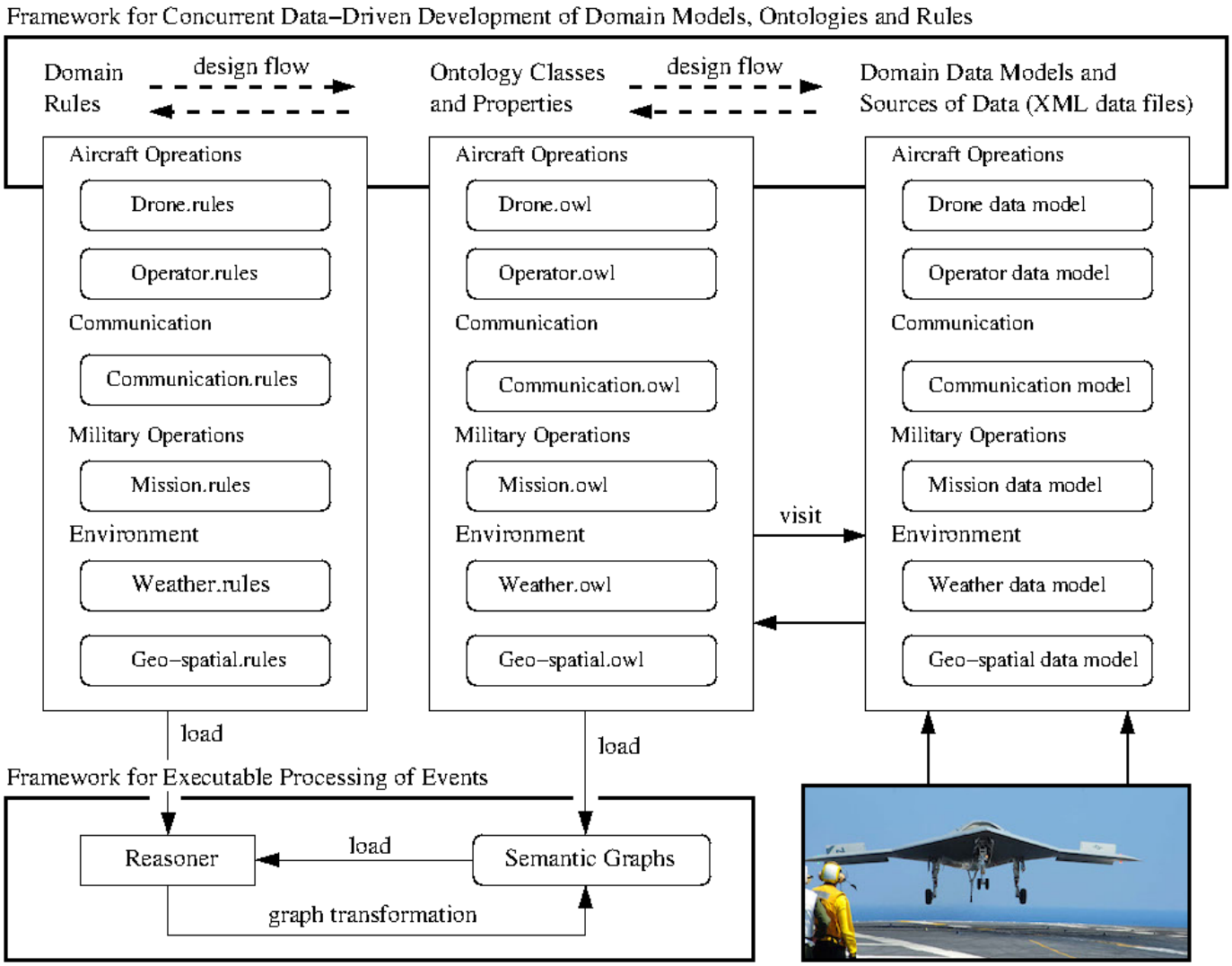

Fig. 2. Framework for: (1) data-driven development of multi-domain systems, ontologies and rules, and (2) executable processing of events.

Instead of creating a few (very large) ontologies and a few rules (perhaps for validation of semantic graph properties), our goal is to put development of the data models, ontologies, and rules on an equal footing. We call this arrangement the data-ontology-rule footing. Each row of the footprint contains rules, ontologies and data models for a specific domain or design concern. And as we will soon see in the case study problem, domains interact through rule interactions. Fig. 2. shows, for example, data-ontology-rule footings for the semantic modelling of drone operations in a military mission. The overall system representation is highly multi-disciplinary covering concerns for drone operations and communications, the military mission, and the potential impact environmental factors may play in affecting the feasibility of proposed drone operations. Our second strategy is to minimize the use of multiple inheritance in the specification of OWL ontologies and, instead, explore opportunities for replacing them with object property relations.

A key advantage of this approach is that it forces designers to provide data that are needed for decision making, and increases the likelihood that data not needed for decision making will be left out. The latter occurs because the co-development of rules, ontologies and data sources forces developers to think about the chain of dependency relationships that allow the rules will work -- rules require data and object properties from the ontologies, which in turn, require data values from the data models shown along the right-hand side of Fig. 2. Rules will be developed for the verification of semantic properties (e.g., has a specific data property been initialized?) and for reasoning with data sources and incoming events, possibly from a multiplicity of domains. Implementation of the latter leads to semantic graphs that can dynamically adapt to the consequences of incoming data and events (e.g., a weather event) acting on the system.

3.2 Data-Driven Approach to Generation of Individuals in Semantic Graphs.

By themselves, ontologies and rules provide a framework for the representation and transformation of semantic graphs, but they do not contain actual data (i.e., individuals in the semantic graph). In a practical implementation the data will come from a variety of sources and be highly heterogeneous (see the right-hand side of Fig. 2).

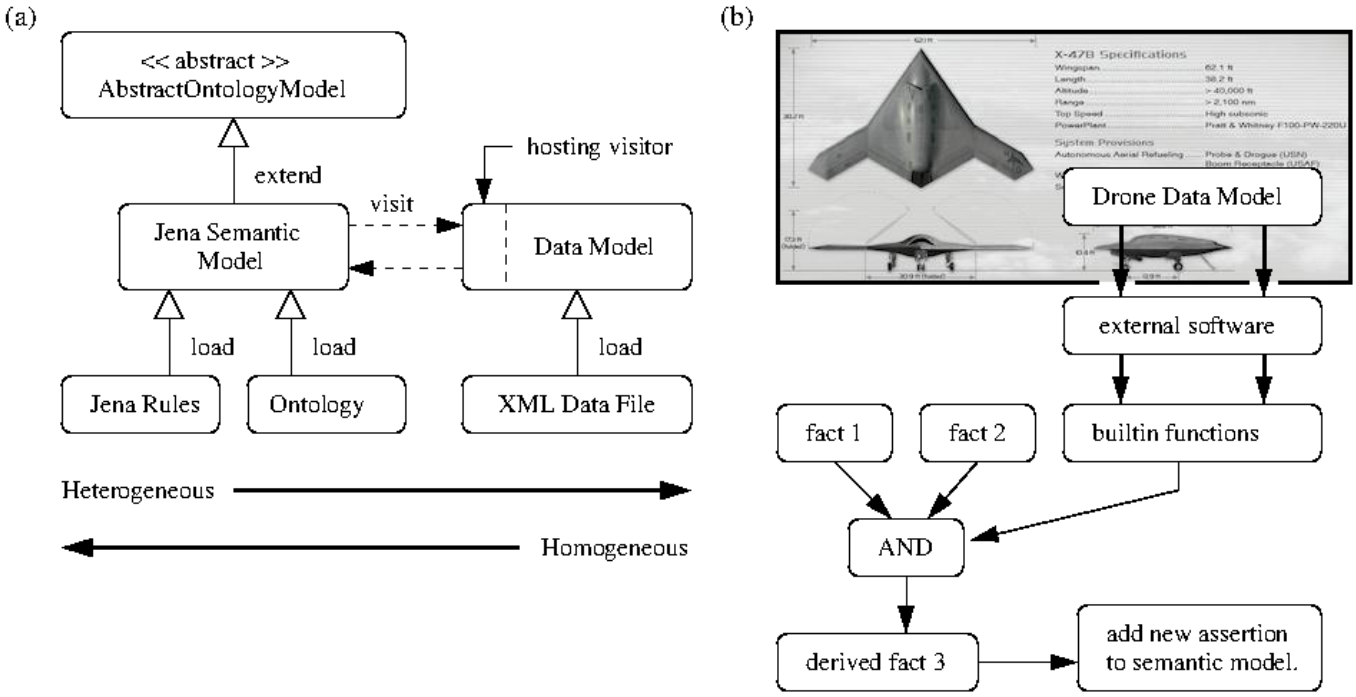

Fig 3. Data-driven approach to semantic modelling: (a) generation of individuals, (b) forward chaining facts and results of built in function evaluations.

We address this challenge with a data-driven approach to generation of semantic models, where as illustrated in Fig. 3a, semantic models obtain data on individuals by systematically visiting the data models that are designed to host visitors. This is an implementation of the visitor software design pattern. Generally speaking, data models obtain their data from data files and a variety of online sources (e.g., weather and geographic information servers) and can be highly heterogeneous. SysML models of system structure and behavior can also be viewed as data. The challenge in parsing these data sources can be vastly simplified if the XML data files and corresponding data models are designed to work with JAXB (the XML binding for Java). We are currently exploring opportunities for using JAXB in the description of general-purpose components, their finite state machine (or statechart) behaviors, and various specifications for performance and functionality. This aspect of our research is a work in progress. It is important to note that when a semantic model visits a data model in search for data, the results have to fit into one of eight basic data types supported in Jena (i.e., integer and floating point numbers, true and false, and character strings). While this transformation simplifies the systems integration problem, and makes the overall data problem more homogeneous, it forces designers to think carefully about how the data will be used by the ontologies and rules. This leads to novel solutions on the semantic level – for example, it is much easier to store sets of geo-spatial coordinate data as character strings than as individual data points. Backend software routines associated with the rule checking procedures (see Fig 3b) can parse these strings and work with the individual data values. Once the data has been transferred to the Jena Semantic Model and used to create an ontology instance, the rules are applied. It is important to note that while Fig. 2 implies a one-to-one association relationship between semantic graphs and data, in practice a semantic graph model might visit multiple data models to gather data on individuals.

3.3 Organization and Visualization of Domain Ontologies

Domain-specific footings are conveniently organized into groups along the lines of their contribution to the system-level functionality. In Fig. 2, for example, data-ontology-rule footings are provided for aircraft operations, communications, military operations and the environment. A footprint model to support fault detection and diagnostic analysis of equipment in buildings would contain data, ontologies and rules for the building geometry and operations, the building occupants, sensors, equipment, and procedures to be followed in the fault detection and diagnostic procedure.

Visualization of domain ontologies and their connection to rules is an essential component of verifying our models make sense. Fig. 4 shows a simplified semantic model for the structure and operation of a drone having behaviour that can be described by a statechart (i.e., states, transitions, guard conditions). The limitations of drone operations (e.g., maximum payload) can be represented by a collection of specifications. We employ red rectangles with heavy dashed edges to highlight classes that participate in rule checking.

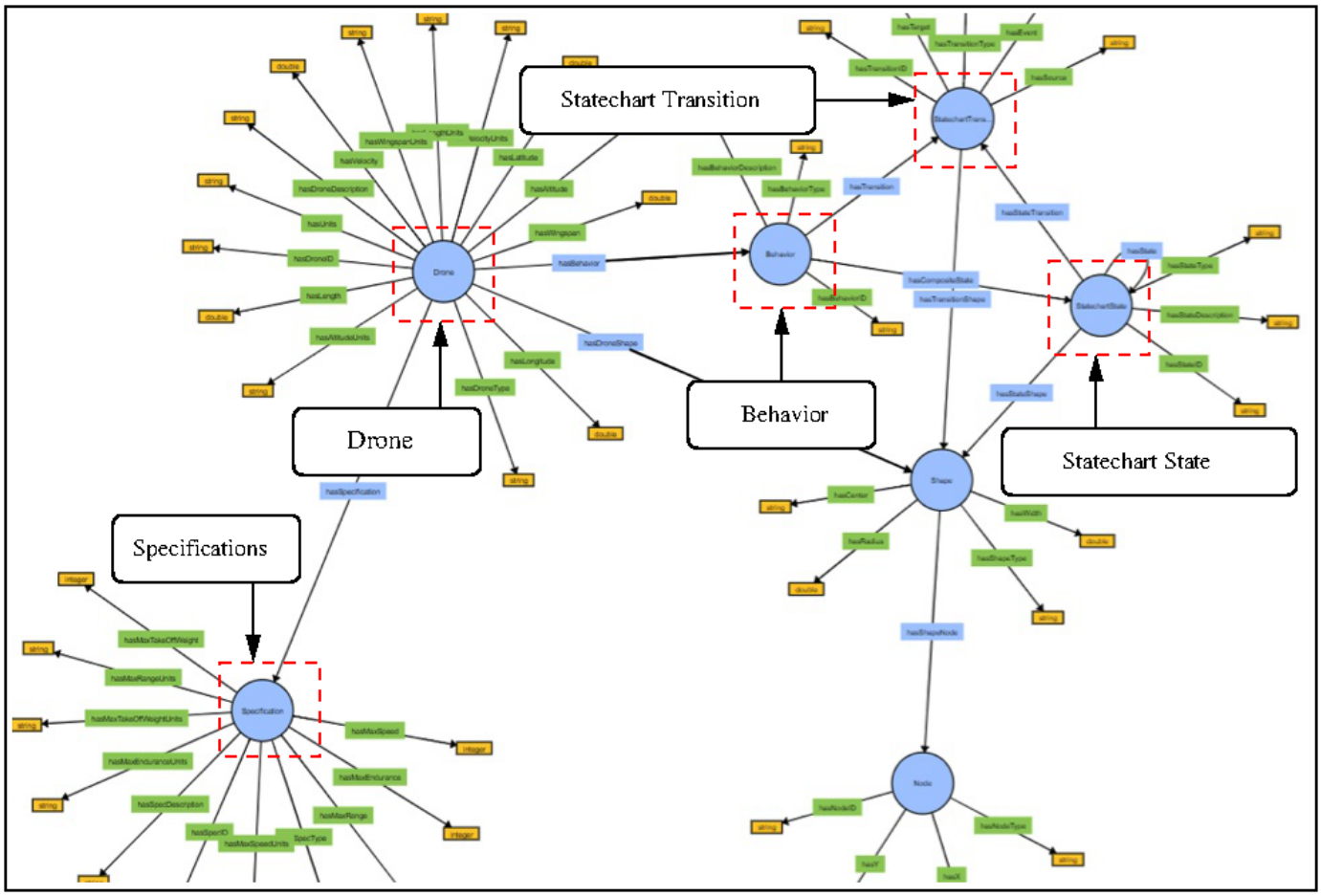

Fig 4. Simplified semantic model for the structure, behaviour, and performance specification of an unmanned aircraft (drone).

4. Case Study Simulation

4.1 Problem Description

Our case study simulation corresponds to a simple scenario where a drone (an unmanned air vehicle) is involved in a reconnaissance operation of a geographical area of military interest. The modelling objective is to obtain a high-level description of the participating processes – a drone, a human operator, a communications channel between the operator and the drone -- and their interactions in response to an unexpected disruption.

The participating domains – rules, ontologies and data models -- are shown along the rows of Fig. 2. We assume (see Fig. 5) that the drone, pilot operator, and communication system behaviours are concurrent and can be represented by a collection of statechart behaviour models. The flight operations will follow a set of states and transitions as the drone travels to and from the mission area. Two modes of flight operation and control are supported: manual (or human-centred) control occurs when the drone is piloted from a remote location. Autonomous operations and control are independent of the remotely positioned pilot and state of the drone-operator communications system, and offer advantages for missions of long duration and/or the area of military interest is large.

At this point, statechart behaviour models are hardcoded in Java, which requires developers to work at a very low level of detail. To make the development process less tedious and hopefully more scalable, we are exploring opportunities for embedding a variant of the statechart XML language specification[8] within our systems data model specification for components – a drone will be modelled as a component – and then automatically generating the model and visualization for statechart behaviors. We are also extending the Whistle scripting environment[9] so that it can handle the specification and integration of data-ontology-rule models.

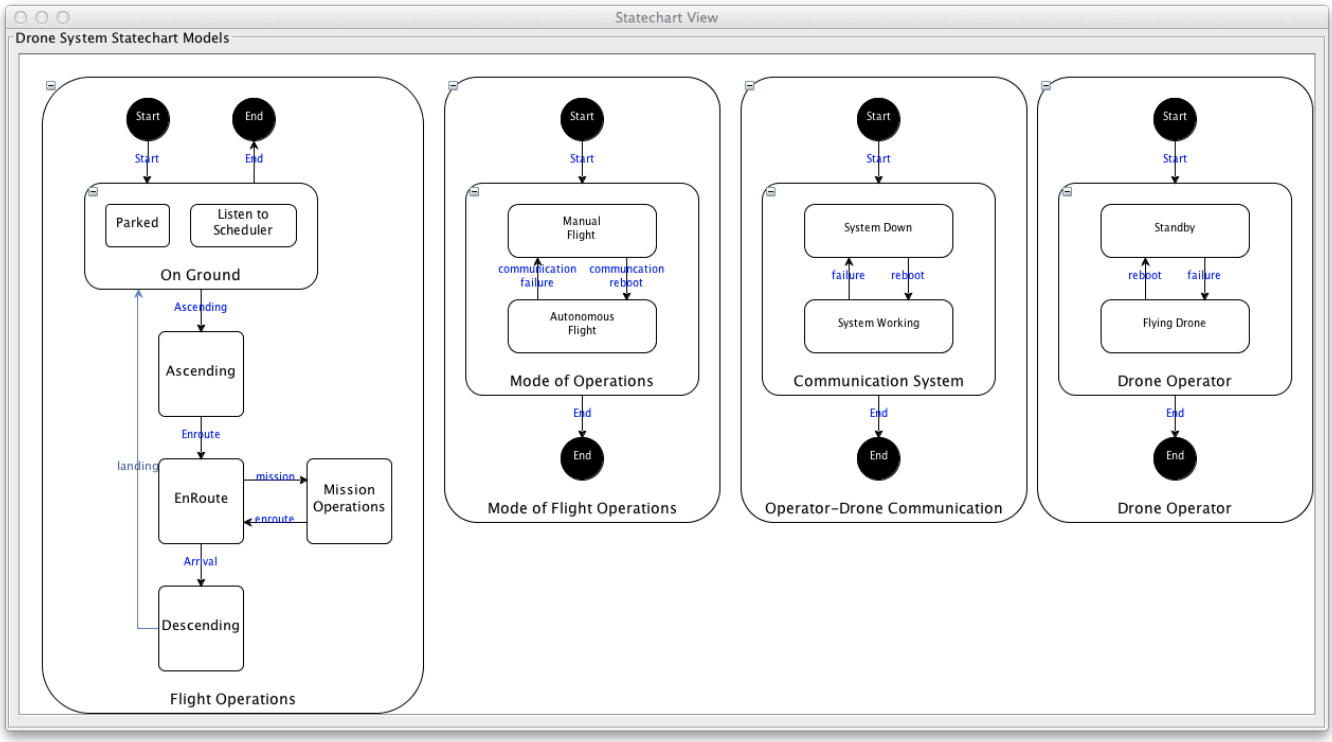

Fig 5. Statechart models for drone, communication system, and human operator behaviour.

4.2 Failure of the Communication System

Now suppose that the ontology-drone communication channel is active, but then suddenly fails. This event triggers a chain of rules and actions to adjust the flight operations from manual to autonomous operation. Rules and actions act across a multiplicity of domains.

```
@prefix dr:  <http://austin.org/drone#>.
@prefix op:  <http://austin.org/operator#>.
@prefix cm:  <http://austin.org/communication#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Active Communication ....

[ ActiveCommunication: (?x rdf:type dr:Drone) (?y rdf:type op:Operator) (?z rdf:type cm:Communication)
                       (?y op:hasConnection ?z) (?x dr:hasConnection ?z) -> (?z cm:isActive af:True)

// Rule 02: Manual Operations ....

[ ManualOperations: (?x rdf:type dr:Drone) (?y rdf:type op:Operator) (?z rdf:type cm:Communication)
                    (?z cm:isActive af:True) -> (?y op:isFlyingDrone af:True) (?x dr:isInAutonomousFlight af:False)]

// Rule 03: Drone Communication Failure ....

[ NoCommunication: (?x rdf:type dr:Drone) (?y rdf:type op:Operator) (?z rdf:type cm:Communication)
                   (?y op:hasConnection ?z) (?x dr:lostConnection ?z) -> (?z cm:isActive af:False)

// Rule 04: Autonomous Operations ....

[ AutonomousOperations: (?x rdf:type dr:Drone) (?y rdf:type op:Operator) (?z rdf:type cm:Communication)
                        (?z cm:isActive af:False) -> (?y op:isFlyingDrone af:False) (?x dr:isInAutonomousFlight af:True)]
```

Fig. 6. Jena rules for transformation of drone, human pilot, and communication system semantic models.

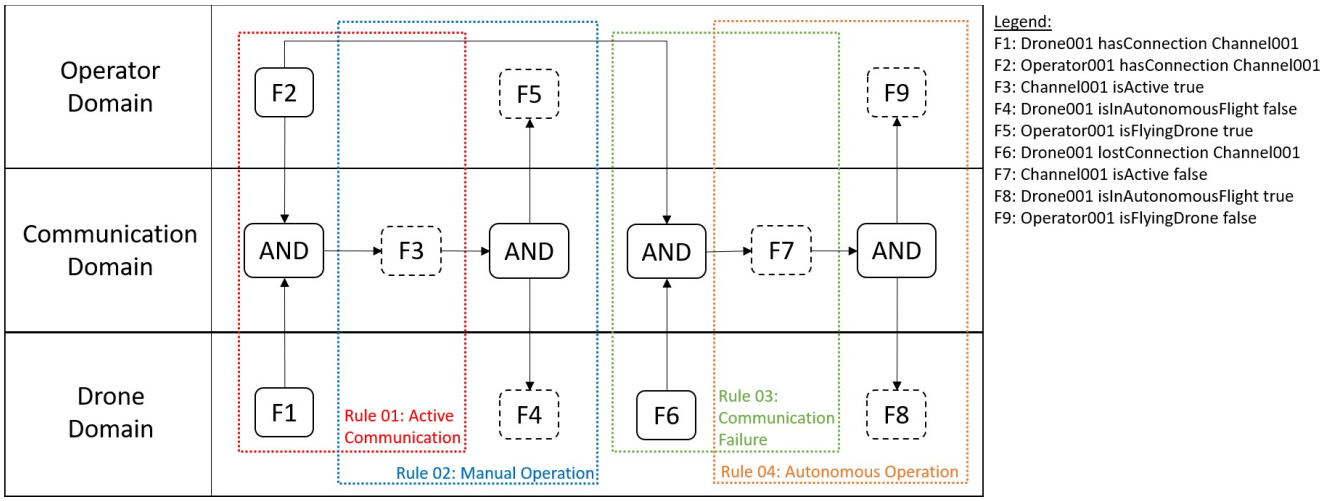

Fig. 7. Snapshot of multi-domain evaluation and forward chaining of rules.

Fig 6 shows details of the Jena rules responsible for the transformation of drone, human operator and communications semantic models in response to the communications system failure. Fig 7. is a snapshot of the corresponding multi-domain evaluation and forward chaining of rules. Rules 01 gathers data from the operator and drone domains and updates the semantic graph to acknowledge the communication channel is active (F3). Then, Rule 02, set parameters in the drone and operator domains for manual operation of the drone. Rule 03 is triggered when the communication system fails – manual operation of the drone is no longer possible – the operator status switches to standby (F9) and the mode of drone operation switches from manual to autonomous (F8). The systems integration of data and rules acting across multiple domains is possible because Jena rules are capable of working across multiple name spaces (see the declarations at the top of Fig. 6), Finally, the actions associated with the activation of these rules can also be pushed to the statechart visualization of behaviour, thereby, providing visual feedback and verification for how the system is responding to the communication network failure.

5. Conclusions and Future Work

The data-ontology-rule footing is a new approach to the development and event-driven execution of semantic models spanning multiple domains. Support for the forward chaining of rules across domains is a powerful mechanism that allows one to develop software that "thinks like a human". Our on-going research will include development of tools to streamline the specification of data models (structure and behaviour), and ontologies and rules.

Acknowledgements

This work was supported by grants RT170 and RT195 from the NAVAIR / SERC / DoD on Systems Engineering Transformation.

References

1. Austin M.A., Delgoshaei P., and Nguyen A. , Distributed Systems Behavior Modeling with Ontologies, Rules, and Message Passing Mechanisms, 13th Annual Conference on Systems Engineering Research (CSER 2015), Hoboken, New Jersey, March 17-19, 2015.
2. Coelho M., Austin M.A., and Blackburn M. , Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Many-to-Many Association Relationships, The 12th International Conference on Systems (ICONS 2017), Venice, Italy, April 23-27, 2017, pp. 10-15.
3. Petnga L. and Austin M.A. , An Ontological Framework for Knowledge Modeling and Decision Support in Cyber-Physical Systems, Advanced Engineering Informatics, Vol. 30, No. 1, January 2016, pp. 77-94.
4. Russomanno, D.J., Kothari, C. and Thomas, O. Sensor Ontologies: From Shallow to Deep Models. In 37th Southeastern Symposium on System Theory, pages 107–112. IEEE, 2005.
5. Wagner D.A., Rouquette N., Bennett M.B., Jenkins S., Karban R. and Ingham M., An Ontology for State Analysis: Formalizing the Mapping to SysML, Proceedings of 2012 IEEE Aerospace Conference, Big Sky, Montana, March, 2012.
6. Fridenthal S., Moore A., and Steiner R., A Practical Guide to SysML, MK-OMG, 2008.
7. Apache Jena:, "An Open Source Java framework for building Semantic Web and Linked Data Applications. For details, see https://jena.apache.org/," 2016.
8. State Chart XML: State Machine Notation for Control Abstraction (SCXML), Wikipedia. See: https://en.wikipedia.org/wiki/SCXML (Accessed April 13, 2018).
9. Delgoshaei P., Austin M.A., and Pertzborn A. , A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems, International Journal On Advances in Systems and Measurements, Vol. 7, No's 3-4, December 2014, pp. 223-238.