Semantic Behavior Modeling and Event-Driven Reasoning for Urban System of Systems

Maria Coelho and Mark A. Austin
Department of Civil and Environmental Engineering,
University of Maryland, College Park, MD 20742, USA
E-mail: memc30@hotmail.com; austin@isr.umd.edu

Mark Blackburn
Stevens Institute of Technology,
Hoboken, NJ 07030, USA
E-mail: mblackbu@stevens.edu

Abstract—**Modern urban infrastructure systems are defined by spatially distributed network structures, concurrent subsystem-level behaviors, distributed control and decision making, and interdependencies among subsystems that are not always well understood. The study of the interdependencies within urban infrastructures is a growing field of research as the importance of potential failure propagation among infrastructures may lead to cascades affecting multiple urban networks. There is a strong need for methods that can describe the evolutionary nature of "system-of-systems" (SoS) as a whole. This paper presents a model of system-level interactions that simulates distributed system behaviors through the use of ontologies, rules checking, message passing mechanisms, and mediators. We take initial steps toward the behavior modeling of large-scale urban networks as collections of networks that interact via many-to-many association relationships. The prototype application is a collection of families interacting with a collection of school systems. We conclude with ideas for scaling up the simulations with Natural Language Processing.**

Keywords-Systems Engineering; Ontologies; Behavior Modeling; Mediator; Network Communication.

I. INTRODUCTION

This paper is concerned with the development of modeling abstractions, procedures, and prototype software for the behavior modeling of urban systems of systems with ontologies, rules and message passing mechanisms. It builds upon our previous work [1], [2] on distributed systems behavior modeling with semantic web technologies.

A. Problem Statement

The past century has been marked by outstanding advances in technology (e.g., the Internet, smart mobile devices, cloud computing) and the development of urban systems (e.g., transportation, electric power, waste-water facilities and water supply networks, among others) whose individual resources and capabilities are pooled together to create new, more complex systems that offer superior levels of performance, extended functionality and good economics. While end-users applaud the benefits that these systems of systems afford, model-based systems engineers are faced with a multitude of new design challenges that can be traced to the presence of heterogeneous content (multiple disciplines), network structures that are spatial, multi-layer, interwoven and dynamic, and behaviors that are distributed and concurrent.

Large-scale urban systems do not follow a standard cradle-to-grave lifecycle. Instead, the constituent domains within a city evolve over extended periods of time in response to external forces (e.g., the need for economic expansion) and disruptive events (e.g., the need for planning of relief actions in response to a natural disaster). In both cases, planning of urban operations is complicated by the large scale of modern cities, the large number of constituent behaviors, and multiple dimensions of interdependency among physical, cyber and geographic systems [3]. These facts are what makes cities "system of systems," rather than just systems, and they change the very nature of systems design and management. For example, in order for the communication among the participating urban domains to occur in an orderly and predictable way, designers need to pay attention to the boundaries (or interfaces) of domains [4]. Similar concerns exit for the replacement of aging infrastructure. In his article on the topic of complex system failure "How Complex Systems Fail," Cook discusses how complex systems are prone to catastrophic failure, due to the impractical cost of keeping all possible points of failure fully protected, and even identifying them all [5]. When part of a system fails, or perhaps an unexpected combination of localized failures occurs, there exists a possibility that the failure will cascade across interdisciplinary boundaries to other correlative infrastructures, and sometimes even back to the originated source, thus making highly connected systems more fragile to various kinds of disturbances than their independent counterparts. Figure 1 presents an overview of some generic interdependencies among key infrastructure sectors: oil and natural gas, electricity, transportation, water, and communications.

B. Scope and Objectives

In order to understand how cascading failures might be best managed, it is necessary to have the ability to model events and the exchange of data/information at the interdependency boundaries, and to model their consequent effect within a subsystems boundary. This points to a strong need for new capability in modeling and simulation of urban infrastructure systems as system-of-systems, and the explicit capture of infrastructure interdependencies. We envision such a system having an architecture along the lines shown in Figure 2, and eventually, tools such as OptaPlanner [7] providing strategies for real-time control of behaviors, assessment of domain resilience and planning of recover actions in response to severe events. This paper presents a model of distributed system-level behaviors based upon the combined use of ontologies, rules checking, and message passing mechanisms, and explores

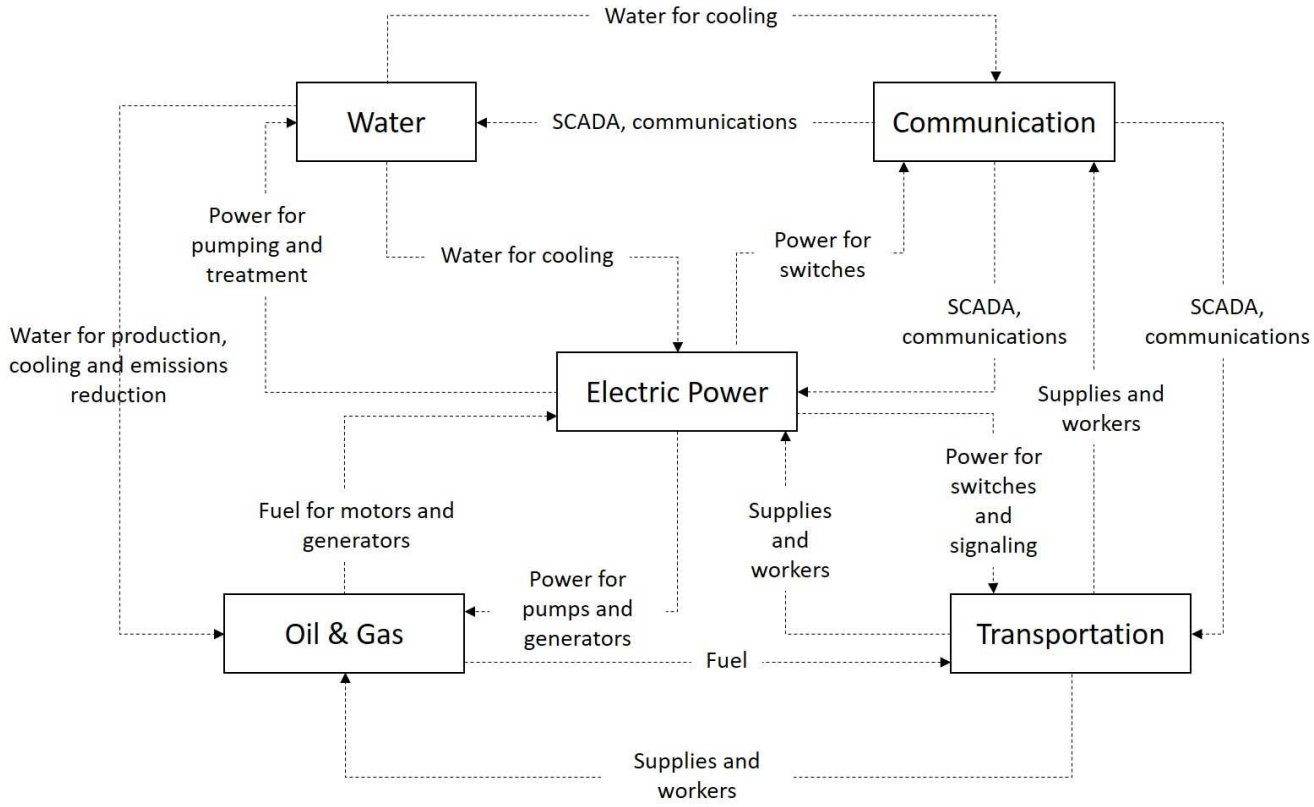

Figure 1. Illustration of the interdependent relationship among different infrastructures [6].

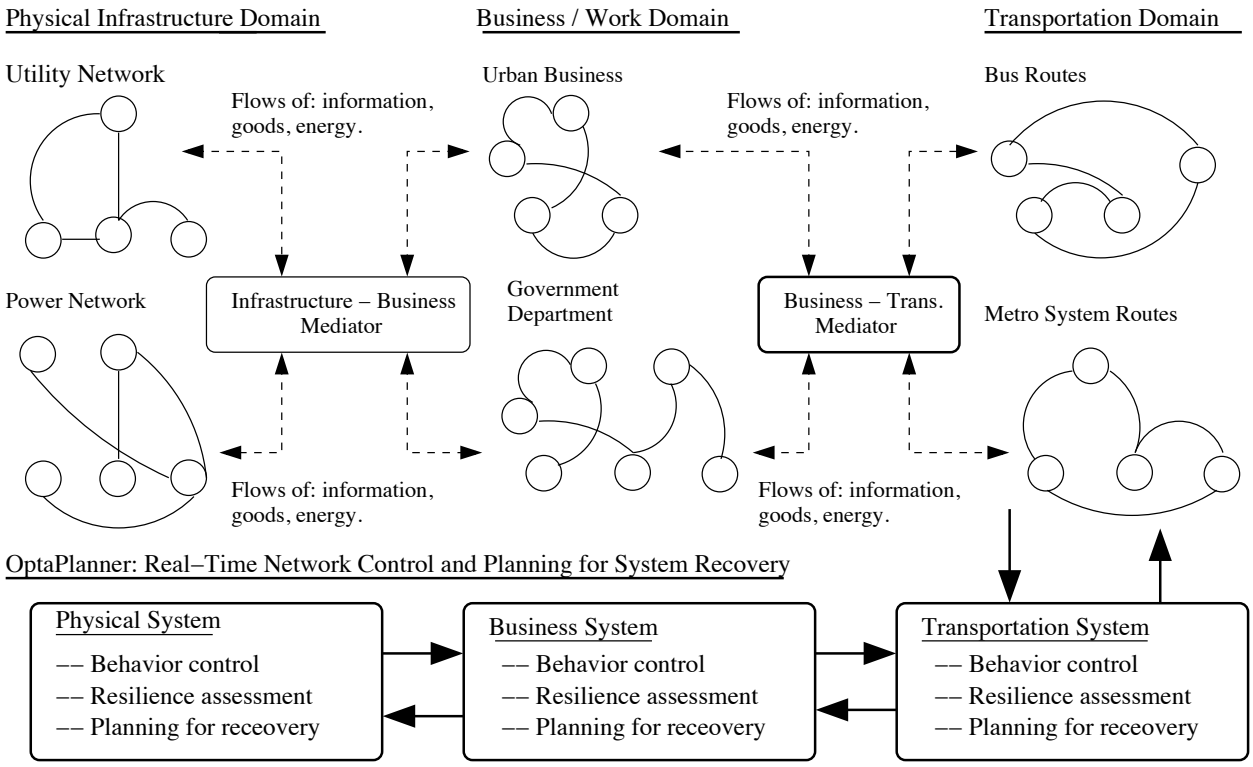

Figure 2. Architecture for multi-domain behavior modeling with many-to-many associations.

opportunities for modeling urban systems as collections of discipline-specific (or community) networks that will dynamically evolve in response to events. As illustrated in Figure 2, each community will have a graph that evolves according to a set of community-specific rules, and subject to satisfaction of constraints. The contributions of this paper are three-fold:

Contribution 1. We provide a framework for modeling concurrent, directed communication between all entities composing a system. The architecture builds upon the framework presented by Austin et al. [2], and in particular, extends the distributed behavior modeling capability from one-to-one association relationships among communities to many-to-many association relationships among networked communities.

As illustrated in Figure 3, one-to-one association relationships can be modeled with exchange of messages in a point-to-point communication setup.

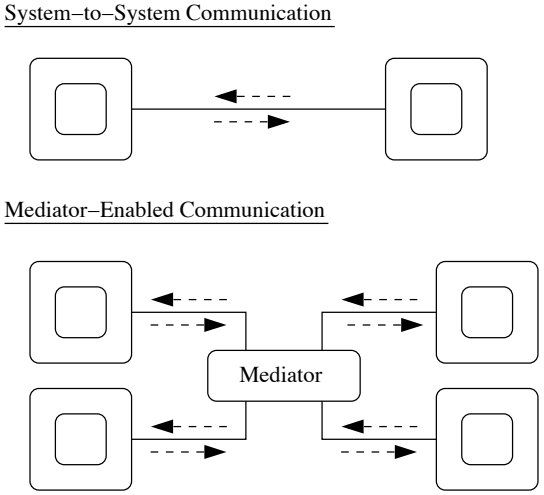

Figure 3. Framework for communication among systems of type A and B.

The top part of the figure shows point-to-point communication in a one-to-one association relationship between systems. Mediator enabled communication in a many-to-many association relationship among systems are shown in the bottom half of the figure. Many-to-many association relationship among systems are enabled by collections of mediators. Each ontology is paired with an interface for communication and information exchange with other ontologies. From a communications standpoint, this architectural setup is simpler than what is commonly found in multi-hop routing of messages in wireless sensor networks.

Contribution 2. We employ a novel use of software design patterns and Apache Camel [8] [9], to allow communication management in the urban system of systems framework. The visitor design pattern is also implemented to allow for data retrieval.

Contribution 3. We explore mechanisms for incorporating notions of space and time in event-driven reasoning processes for urban decision making.

The remainder of this paper proceeds as follows: Section II covers related research that has been done in critical infrastructure simulation. Section III explains how Semantic Web technologies [10] can be employed for semantic modeling and rule-based reasoning. Section IV explains the advantages of constructing a model with time and space reasoning. Section V describes several aspects of our work in progress, including: (1) Distributed system behavior modeling with ontologies and rules, and (2) Use of mediators for behavior modeling of distributed systems having many-to-many association relationships among connected networks. We describe the software architecture for an experimental platform for assembling ensembles of community graphs and simulating their discrete, event-based interactions. In Section VI we exercise this capability with an application involving collections of families interacting with multiple school systems. Domain-specific ontologies are developed for family and school system domains, which, in turn, import spatial (geometry) ontologies and rules [11] [12]. We conclude with ideas for scaling up the simulations with Natural Language Processing (NLP).

II. Related Work

A. Critical Infrastructure

Experience over the past decade with major infrastructure disruptions, such as the 2011 San Diego blackout, the 2003 Northeast blackout, and Hurricane Irene in 2011, has shown that the greatest losses from disruptive events may be distant from where damages started. For example, Hurricane Katrina disrupted oil terminal operations in southern Louisiana, not because of direct damage to port facilities, but because workers could not reach work locations through surface transportation routes and could not be housed locally because of disruption to potable water supplies, housing, and food shipments [13]. Interdependencies constitute a significant dimension for understanding system vulnerability. Examples of vulnerabilities where systems could be brought down are an important basis for identifying interdependencies and focusing on those that are critical. Using data provided by references [14], [15] and [16], Table I provides some examples of faults that propagate through interdependency relationships of different critical infrastructure sectors.

In its October 1997 report to the U.S. President, the President's Commission on Critical Infrastructure Protection identified the nation's eight critical infrastructures. It recognized the importance that interdependencies play in their continuous and reliable operation, as well as the increased security concerns and risks associated with them [17]. Although interdependencies are a complex and difficult problem to analyze, over the past twenty years increased effort by the operational, research and development, and policy communities has led to improvements in our ability to identify and understand interdependencies among infrastructures, and their influence on infrastructure operations and behavior. As a case in point, Rinaldi and co-investigators [3] have proposed a multi-dimensional taxonomy to frame the major aspects of interdependencies: types of interdependencies, infrastructure environment, coupling and response behavior, infrastructure characteristics, types of failures, and state of operations. These dimensions point to the need for development of a comprehensive architecture for interdependency modeling and simulation. Many models and simulations exist for individual

	Energy: Oil and Gas	Energy: Electricity	Transportation	Water	Communication
Energy: Oil and Gas		No fuel to operate power plant motors and generators	No fuel to operate transport vehicles	No fuel to operate pumps and treatment. Gas pipeline failure located beneath roads may contaminate water pipeline also located beneath roads	No fuel to maintain temperatures for equipment; no fuel to backup power
Energy: Electricity	No electricity for extraction and transport (pumps, generators, control systems)		No power for traffic lights, rail systems, street lights. Passengers may be trapped inside trains. Air transport may become compromised due to to the loss of communications and unlit runways.	No electric power to operate pumps and treatment leading to potential water quality issues and pumping issues in buildings. No power to operate flood protection systems.	No energy to run cell towers and other transmission equipment
Transportation	Delivery of supplies and workers interruption	Delivery of supplies and workers interruption		Delivery of supplies and workers interruption	Delivery of supplies and workers interruption
Water	No water available for production, cooling, and emissions reduction	No water available for production, cooling, and emissions reduction. Water pipeline failure located beneath roads may damage power lines located beneath and above roads	No water for vehicular operation. Water pipeline failure located beneath roads may interrupt traffic.		No water available for equipment cooling. Water pipeline failure located beneath roads may damage cables and underground wiring also located beneath roads, and above ground networks aligned with roads
Communication	Inability to detect breakages and leaks. Remote control of operations interruption	Inability to detect and maintain operations and electric transmission	Inability to identify and locate disabled vehicles, rails, and roads. No provision of user service information.	Inability to detect and control water supply and quality	

TABLE I. Summary of urban faults propagated by interdependencies between critical infrastructure systems.

infrastructure behavior, but simulation frameworks that allow for the coupling of multiple interdependent infrastructures to address infrastructure protection, mitigation, response, and recovery issues are only beginning to emerge.

B. Urban Interdependence Simulators

Pederson et al. [18] have compiled a survey on contemporary research on critical infrastructure modeling and simulation. This study showed a wide variety of ideas proposed in recent years, and observed that the vast majority of these recently implemented frameworks are based on agent-based technology. In an effort to overcome some of the limitations associated with agent-based frameworks, such as scalability and distorted results, Rahman et al. [19] proposed a new type of framework for simulating infrastructure interdependencies. The proposed model captures physical interdependencies among different critical infrastructures using precise mathematical expression. Each entity and interaction between infrastructures is mapped to a single equivalent semantic. In this way, components defined in physical layer can interact with the decision making layer through event forwarding mechanisms.

C. Urban System Ontologies

A detailed discussion the use of ontologies in urban development projects can be found in Falquet, Metral, Teller and Tweed [20]. Ontologies have been developed for the geographic information sector, to model interconnections (mediators) among urban models, and to describe urban mobility processes. Extensive studies have been conducted on the development of ontologies for the geography markup language (GML) and CityML, the XML markup language for cities [21].

As part of the recent interest in Smart Cities, researchers have proposed so-called smart city ontologies. A close examination reveals that they contain an exhaustive list of things you might find in a smart city, and proposals for relationships among things, but are otherwise not smart at all.

Our viewpoint is that ontologies (including classes and their associated data and object properties) need to be developed alongside rules, and that the resulting semantic modeling systems need to be executable and capable of event-driven processing. A notable effort in this direction is the DogOnt ontology and rules for statechart behavior modeling of devices in home automation [22].

III. Semantic Modeling and Rule-based Decision Making

A. Framework for Semantic Modeling

Model-based systems engineering development is an approach to systems-level development in which the focus and primary artifacts of development are models, as opposed to documents. As engineering systems become increasingly complex the need for automation arises [23]. A tenet of our work 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 communication (or mappings) among multiple disciplines [24] [25] [26].

The upper half of Figure 4 complements Figure 2, and pulls together the different pieces of the proposed architecture for distributed system behavior modeling with ontologies, rules, mediators and message passing mechanisms. On the left-hand side, the textual requirements are defined in terms of mathematical and logical rule expressions for design rule checking. Engineering models will correspond to a multitude of graph structure and composite hierarchy structures for the system

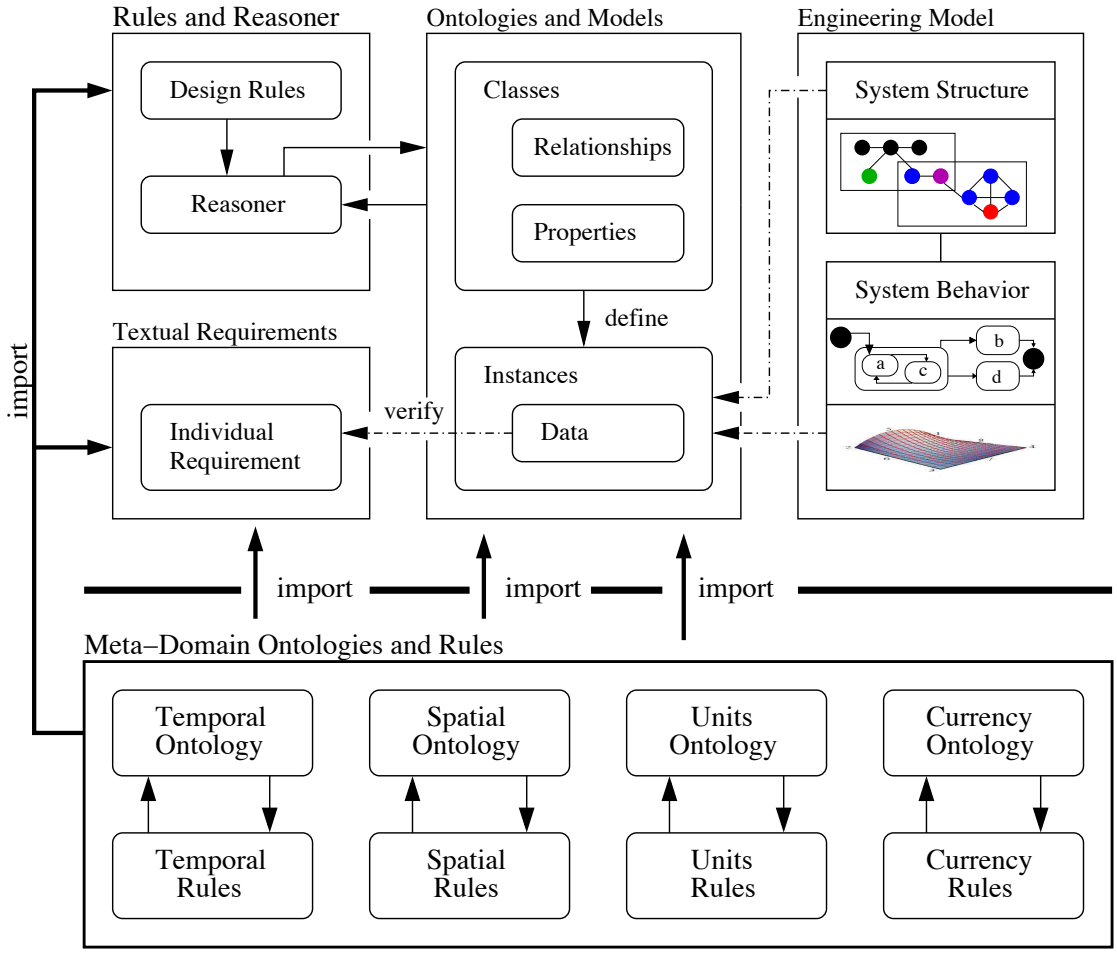

Figure 4. Framework for implementation of semantic models using ontologies, rules, and reasoning mechanisms (Adapted from Delgoshaei, Austin and Nguyen [12]).

structure and system behavior. Behaviors will be associated with components. Discrete behavior will be modeled with finite state machines. Continuous behaviors will be represented as the solution to ordinary and partial differential equations. Ontology models and rules will glue the requirements to the engineering models and provide a platform for simulating the development of system structures, adjustments to system structure over time, and system behavior. In a typical application, collections of ontologies and rules will be developed for the various domains (see, for example, Figures 1 and 2) that participate in the system structure and system behavior models.

The use of Semantic Web technologies for rule checking has several key benefits [27], [28]: (1) Rules that represent policies are easily communicated and understood, (2) Rules retain a higher level of independence than logic embedded in systems, (3) Rules separate knowledge from its implementation logic, and (4) Rules can be changed without changing source code or the underlying model. A rule-based approach to problem solving is particularly beneficial when the application logic is dynamic (i.e., where a change in a policy needs to be immediately reflected throughout the application) and rules are imposed on the system by external entities. Rules can be developed to resolve situations of conflict and/or competing objectives – such strategies use notions of fairness to prevent deadlocks in the system operation. All three of these conditions apply to the design and management of urban systems.

B. Working with Jena and Jena Rules

Our experimental software prototypes employ Apache Jena and Jena Rules. Apache Jena [29] is an open source Java framework for building Semantic Web and linked data applications. Jena provides APIs (application programming interfaces) for developing code that handles RDF (resource description framework), RDFS, OWL (web ontology language) and SPARQL (support for query of RDF graphs). The Jena rule-based inference subsystem is designed to allow a range of inference engines or reasoners to be plugged into Jena. Jena Rules is one such engine.

Jena Rules employs facts and assertions described in OWL to infer additional facts from instance data and class descriptions. As we will soon see in the case study example, domain-specific ontologies can import and use multi-domain (or cross-cutting) ontologies, rules can be distributed among domains (which is at odds with ideas within the Semantic Web community that ontologies should be tightly coupled to ontologies), and rules can be written to respond to events that involve (or affect) reasoning among multiple domains. Such

inferences result in event-driven structural transformations to the semantic graph model.

Jena also provides support for the development of builtin functions that can link to external software programs and streams of data sensed in the real world, thereby extending its reasoning capability beyond what is possible with the basic data types provided in OWL.

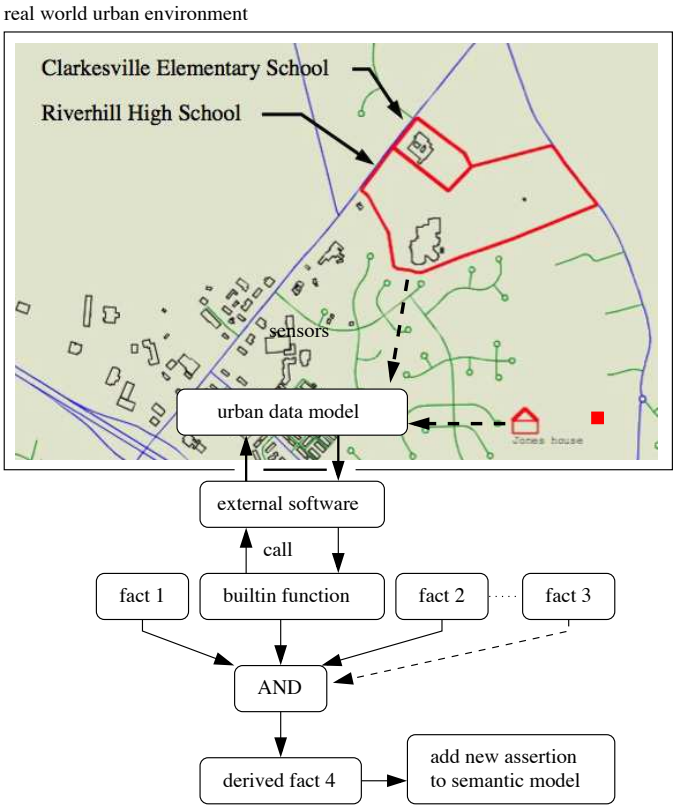

Figure 5. Framework for forward chaining of facts and results of builtin functions to new assertions (derived facts).

Figure 5 shows, for example, the essential details for forward and backward chaining driven by data collected from an urban setting. To combat the lack of support for complex data types, such as those needed to represent data for spatial and temporal reasoning, we adopt a strategy of embedding the relevant data in character strings, and then designing builtin functions and external software that can parse the data into spatial/temporal models, and then make the reasoning computations that are required.

C. Data-Driven Generation of Semantic Models

In order to build the semantic models presented in Figure 4, there needs to be a pathway from the specification of ontologies and rules to population of the semantic graphs with individuals representing various forms of urban data.

As illustrated along the left-hand side of Figure 6, the process begins with development of software for an abstract ontology model (i.e., AbstractOntologyModel). AbstractOntologyModel contains software for the domain-neutral specification and handling of ontologies and rules. Domain-specific Jena Models are an extension of the abstract model. They are

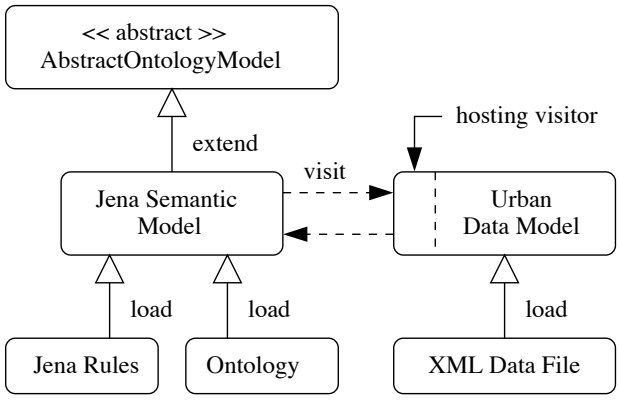

Figure 6. Data-driven approach to generation of individuals in semantic graphs.

capable of systematically assembling semantic graphs, transforming the graph structure with rules, and querying the graph structure. Next, data is imported into Java Object data models using JAXB, the XML binding for Java. After the ontologies and rules have been loaded into the Jena Semantic Model, the semantic model creates instances of the relevant OWL ontologies by visiting the urban data models and gathering information on the individuals within a particular domain. 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 Figure 6 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 individuals.

IV. Reasoning with Time and Space

Urban decision making processes are nearly always affected by notions of time and space, which have universal application across domains.

A. Reasoning with Time

Temporal logic describes how a system changes over time, and apply when we want to know not what is true, but when? For example, temporal logic allows us to determine if the schools shown in Figure 5 have an age beyond their working lifetime, and if the young residents of the house are old enough to attend the local schools.

Formal theories for reasoning with points and intervals of time are covered by Allen's temporal interval calculus [30], [31]. Notions of (calendar) time are supported as a data type in Jena. Ontologies of time can be loaded into Jena. Procedures for reasoning about points and intervals of time can be implemented in Jena Rules.

B. Reasoning with Space.

Spatial logic is concerned with regions and their connectivity, allowing one to address issues of the form: what is true, and where? Figure 5 shows, for example, the border for two schools and a house in the local neighborhood. Spatial reasoning mechanisms allows us to verify if the schools share a boundary and/or if the house is within the school zone.

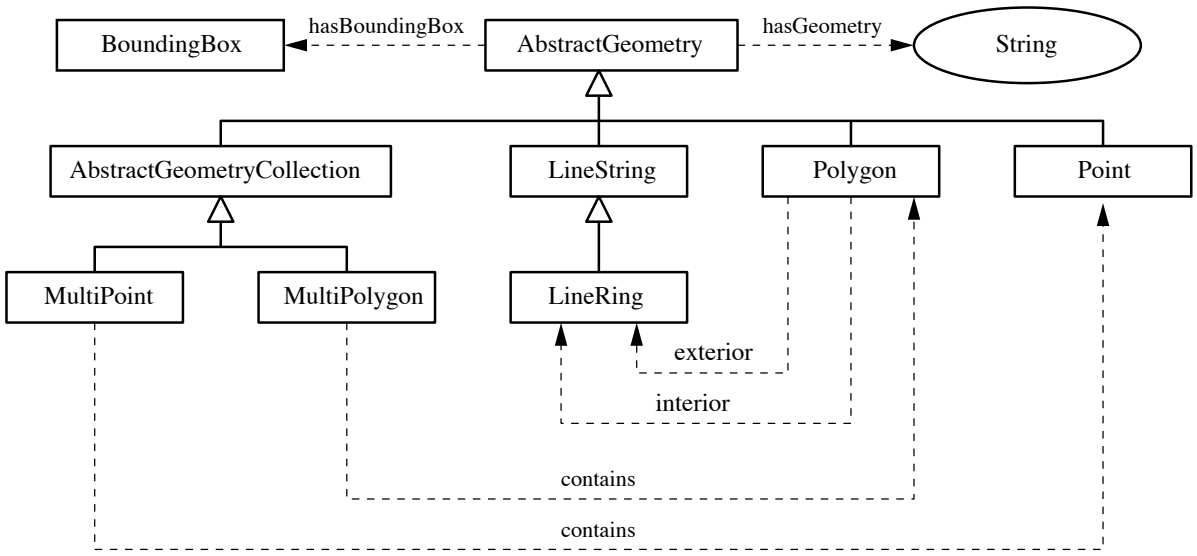

Figure 7. Abbreviated representation of spatial (geometry) ontology and associated data and object properties.

Formal theories for reasoning with space – points, lines, and regions – are covered by region connected calculus [32]. A robust implementation of two-dimensional spatial entities and associated reasoning procedures is provided by the Java Topology Suite (JTS) [33].

An important detail of implementation implied by Figure 5 is the need for backend reasoning procedures associated with JTS to operate independently of the source domains. This is achieved with the spatial (geometry) ontology and associated data and object properties shown in Figure 7. High-level classes – abstract concepts – are provided for entities that represent singular geometry (e.g., AbstractGeometry) and groups of entities (e.g., AbstractGeometryCollection). Specific types of geometry (e.g,, Polygon, MultiPoint) are organized into a hierarchy similar to the Java implementation in JTS. The high-level class AbstractGeometry contains a Datatype property, hasGeometry, which stores a string representation of the JTS geometry. For example, the abbreviated string "POLYGON ((0 0, 0 5, ... 0 0))" shows the format for pairs of (x,y) coordinates defining a two-dimensional polygon. Within Jena Rules, families of builtin functions can be developed to evaluate the geometric relationship between pairs of spatial entities (e.g., to determine whether or not a point is contained within a polygon) and return a boolean result. The latter is fact in the reasoning process shown in Figure 5.

C. Reasoning with Time and Space.

Logics for time and space can be combined allowing one to address issues of the form: We want to know when and where something will be (or has been) true? Spatio-temporal reasoning procedures in geoinformatics can be used for predictive (i.e., looking forward in time) and historical (i.e., looking back in time) purposes. For example, Figure 5 shows there are now two schools in our geographical area of interest. But what about 50 years ago – perhaps it was farmland back then?

V. Distributed System Behavior Modeling

A. Distributed System Behavior Modeling

Urban systems have decentralized system structures. 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 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, its goals and objectives. While each of the participating disciplines may have a preference toward operating their urban domain as independently as possible from the other disciplines, achieving target levels of performance and correctness of functionality nearly always requires that disciplines coordinate activities at key points in the system operation. This is especially important for the planning of relief actions in response to natural disasters.

Until very recently infrastructure management systems did not allow a manager of one system to access the operations and conditions of another system. Therefore, emergency managers would fail to recognize this interdependence of infrastructures in responding to an incident, a fact recognized by The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets [34]. In such situations, where there is no information exchange between interdependent systems, interdependencies can lead to cascading disruptions throughout the entire system in unexpected, undesirable and costly ways. The objective of this research effort is to explore opportunities for overcoming these limitations.

B. Software Architecture

Figure 8 shows the software architecture for distributed system behavior modeling for collections of graphs that have dynamic behavior defined by ontology classes, relationships among ontology classes, ontology and data properties, listeners, mediators and message passing mechanisms. The abstract

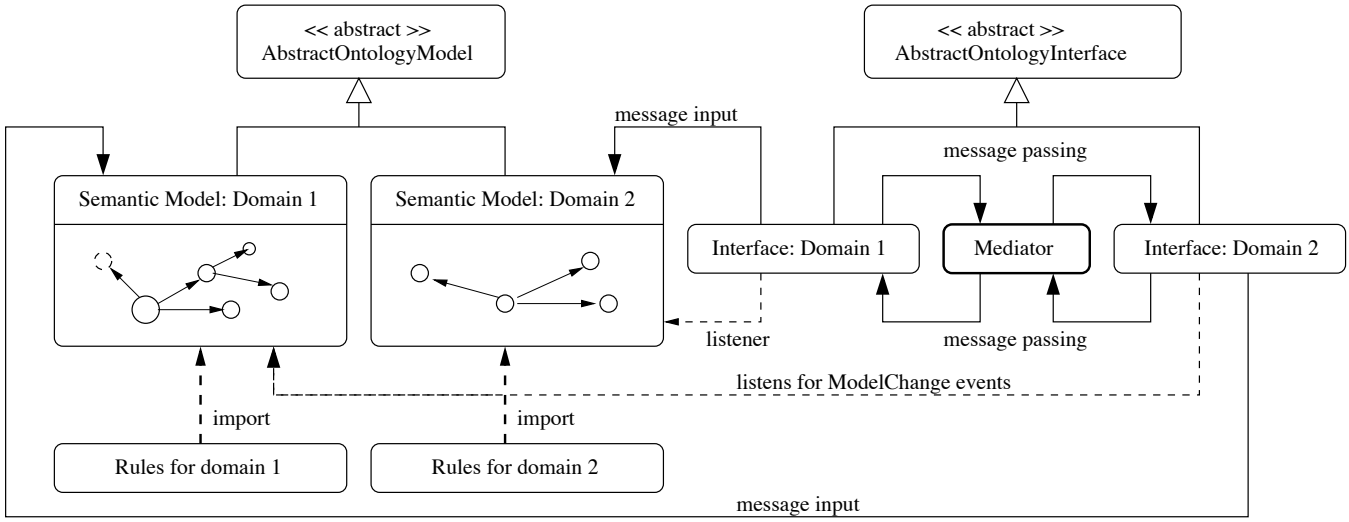

Figure 8. System architecture for distributed system behavior modeling with ontologies, rules, mediators and message passing mechanisms.

ontology model class contains concepts common to all ontologies (e.g., the ability to receive message input).

Domain-specific ontologies are extensions of the abstract ontology classes. They add a name space and build the ontology classes, relationships among classes, properties of classes for the domain. In an urban setting, individual domain ontologies may be constructed for infrastructure systems such as water, communications, oil and gas, transportation, and electric power systems shown in Figure 1. Instances (see Figure 4) are semantic objects in the domain. By themselves, the ontologies provide a framework for the representation of knowledge, but otherwise, cannot do much and really aren't that interesting. This situation changes when domain-specific rules are imported into the model and graph transformations are enabled by formal reasoning and event-based input from external sources.

C. Distributed Behavior Modeling with Ontologies and Rules

Distributed behavior modeling involves multiple semantic models, multiple sets of rules, mechanisms of communication among semantic models, and data input, possibly from multiple sources. We provide this functionality in our distributed behavior model by loosely coupling each semantic model to a semantic interface. Each semantic interface listens for changes to the semantic domain graph and when required, forwards the essential details of the change to other domains (interfaces) that have registered interest in receiving notification of such changes. They also listen for incoming messages from external semantic models. Since changes to the graph structure are triggered by events (e.g., the addition of an individual; an update to a data property value; a new association relationship among objects), a central challenge is design of the rules and ontology structure so that the interfaces will always be notified when exchanges of data and information need to occur. Individual messages are defined by their subject (e.g., report receipt confirmation), a source and a destination, and a reference to the value of the data being exchanged. The receiving interface will forward incoming messages to the semantic model, which, in turn, may trigger an update to the graph model. Since end-points of the basic message passing infrastructure are common to all semantic model interfaces, it makes sense to define it in an abstract ontology interface model.

VI. Case Study Problem

Whilst there are a number of definitions for critical national infrastructure, from a city perspective the concept of critical infrastructure is not well defined. Boyes et al. [35] proposed that criticality in a city's context addresses elements necessary for the delivery of essential services to the populace who are resident and/or work in the city and that impact is focused at city rather than national level. The critical infrastructure must encompass both the citys normal operating state, and its ability to the basic facilities, services, and installations needed for the functioning of a community or society. This includes transportation and communications systems, water and power lines, and public institutions including schools, post offices and prisons.

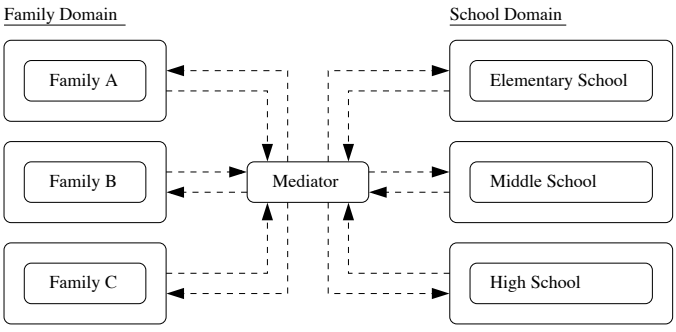

Figure 9. Framework for communication among multiple families and schools enabled by a mediator.

To illustrate the capabilities of our experimental architecture, we now present the essential details of a simulation framework for event-driven behavior modeling of a critical urban system: education. In this case study set up, a multiplicity of families interact with schools embedded in an urban environment. Interactions among groups of families and

schools is governed by ontologies, rules, and exchange of information as messages, which pass through and are managed by a mediator (see Figure 9). The decision making framework includes reasoning with spatial attributes of families and schools, and time-driven events.

A. Scenario for Family-School System Behavior Modeling

We now illustrate the capabilities of the proposed modeling abstractions by working step by step through the following scenario of interactions between families and the school system: (1) Determine eligibility for enrollment, (2) Complete enrollment form, (3) Receive enrollment confirmation, (4) Report period starts, (5) Send reports home, and (6) Receive parent signature. Evaluation of Step 1 involves combinations of spatial and temporal reasoning. Steps 2 through 6 focus on the exchange and processing of message among the participating urban domains.

Figure 10 is a detailed view of the connectivity relationships and flows of data/information in the family-school case study scenarios. The enrollment process involves an exchange of data from a family to the corresponding school in which the child should enroll. Then, and some point later in time, the school system sends a school report home.

B. Framework for Family-School-Urban Interactions

We begin by abstracting the urban components of the problem from consideration, and simply focus on the model for family school interactions.

Figure 11 shows a schematic of the schools in the Columbia-Clarksville Area (shown on left) and fictitious school zone boundaries (shown on right-hand side). As every parent knows, the enrollment process involves the exchange of specific information between schools and families. The school system only allows enrollment of students who meet the age requirements, and live within the school zone jurisdiction. Once the child is accepted the school system takes over. They decide when school reports will be sent home, and if the child is entitled to school bus service. Some of these determinations are done by comparing spatial entities, such as family addresses, school addresses, and school zone boundaries. Addresses are defined by latitude and longitude coordinates; therefore, a simple calculation using the latitudes and longitudes of two addresses can determine the distance between them. Similarly, school zones are defined by a collection of latitude and longitude coordinates that compose a polygon geometric shape. Any algorithm that solves the point-in-polygon (PIP) problem can determine if the address lies within the school zone boundaries. This work uses OpenStreetMap tool to retrieve the latitudes and longitudes necessary for the these comparisons. Figure 10 is an instantiation of the concepts introduced in Figure 8 and shows the software architecture for a family-school interaction.

C. Instantiating Semantic Models with Data

In this problem setup, the information to be exchanged between ontologies is stored as key/value pairs in XML datafiles. The key (e.g. “first name”, “citizenship”, etc.) identifies, and is used to retrieve the values (e.g., “Mark”, “New Zealand”, etc.). Textual content stored in the XML datafiles is extracted and instantiated as class instances in the data model. Our prototype implementation employs JAXB technology for the creation of data models as shown in Figure 12. We then systematically visit each element of the data model (the code is implemented as a visitor software design pattern) and create instances of the ontology classes. The latter are called Individuals), and they are laden with the data from XML files.

D. Family and School System Ontologies

Our application employs OWL to define ontologies as collections of classes, data and object properties, and the relationships among them.

Figure 13 shows the relationship between classes in the family ontology. Male, Female, Child and Student are subclasses of class Person. The class Boy is a subclass of class Male. The class Person has properties that get inherited by all subclasses such as hasAge, hasWeight, hasBirthdate, hasFamilyName, has FirstName, hasSocialSecurityNo, hasCitizenship. The class Student has properties associated with school enrollment, such as livesInSchoolZoneOf, attendsPreschool, attendsSchool, attend sElementarySchool, attendsMiddleSchool, attendsHighSchool, and hasReportFrom. The class family has property hasFamilyName, and the class Address has proper ties hasLatitude and hasLongitude. Other properties such as hasFamilyMember, belongsToFamily, hasFather, hasSon, hasDaughter, and hasAddress define relation ships that hold between objects.

In the same fashion, an ontology can be constructed for the school system. Figure 14 shows the relationship between classes in a school ontology. Elementary School, Middle School and High School are subclasses of School. Grades 1 through 12 are subclasses of Grade. A school has properties that get inherited by all school subclasses such as hasName. A grade also has properties that get inherited by all grade subclasses such as hasEnrollment. A student has properties similar to the ones dened in the classes Person and Student in the family ontology such as hasFirstName, hasFamilyName, hasBirthDate, hasAge, hasSocialSecurityNo, attendsElemntarySchool, attendsMiddleSchool, attendsHighSchool, and hasReport. In addition, it also has properties such as eligibleForSchoolBus and willArriveLate. The class Address also follows the same pattern of the family ontology, with properties hasLatitude and hasLongitude. The classes Calendar and Event are included in this ontology to provide temporal behavior modeling capabilities. The class Event has properties hasStartTime and hasEndTime. The class Bus has property hasArrivalTime. Other properties such as hasGrade, hasStudent, isInGrade, hasStudentAddress, hasSchoolAddress, hasBus, livesInSchoolZoneOf and hasEvent define relationships that can hold between objects.

E. Family and School System Rules

By themselves ontologies cannot model the dynamic evolution of objects, properties and relationships. Consider the family ontology, some of the data remains constant over time (e.g., birthdates), while other data is dynamic (e.g., attending preschool). However, when coupled with a set of domain-specific rules, ontological representations enable graph transformations. In our application, we use Jena Rules to define

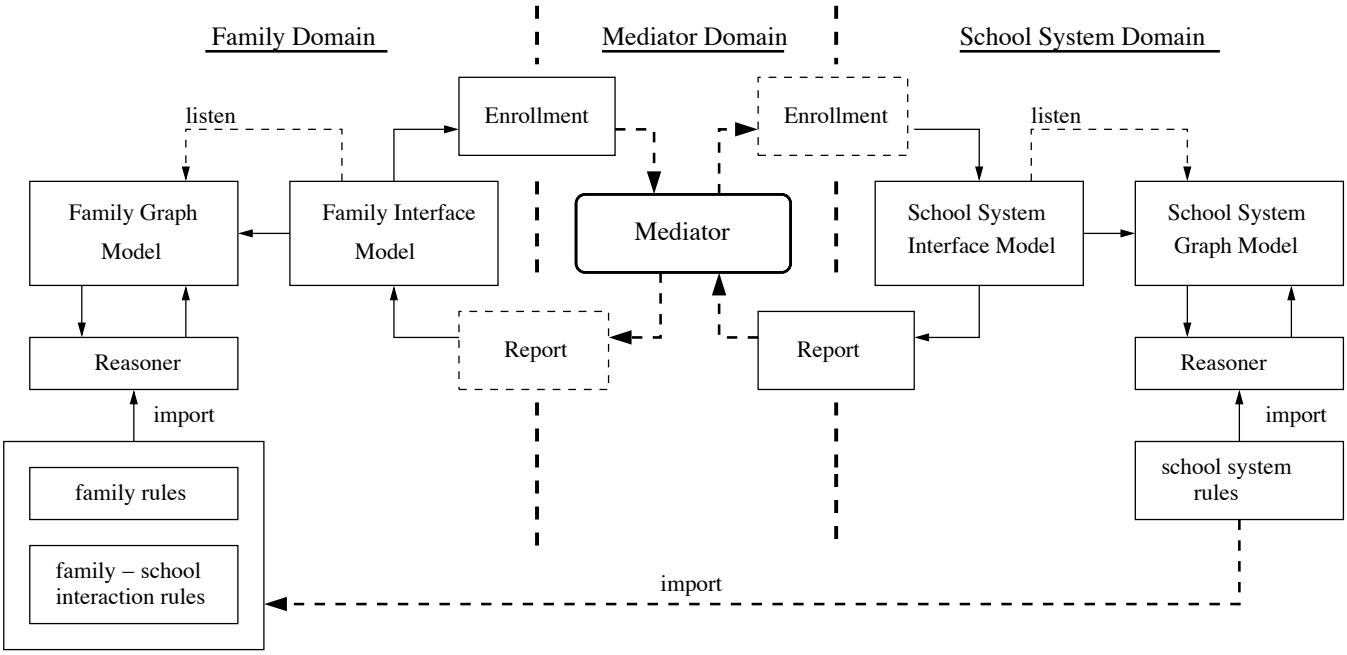

Figure 10. Software architecture for distributed behavior modeling in the family-school case study.

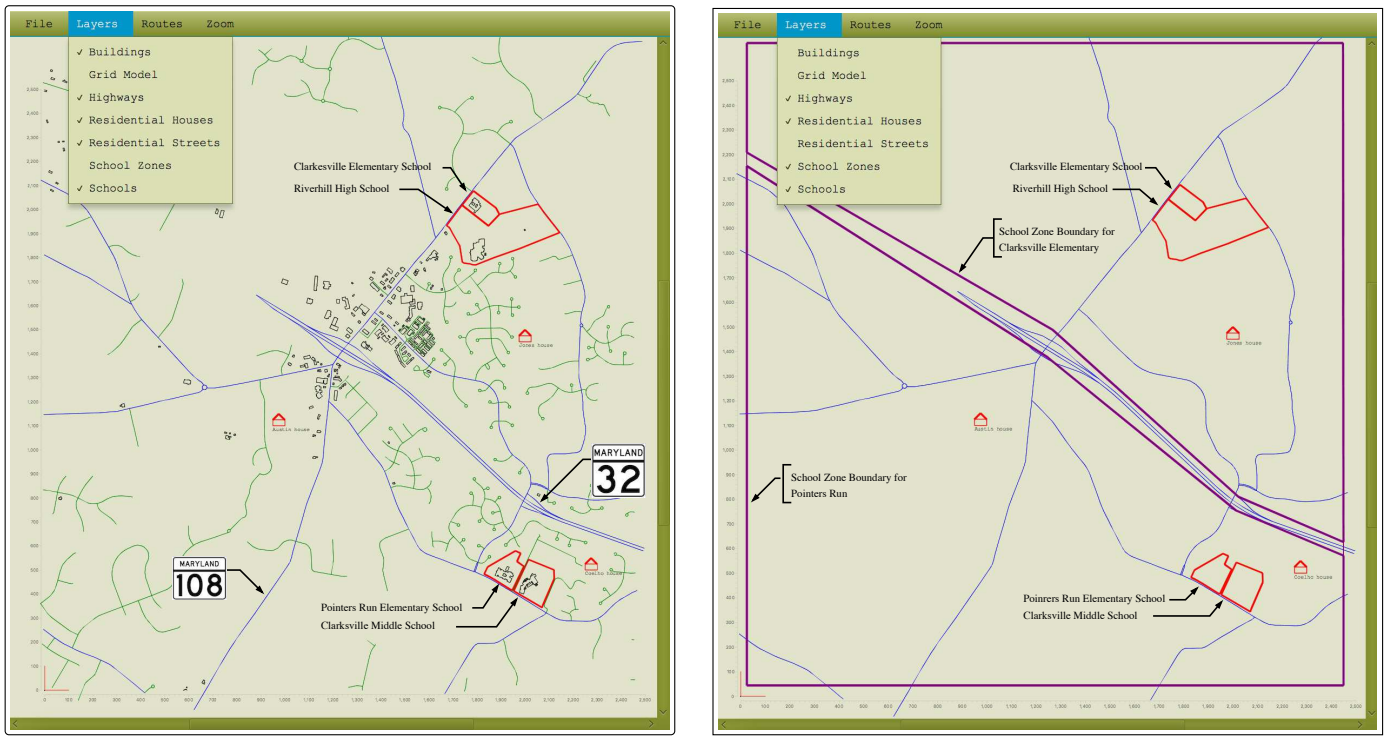

Figure 11. Graphical interface for behavior modeling of family-school-urban geography system dynamics. The school and school zones correspond to the Columbia-Clarksville Area, Maryland, USA.

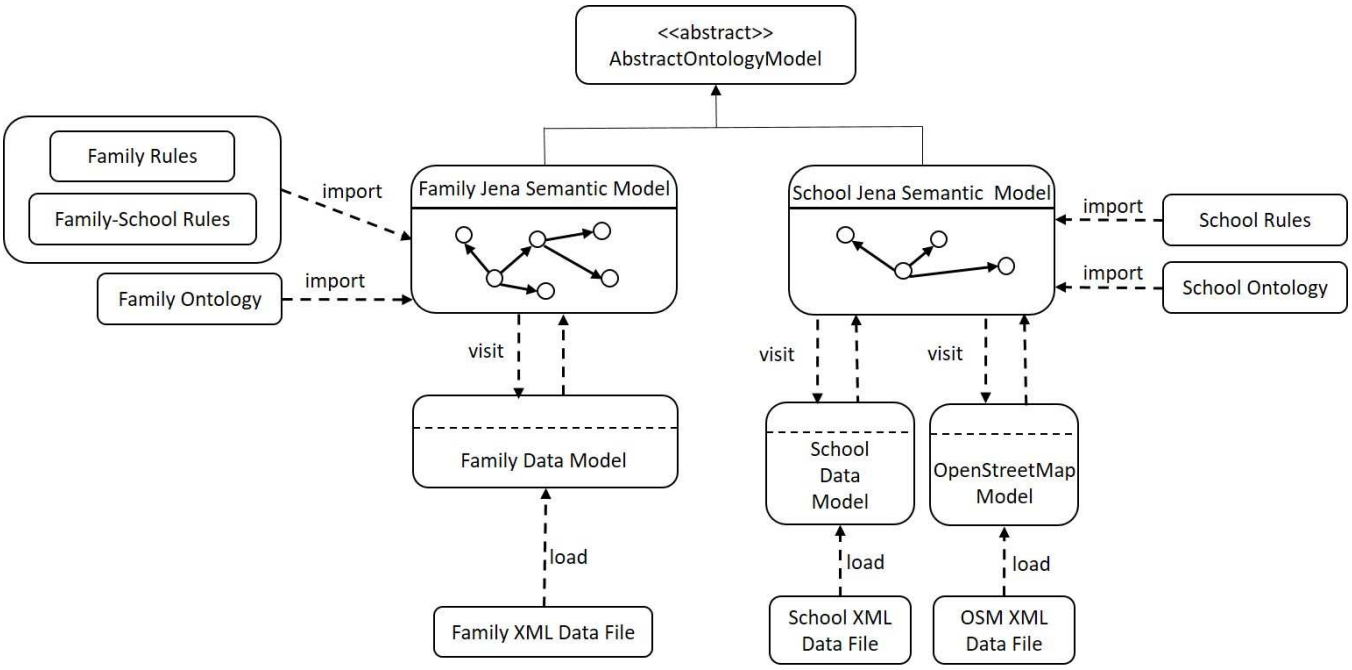

Figure 12. Generation of family and school semantic models, with input from the family data file, the school system data file, and data from OpenStreetMap.

domain-specific rules.

Figure 15 contains an abbreviated list of Jena rules for identifying relationships and properties within a family semantic model. The combination of ontologies and ontology rules is extremely powerful in scenarios where ontology graphs are dynamic. Suppose, for example, that a boy Sam was born December 10, 2007. Given a birthdate and the current year, a built-in function getAge() computes Sam's age. An age rule defined using Jena Rules determines whether or not a person is also a child. Therefore, the behavior modeling for the family system is defined by the set of rules governing graph transformations. Graph transformation can occur due to input (e.g., family graph changes because a new child is born) or time (e.g., the family graph changes because a specific member is no longer a child).

Figure 16 contains an abbreviated list of Jena rules for event-driven transformation of the School Semantic Model. Rules are provides for attendance, progression through the grades, timing of school reports, eligibility for transportation services and event induced alerts. Transformations in the semantic graph structure can also be induced by a variety of temporal and spatial factors. From a family perspective, individuals such as Sam are modeled as instances of the classes Boy, Male and Child. From a school perspective, Sam is eligible to become a student when he is between the ages of 5 and 18, and his family lives within the defined school zone. School reporting periods are events defined by intervals of time on an academic calendar. When a built-in function getToday() determines that the current time falls within one of the "reporting intervals" school reports are sent home. Similarly, the built-in function getDistance() computes the the distance between Sam's home address and the school address, and a rule determines whether or not he is eligible for school bus service. Each of these entities triggers a change in the school semantic graph.

F. Rules for Family-School System Interaction

So far the family and school rule systems have been completely decoupled and one might think that they operate independently. In reality, a small set of rules that govern family behavior are defined by the school system and distributed to individual families in the family system. As illustrated in Figure 17, rules for family-school system interaction define the grades that are appropriate for each age and the schools (e.g., elementary, middle, high) that will be attended. In practice, the family-school interaction rules are loaded into the family system alongside the regular family system rules. The former will inform Sam's family when he is now old enough to attend regular school by triggering a change to the family graph. This change, in turn, will trigger the school enrollment process for Sam to start preschool.

Family-school system interactions are also affected by spatial concerns. In particular, a child can only enroll in a particular school if he/she has a home address the lies within its school zone. From a geometric standpoint (see Figure 7), this test is equivalent to verifying that the home address (a geographic point) is contained within the school zone (a geographic polygon). JTS can easily handle this computation. In practice, however, resolving this issue is complicated by the fact that the home address and school zone are contained in different models. Thus, a strategy is needed whereby a family can query the school system for details on the school zone and do the point-in-polygon computation on the family model, or, the child's address is part of the enrollment package and the school verifies spatial eligibility on the school system side. In either case, a simple Jena rule can retrieve details of the point and polygon in a string format – see the top right-hand side of Figure 7 – and a Jena built-in function working with JTS

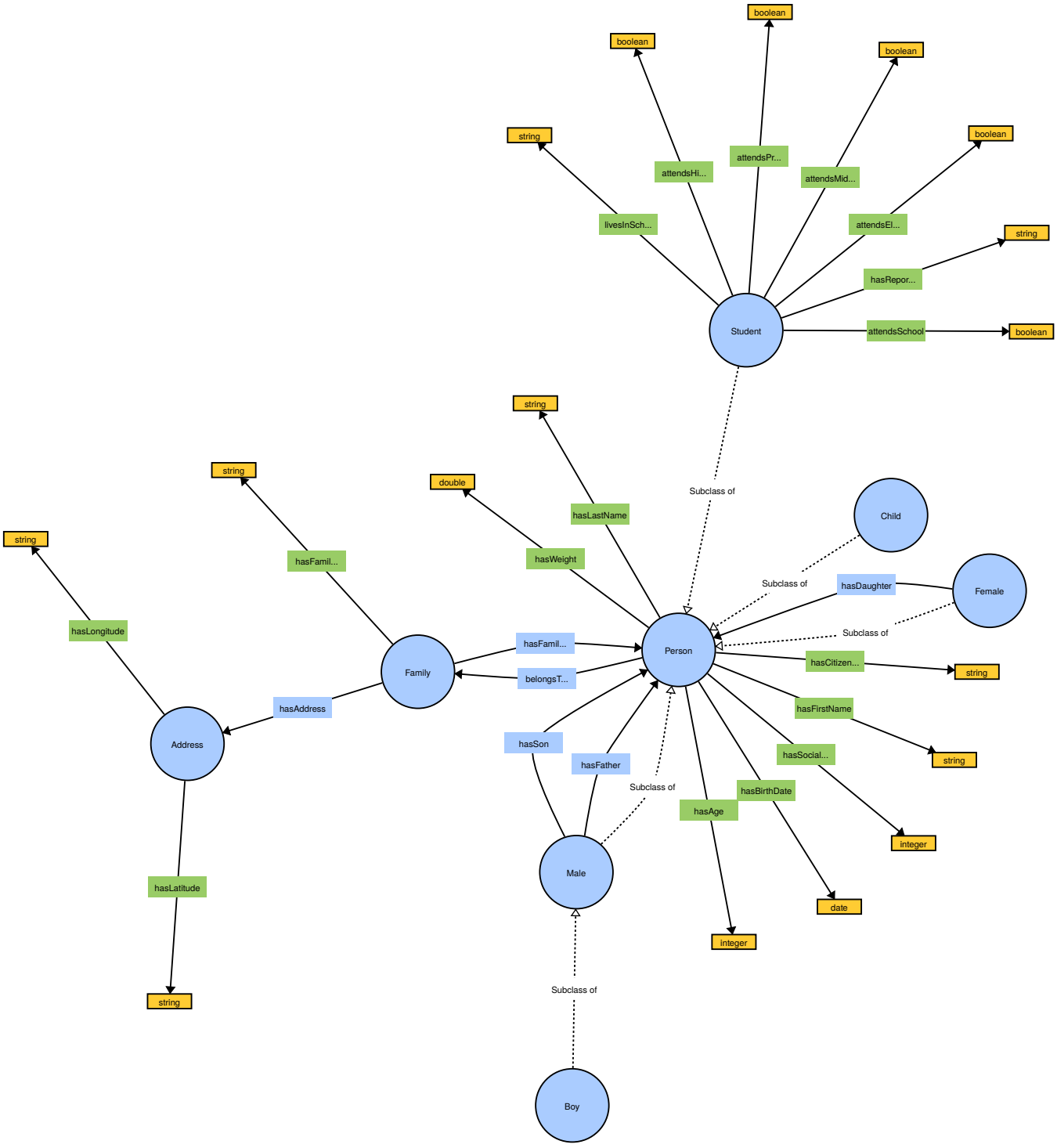

Figure 13. Family ontology diagram with classes, properties, and relationships among classes and properties.

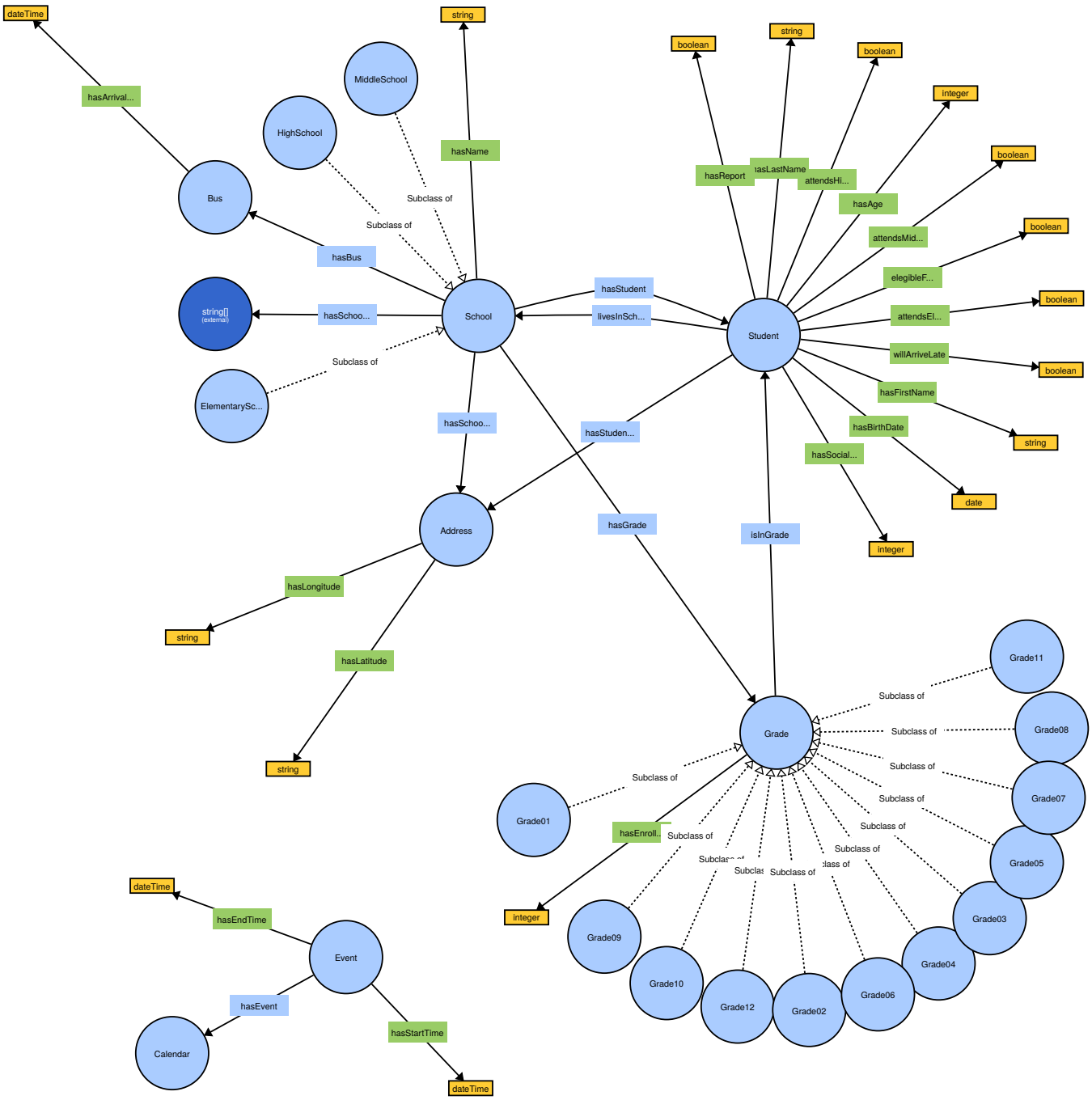

Figure 14. School system ontology diagram with classes, properties, and relationships among classes and properties.

can evaluate the point-in-polygon containment.

G. Mediator Design

When the number of participating applications domains is very small, point-to-point channel communication between interfaces is practical. Otherwise, an efficient way of handling domain communication is by delegating the task of sending and receiving specific requests to a central object. In software engineering, a common pattern used to solve this problem is the Mediator Pattern.

As illustrated in Figures 2 and 3, the mediator pattern defines a object responsible for the overall communication of the system, which from here on out will be referred as the mediator. The mediator has the role of a router, it centralizes the logic to send and receive messages. Components of the system send messages to the mediator rather than to the other components; likewise, they rely on the mediator to send change notifications to them [36]. The implementation of this pattern greatly simplifies the other classes in the system; components are more generic since they no longer have to contain logic to manage communication with other components. Because other

```
@prefix af:  <http://www.isr.umd.edu/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Propagate class hierarchy relationships

[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y),(?a rdf:type ?x) -> (?a rdf:type ?y) ]

// Rule 02: Family rules

[ Family: (?x rdf:type af:Family) (?x af:hasFamilyMember ?y) -> (?y af:belongsToFamily ?x) ]

// Rule 03: Identify a person who is also a child

[ Child: (?x rdf:type af:Person) (?x af:hasAge ?y) lessThan(?y, 18) -> (?x rdf:type af:Child) ]
[ UpdateChild: (?x rdf:type af:Child) (?x af:hasBirthDate ?y) getAge(?y,?b) ge(?b, 18) -> remove(0) ]

// Rule 04: Identify a person who is also a student

... Student rules removed ...

// Rule 05: Compute and store the age of a person

[ GetAge: (?x rdf:type af:Person) (?x af:hasBirthDate ?y) getAge(?y,?z) -> (?x af:hasAge ?z) ]

[ UpdateAge: (?a rdf:type af:Person) (?a af:hasBirthDate ?b) (?a af:hasAge ?c)
  getAge(?b,?d) notEqual(?c, ?d) -> remove(2) (?a af:hasAge ?d) ]

// Rule 05: Set father-son and father-daughter relationships

[ SetFather01: (?f rdf:type af:Male) (?f af:hasSon ?s)-> (?s af:hasFather ?f)]
[ SetFather02: (?f rdf:type af:Male) (?f af:hasDaughter ?s)-> (?s af:hasFather ?f)]
```

Figure 15. Abbreviated list of Jena rules for transformation of the Family Semantic Model.

components remain generic, the mediator has to be application specific in order to encapsulate application-specific behavior. One can reuse all other classes for other applications, and only need to rewrite the mediator class for the new application.

H. Working with Apache Camel

Looking to the future, we envision a full-scale implementation of distributed behavior modeling (see Figure 1) having to transmit a multiplicity of message types and content, with the underlying logic needed to deliver messages possibly being a lot more complicated than send message A in domain B to domain C. In our preliminary work [1] the mediator capability was simplified in the sense that domain interfaces were assumed to be homogeneous. But looking forward, this will not always be true. Cities are transitioning from an industrial- to information-age fabric, where highly efficient communication networks are employed to minimize the importance of time constraints and relieve the need for urban congestion. Information and Communication Technologies (ICT) have become a significant part of information-age cities. ICT can be found at many levels, ranging from the collection of data from ordinary daily tasks (e.g. traffic monitoring), to informing managerial tasks that involve decision-making based on the monitored data (e.g. electricity and water management; education and health; climate change monitoring) [37]. Typically, each of the smart systems and sensors has specific requirements, processes and outputs. The flow and variety of urban data captured by these smart systems and sensors is only going to grow and diversify in years to come. This situation points to a strong need for new approaches to the construction and operation of message passing mechanisms.

One promising approach that we will explore in this work is Apache Camel [8] [9], an open source Java framework that focuses on making Enterprise Integration Patterns (EIP) accessible through carefully designed interfaces, base objects, commonly needed implementations, debugging tools and a configuration system. It joins together messaging start and end points, allowing for the transferring of messages from different sources to different destinations. Figure 18 shows, for example, a platform infrastructure for behavior modeling of three connected application (networked) domains. In addition to basic content-based routing, Apache Camel provides support for filtering and transformation of messages. The latter is an essential feature to future cities, where heterogeneous domain interfaces will need to produce and consume messages that are not always in the same language or format.

A project developed in 2015 by Abdellatif Bouchama has successfully implemented Apache Camel for data transfer in an urban scenario. The project demonstrates how to improve urban air quality by gathering real time data from cities in France, and adding value to it by using Apache Camel to process the data and notifying users of the system [38]. Apache Camel can also be congured to receive data from Twitter, Facebook, Open Weather Map and many other web environments [39] of interest to an urban model. A study

```
@prefix af:  <http://www.isr.umd.edu/school#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Propagate class hierarchy relationships

... Class hierarchy rules removed ...

// Rules 02: Elementary school rules

[ EnterElementarySchool: (?x rdf:type af:Student) (?y rdf:type af:ElementarySchool)
     (?x af:hasBirthDate ?a) getAge(?a,?b) ge(?b, 6) le(?b, 10) ->
     (?x af:attendsElementarySchool af:True) (?y af:hasStudent ?x)]

[ LeaveElementarySchool: (?x rdf:type af:Student) (?x af:hasBirthDate ?a)
     (?x af:attendsElementarySchool af:True) (?y af:hasStudent ?x)
     getAge(?a,?b) ge(?b, 10) -> remove(2) ]

[ GradeOne: (?x rdf:type af:Student) (?x af:hasBirthDate ?a)
            getAge(?a,?b) equal(?b, 6) -> (?x af:isInGrade af:Grade01) ]

... Rules for Grades 2 through 5 removed ...

// Rules 05: If today is report period, send school report

[ GenerateReport: (?x rdf:type af:Event) (?y rdf:type af:Student) (?z rdf:type af:School)
      (?z af:hasStudent ?y) (?x af:hasStartTime ?t1) (?x af:hasEndTime ?t2) getToday(?t3)
      lessThan(?t3,?t2) greaterThan(?t3,?t1) -> (?y af:hasReport af:True)  ]

// Rules 06: School transporation service rules

[ ESTransportationService: (?x rdf:type af:Student) (?y rdf:type af:ElementarySchool)
      (?y af:hasStudent ?x) (?x af:hasStudentAddress ?k) (?y af:hasSchoolAddress ?z)
      (?k af:hasLatitude ?l1) (?k af:hasLongitude ?l2) (?z af:hasLatitude ?l3) (?z af:hasLongitude ?l4)
      getDistance(?l1,?l2,?l3,?l4,?d) greaterThan(?d,1000) -> (?x af:isElegibleForSchoolBus af:True)  ]

// Rules 07: If bus is late, send alert to parents

[ DelayAlert: (?x rdf:type af:School)(?y rdf:type af:Bus)(?z rdf:type af:Student) (?x af:hasBus ?y)
              (?y af:hasArrivalTime ?t) greaterThan(?t,"2020-09-20T03:00:00"^^xsd:dateTime)
              (?x af:hasStudent ?z) (?z af:isElegibleForSchoolBus af:True) -> (?z af:willArriveLate af:True) ]
```

Figure 16. Abbreviated list of Jena rules for transformation of the School Semantic Model. Middle and high school rules for grade assignment and use of transportation services are not shown.

performed in 2017 by Oliveira et al., investigated the use of an intelligent middleware, containing Apache Camel, to support data capture and analysis techniques to inform urban planning and design. Results were reported from a "Living Campus" experiment at the University of Melbourne, Australia, focused on a public learning space case study. Local perspectives, collected via crowd sourcing, are combined with distributed and heterogeneous environmental sensor data [37].

I. Extension 1: Using Apache Camel as a Mediator

In the first extension, communication among the family and school communities is handled by a mediator built using Apache Camel. Figure 9 is the network setup for three families interacting with elementary, middle and high schools. Every component of the system (i.e., families and schools) register in a JDNI Registry as bean components. Once a family member reaches a certain age, the age rules associated with the family system will trigger a school enrollment form to be sent to the mediator in the form of an XML file, with source, subject and destination attributes. The mediator logic routes the message according to its content, more specifically the destination attribute value and sends it to the matching bean in the registry. Similarly, once the system calendar reaches a certain date, the reporting rules associated with the school system will trigger a school report to be sent to the mediator. The messaging design allows the school enrollment form to be received only by the school of interest, and not broadcasted to the entire school system. Likewise, this design allows the school reports to be sent only to the student's family. This mediator logic design is known as point-to-point channel, and it ensures that only one listener consumes any given message. The channel can have multiple listeners that consume multiple messages concurrently, but the design ensures that only one of them can successfully consume a particular message. Using this approach, listeners do not have to coordinate with each other; coordination could be complex, create a lot of communication overhead, and increase coupling between otherwise independent receivers.

J. Extension 2: Failure Simulation

The second case study extension examines computational support for simulating failures in the distributed system oper-

```
@prefix af:  <http://www.isr.umd.edu/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rules 01: Children of age 4 and 5 attend preschool

[ EnterPreSchool: (?x rdf:type af:Student) (?x af:hasBirthDate ?a) getAge(?a,?b) ge(?b, 4)
   le(?b, 5) -> (?x af:attendsPreSchool af:True) ]

[ LeavePreSchool: (?x rdf:type af:Student) (?x af:hasBirthDate ?a) (?x af:attendsPreSchool af:True)
   getAge(?a,?b) ge(?b, 6) -> remove(2) ]

// Rules 02: Children aged 6 through 10 attend elementary school

... Rules for attending Elementary school removed ...

// Rules 03: Children aged 11 through 13 attend middle school ....

... Rules for attending Middle school removed ...

// Rules 04: Children aged 14 through 17 attend high school ....

... Rules for attending High school removed ...

// Rules 05: Children aged 6 through 18 attend regular school ....

... Rules for attending school removed ...
```

Figure 17. Jena rules for family-school system interactions at the preschool level. Rules for interactions among elementary, middle, and high schools and families are not shown.

ation. As already noted in Section I, complex urban systems always run on degraded mode, which means at some point failure and loss of urban system functionality is an inevitable fact. A resilient urban system recovers quickly and continues operating. In order to show how the architecture proposed by this work can contribute to a resilient complex system design, we introduce failure within the family and schools interaction simulation. The school rules defines which students are eligible for school bus service (a spatial decision), and by what time such students should be delivered back to their parents after school (a temporal schedule). Now imagine that a school bus is running late. The boolean property willArriveLate will be set to True. The school's semantic model interface will identify the corresponding update to the semantic graph, and in response, send an alert to the families of students in the late bus in the form of a message. The mediator will match the message destination, with each of the families' semantic model interface and forward the message. The family semantic model interface will identify the message type (i.e., late bus alert), and could potentially trigger changes to the semantic model graph to accommodate their own schedule. While this urban scenario seems urealistically simple, it captures the essense of safety and security concerns facing young urban residents. If communication among the participating parties is not handled properly and in a timely manner, uncertainties in situational awareness can easily trigger the involvement of other related systems, such as the police department.

VII. Discussion

Our vision for future (more advanced) uses of Apache Camel in behavior modeling of urban environments is focused on its ability to integrate interfaces from multiple disciplines that may not speak and understand the same language. Today, Civil Engineers are faced with the challenge of designing systems that transmit and consume a multiplicity of message types and content. Looking into the future, this challenge will be aggravated by the growth of ICT presence in urban settings. Apache Camel avoids vulnerabilities introduced by the growing flow and variety of urban data being transmitted, and allows for more resilient message passing mechanisms in urban scenarios.

VIII. Conclusions and Future Work

This paper has focused on the design and preliminary implementation of a message passing infrastructure needed to support communication in many-to-many association relationships connecting domain-specific networks.

Our long-term research objective is computational support for the design, simulation, and validation of models of distributed behavior in real-world urban environments. The family-school distributed behavior model is merely a starting point. We anticipate that the end-result will look something like Figure 2, and provide strategies for real-time control of behaviors, assessment of domain resilience, and planning of recovery actions in response to severe events. Models of urban data and system state will be coupled to tools for spatial and temporal reasoning, and will synchronize with layers of domain-specific visualization (not shown in Figure 2). In order to drive the design and validation of domain rules, and rules for exchange of messages between domains, we will design and simulate a series of progressively complicated urban case study problems.

Our future work will investigate opportunities for linking

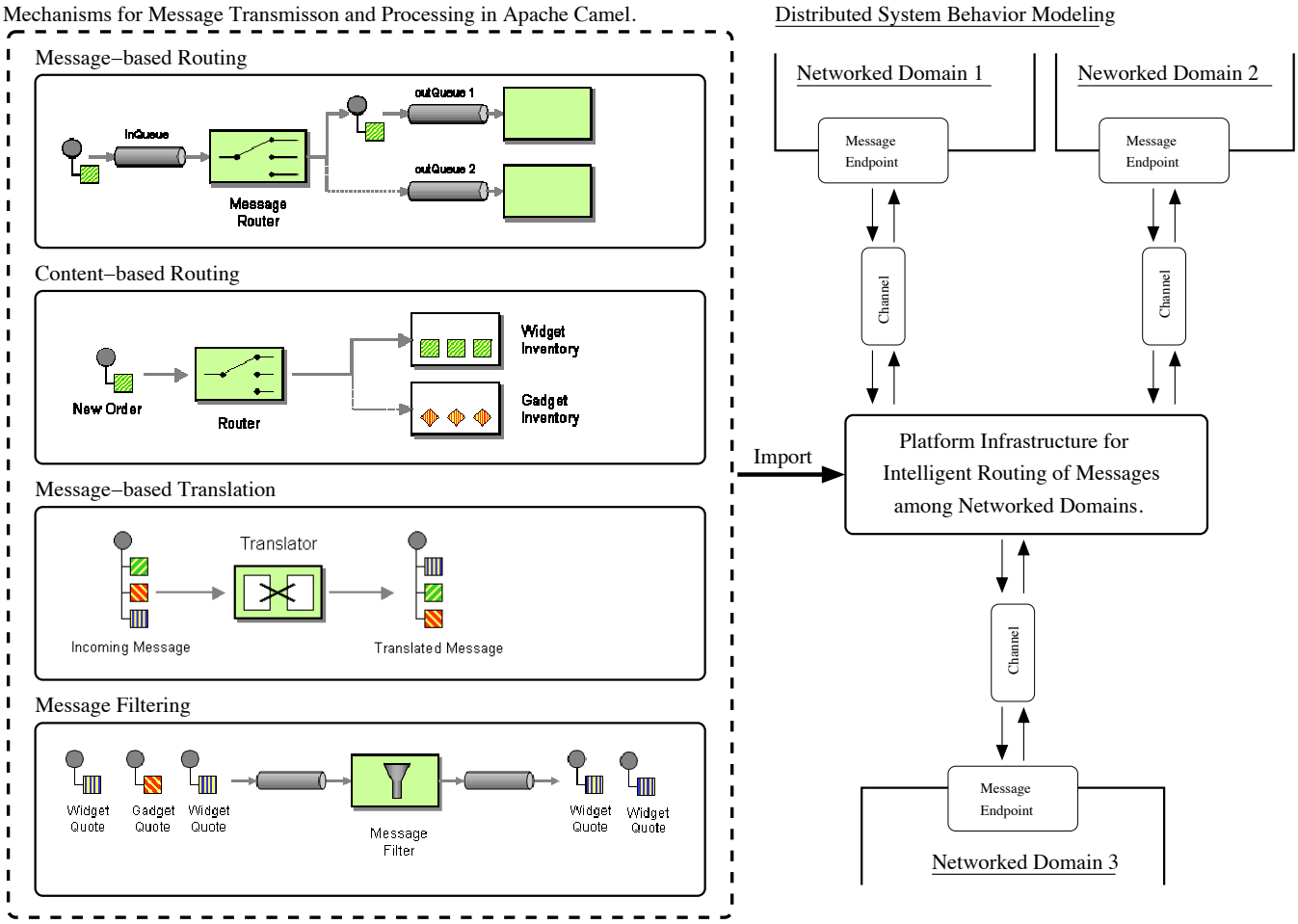

Figure 18. Platform infrastructure for distributed behavior modeling and intelligent communication (message passing) among networked domains.

of our simulation framework to tools for optimization and trade-off analysis. Such tools would allow decision makers to examine the sensitivity of design outcomes to parameter choices, understand the impact of resource constraints, understand system stability in the presence of fluctuations to modeling parameter values, and potentially, even understand emergent interactions among systems.

Lastly, a potential extension to the presented work, is in the development of ontologies. As it is presented in this work, the construction of ontologies is based on the data available from the XML datafiles, but this process is done manually. When modeling complex urban systems, this approach may become troublesome. A necessary step forward would be to implement Natural Language Processing (NLP) for the semi-automated identification of knowledge provided by the datafiles.

REFERENCES

[1] M. Coelho, M.A. Austin, and M. Blackburn, "Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Many-to-Many Association Relationships," The Twelth International Conference on Systems (ICONS 2017), April 23-27 2017, pp. 10–15.

[2] M. A. Austin, P. Delgoshaei, and A. Nguyen, "Distributed Systems Behavior Modeling with Ontologies, Rules, and Message Passing Mechanisms," in Thirteenth Annual Conference on Systems Engineering Research (CSER 2015), Hoboken, New Jersey, March 17-19 2015, pp. 373–382.

[3] S.M. Rinaldi, J.M. Peerenboom, and T.K. Kelly, "Identifying, Understanding, and Analyzing Critical Infrastructure Interdependencies," IEEE Control Systems Magazine, vol. 21, December 2001, pp. 11–25.

[4] S. Selberg, and M.A. Austin, "Toward an Evolutionary System of Systems Architecture," in 18th Annual International Symposium of The International Council on Systems Engineering (INCOSE 2008), Utrecht, The Netherlands, July 15-19 2008.

[5] R. I. Cook, "How Complex Systems Fail." Cognitive Technologies Laboratory, University of Chicago, Chicago IL., 1998.

[6] J. Gao, X. Liu, D. Li, and S. Havlin, "Recent Progress on the Resilience of Complex Networks," Energies, vol. 8, 2015, pp. 12 187–12 210.

[7] OptaPlanner (2016), A Constraint-Satisfaction Solver. For details, see: https://www.optaplanner.org (Accessed, Jan 4., 2017).

[8] C. Ibsen, J. Antsey, and Z. Hadrian, Camel in Action. Manning Publications Company, 2010.

[9] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building and Deploying Message Passing Solutions. Addison Wesley, 2004.

[10] T. Berners-Lee, J. Hendler, and O. Lassa, "The Semantic Web," Scientific American, May 2001, pp. 35–43.

[11] P. Delgoshaei, M. A. Austin, and D. A. Veronica, "A Semantic Platform Infrastructure for Requirements Traceability and System Assessment," The Ninth International Conference on Systems (ICONS 2014), February 2014, pp. 215–219.

[12] P. Delgoshaei, M. A. Austin, and A. Pertzborn, "*A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems*," in International Journal On Advances in Systems and Measurements, Vol. 7, No. 3-4, December, 2014, pp. 223–238., 2014.

[13] C.A. Myers, T. Slack, and J. Singelmann, "Social Vulnerability and Migration in the Wake of Disaster: The case of Hurricanes Katrina and Rita," Population and Environment, vol. 29, 2008, pp. 271–291.

[14] R. Zimmerman and C. E. Restrepo, "Analyzing Cascading Effects within Infrastructure Sectors for Consequence Reduction." 2009 IEEE International Conference on Technologies for Homeland Security, HST 2009, Waltham, MA. , 2009.

[15] Association of Bay Area Governments (ABAG), "Water System and Disasters." 2009-2010 Update of the ABAG-Led Multi-Jurisdictional Local Hazard Mitigation Plan for the San Francisco Bay Area, 2009.

[16] M. Hogan, "Anytown: Final Report." London Resilience Team, London, England, 2013.

[17] C. Robert T. Marsh, "Critical foundations: Protecting america's infrastructures - the report of the president's commission on critical infrastructure protection," Tech. Rep., October 1997. [Online]. Available: https://www.fas.org/sgp/library/pccip.pdf

[18] P. Pederson, D. Dudenhoeffer, S. Hartley, and M. Permann, "Critical infrastructure interdependency modeling: A survey of us and international research," Tech. Rep., August 2006. [Online]. Available: https://inldigitallibrary.inl.gov/sites/sti/sti/3489532.pdf

[19] H. Rahman, M. Armstrong, D. Mao, J. Marti, "I2Sim: A matrix-partition based framework for critical infrastructure interdependencies simulation," IEEE Canada Electric Power Conference, 2008, pp. 1–8.

[20] G. Falquet, C. Metral, J. Teller, and C. Tweed, Ontologies in Urban Development Projects. Springer, 2005.

[21] "OpenGIS Geography Markup Language Encoding Standard (GML). See http://www.opengeospatial.org/standards/gml (Accessed December 1, 2017)."

[22] D. Bonino, and F. Corno, DogOnt - Ontology Modeling for Intelligent Domotic Environments. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–803.

[23] M. A. Austin and J. S. Baras, An Introduction to Information-Centric Systems Engineering. Toulouse, France: Tutorial F06, INCOSE, June 2004.

[24] M. A. Austin, V. Mayank, and N. Shmunis, "Ontology-Based Validation of Connectivity Relationships in a Home Theater System," 21st International Journal of Intelligent Systems, vol. 21, no. 10, October 2006, pp. 1111–1125.

[25] ——, "PaladinRM: Graph-Based Visualization of Requirements Organized for Team-Based Design," Systems Engineering: The Journal of the International Council on Systems Engineering, vol. 9, no. 2, May 2006, pp. 129–145.

[26] N. Nassar and M. A. Austin, "Model-Based Systems Engineering Design and Trade-Off Analysis with RDF Graphs," in 11th Annual Conference on Systems Engineering Research (CSER 2013), Georgia Institute of Technology, Atlanta, GA, March 19-22 2013, pp. 216–225.

[27] Q.H. Mahmoud, "Getting started with the Java Rule Engine API (JSR 94): Toward Rule-Based Applications," Sun Microsystems, 2005, For more information, see http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html (Accessed, March 10, 2008).

[28] G. Rudolf, "Some Guidelines For Deciding Whether To Use A Rules Engine," 2003, Sandia National Labs. For more information see http://herzberg.ca.sandia.gov/guidelines.shtml (Accessed, March 10, 2008).

[29] Apache Jena:, "An Open Source Java framework for building Semantic Web and Linked Data Applications. For details, see https://jena.apache.org/," 2016.

[30] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," Communications of the ACM, vol. 26, no. 11, 1983, pp. 832–843.

[31] ——, "Towards a General Theory of Action and Time," Artificial Intelligence, vol. 23, no. 2, 1984, pp. 123–154.

[32] D.A. Randell, Z. Cui, and A.G. Cohn, "A Spatial Logic based on Regions and Connectivity," 1994, Division of Artificial Intelligence, School of Computer Studies, Leeds University.

[33] Java Topology Suite (JTS). See http://www.vividsolutions.com/jts/ (Accessed August 4, 2017).

[34] White House (2003), The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets. Washington, DC.

[35] T. W. H. Boyes, R. Isbell, "Critical infrastructure in the future city - developing secure and resilient cyber-physical systems," in Critical Information Infrastructures Security - 9th International Conference, CRITIS 2014, Limassol, Cyprus, October 13-15, 2014, Revised Selected Papers, 2014, pp. 13–23.

[36] S. Stelting and O. Maassen, Applied Java Patterns. SUN Microsystems Press, Prentice-Hall, 2002.

[37] E.A. Oliveira, M. Kirley, T. Kvan, J. Karakiewicz, and C. Vaz, Distributed and Heterogeneous Data Analysis for Smart Urban Planning. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 37–54.

[38] A. Bouchama, The IoT in the Service of the Environment using Apache Camel & JBoss A-MQ. For details, see: http://bushorn.com/iot-service-environment-using-apache-camel-jboss-mq/ (Accessed, Jul 1., 2017).

[39] Apache Camel (2017), Components Included. For details, see: http://camel.apache.org/components.html (Accessed, Jul 1., 2017).