Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Many-to-Many Association Relationships

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 societal-scale infrastructures 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. This work-in-progress 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 preliminary implementation is a collection of families interacting with a collection of school systems. We conclude with ideas for scaling up the simulations with mediators assembled from Apache Camel technology.**

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

I. Introduction

A. Problem Statement

The modern way of life is enabled by remarkable advances in technology (e.g., the Internet, smart mobile devices, cloud computing) and the development of urban systems (e.g., transportation, electric power, wastewater facilities and water supply networks, among others) whose operations and interactions have superior levels of performance, extended functionality and good economics. While end-users applaud the benefits that these technological advances 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. As a case in point, modern urban infrastructure systems comprise physical, communication and social networks that are spatially distributed, and defined by concurrent subsystem-level behaviors, distributed control and decision making, and interdependencies among subsystems that are not always well understood. In the past, engineers have kept these difficulties in check by designing subsystems that operate as independently as possible from one another. Today, however, it is recognized that subsystem independence and inferior levels of situational awareness come at a cost of sub-optimal functionality and performance. Overcoming these barriers makes future challenges in urban systems design and management are a lot more difficult than they used to be.

B. Cascading Failures in Decentralized Systems

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 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 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. 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 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.

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. In another 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 [1]. To complicate matters, 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 [2]. 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 objectives of

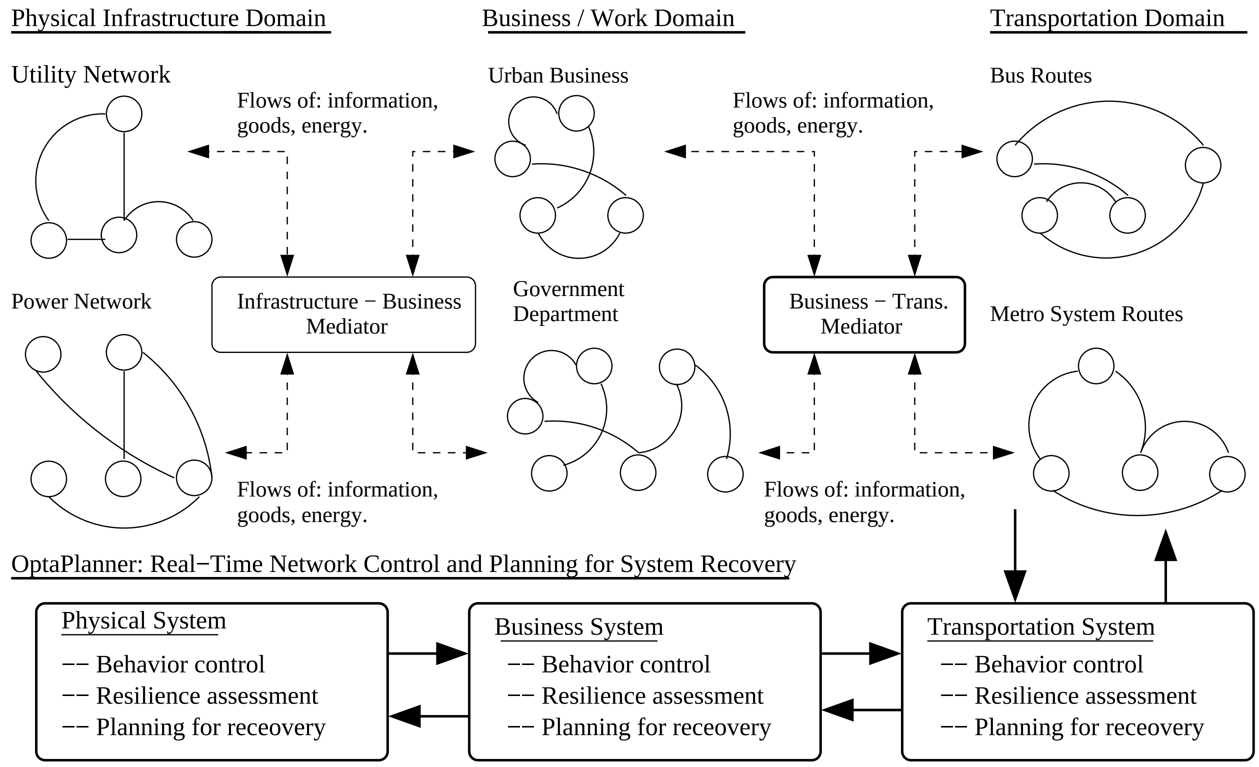
Physical Infrastructure Domain
Business / Work Domain
Transportation Domain
Utility Network
Flows of: information, goods, energy.
Urban Business
Flows of: information, goods, energy.
Bus Routes
Power Network
Infrastructure − Business Mediator
Government Department
Business − Trans. Mediator
Metro System Routes
Flows of: information, goods, energy.
Flows of: information, goods, energy.
OptaPlanner: Real−Time Network Control and Planning for System Recovery
Physical System
−− Behavior control
−− Resilience assessment
−− Planning for receovery
Business System
−− Behavior control
−− Resilience assessment
−− Planning for receovery
Transportation System
−− Behavior control
−− Resilience assessment
−− Planning for receovery

Figure 1. Architecture for multi-domain behavior modeling with many-to-many associations.

this work-in-progress paper are to explore opportunities for overcoming these limitations.

C. Scope and Objectives

In order to understand how cascading failures might be best managed, it is necessary to have the ability to model information exchange 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 1, and eventually, tools such as OptaPlanner [3] providing strategies for real-time control of behaviors, assessment of domain resilience and planning of recover actions in response to severe events. Instead of modeling the dynamic behavior of systems with centralized control and one large catch-all ontology, our work explores opportunities for modeling systems as collections of discipline-specific (or community) networks that will dynamically evolve in response to events. Each community will have a graph that evolves according to a set of community-specific rules, and subject to satisfaction of constraints. Communities will interact when then need to in order to achieve system-level objectives. If goals are in conflict, or resources are insufficient, then negotiation will need to take place.

This work-in-progress paper presents a model of system-level interactions that simulates distributed system behaviors through the use of ontologies, rules checking, and message passing mechanisms. The architecture builds upon the framework presented by Austin et al. [4], 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.

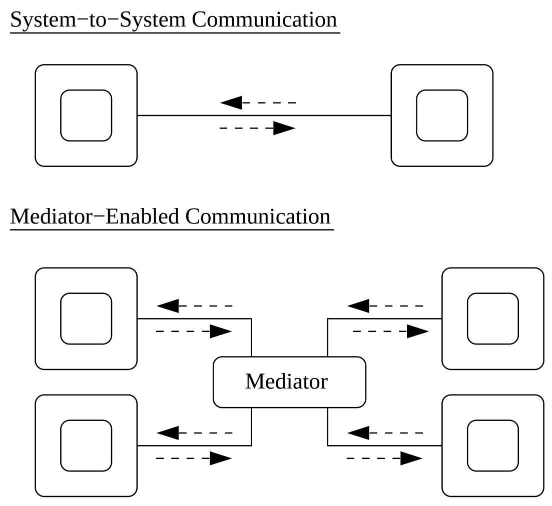
System−to−System Communication
Mediator−Enabled Communication
Mediator

Figure 2. Framework for communication among systems of type A and B.

As illustrated in Figure 2, one-to-one association relationships can be modeled with exchange of messages in a point-to-point communication setup. 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.

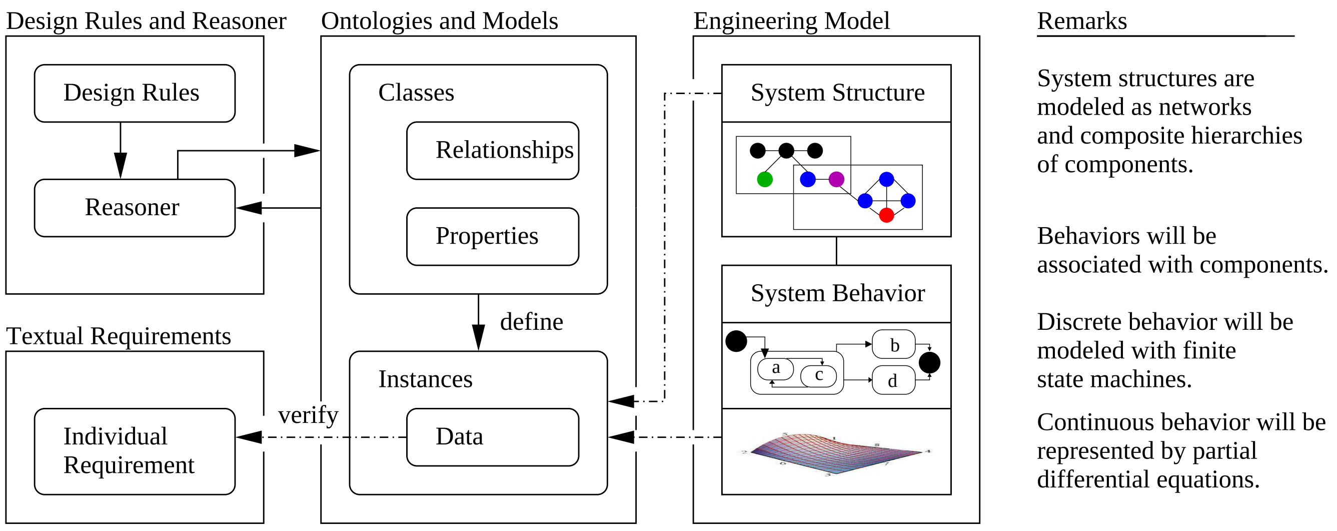
Design Rules and Reasoner
Design Rules
Reasoner
Ontologies and Models
Classes
Relationships
Properties
define
Instances
Data
Engineering Model
System Structure
System Behavior
a
c
b
d
Remarks
System structures are modeled as networks and composite hierarchies of components.
Behaviors will be associated with components.
Discrete behavior will be modeled with finite state machines.
Continuous behavior will be represented by partial differential equations.
Textual Requirements
Individual Requirement
verify

Figure 3. Framework for implementation of semantic models using ontologies, rules, and reasoning mechanisms (Adapted from Delgoshaei, Austin and Nguyen [5]).

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.

Section II covers the relationship of ontologies and rules to our related work in model-based systems engineering, Section III 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, and exercise this capability with an application involving collections of families interacting with multiple school systems. We conclude with ideas for scaling up the simulations with mediators assembled from Apache Camel technology.

II. Related Work

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 [6]. 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 [7] [8] [9].

Figure 3 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 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. This is a work in progress [10] [5].

III. Work in Progress

Topic 1. Distributed System Behavior Modeling with Ontologies and Rules

Figure 4 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 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. Instances (see Figure 3) 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 arent 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. 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

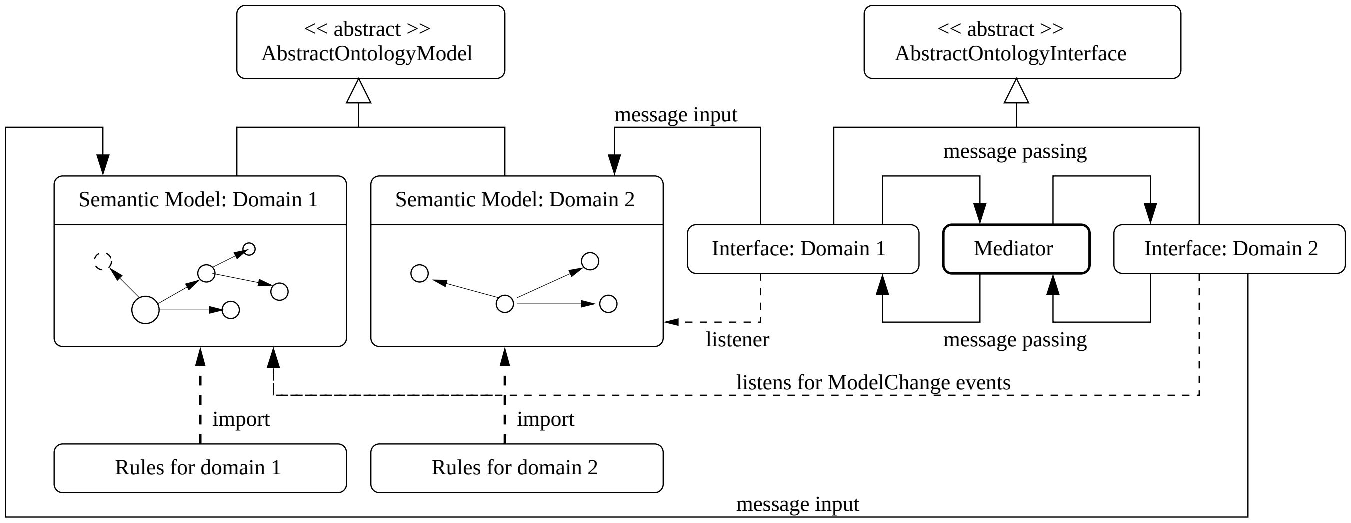

Figure 4. System architecture for distributed system behavior modeling with ontologies, rules, mediators and message passing mechanisms.

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 type (e.g., MessageType.miscellaneous), a message source and 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.

Topic 2. 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 1 and 2, 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 object. 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 [11]. 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 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.

Topic 3. 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. Our present-day capability is simplified in the sense that domain interfaces are assumed to be homogeneous. But looking forward, this will not always be true. 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 is Apache Camel [12] [13], 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. Figure 5 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.

IV. Case Study Problem

To illustrate the capabilities of our experimental architecture, we now present the essential details of a simulation framework for the behavior modeling of a multiplicity of families and school, defined by ontologies, rules, and exchange of information as messages. Figure 6 is an instantiation of the concepts introduced in Figure 4 and shows the software architecture for a family-school interaction. And Figure 7 is the network setup for three families interacting with elementary, middle and high schools.

As every parent knows, the enrollment process involves the exchange of specific information, such as the name, birth

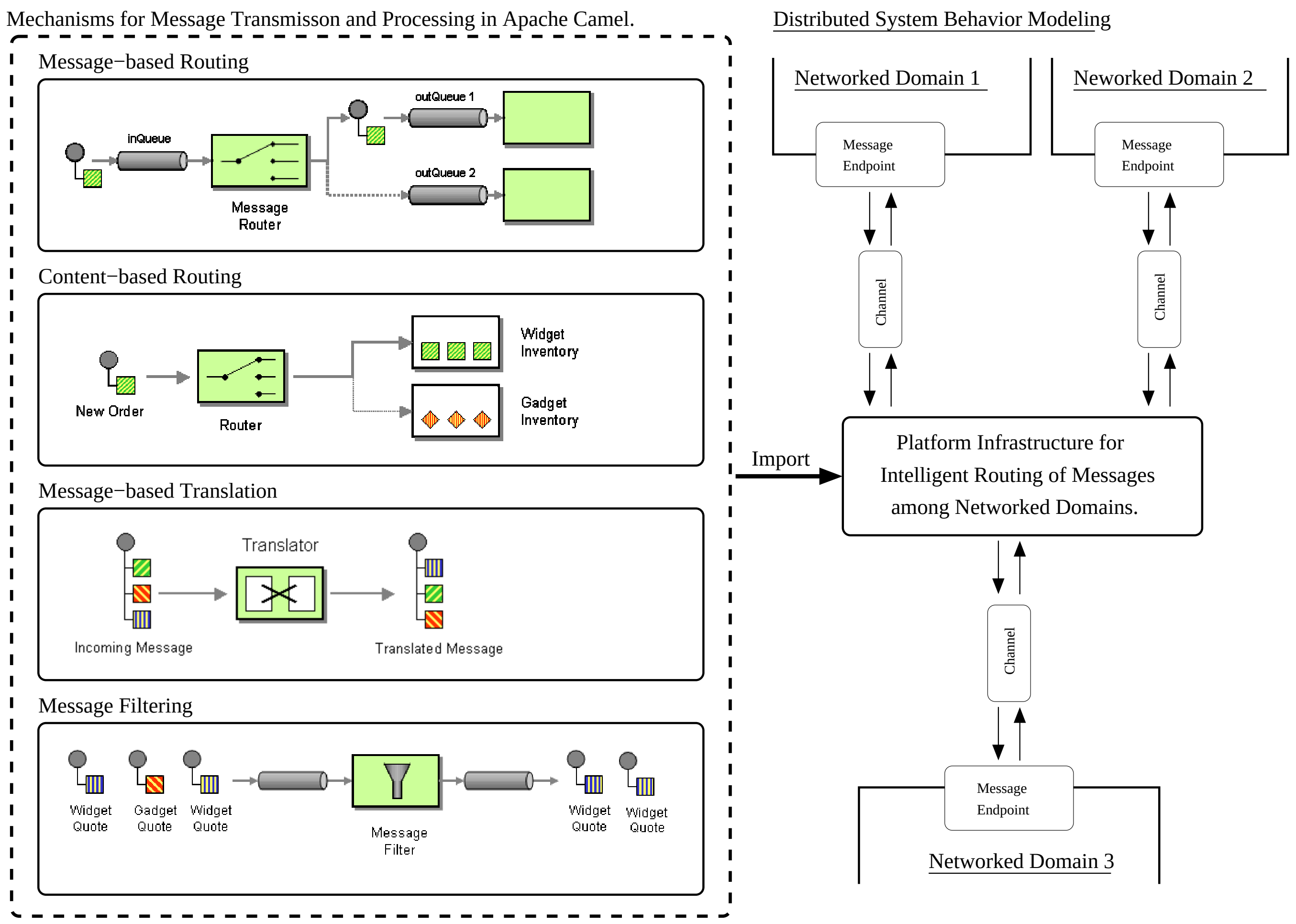

Figure 5. Platform infrastructure for distributed behavior modeling and intelligent communication (message passing) among networked domains.

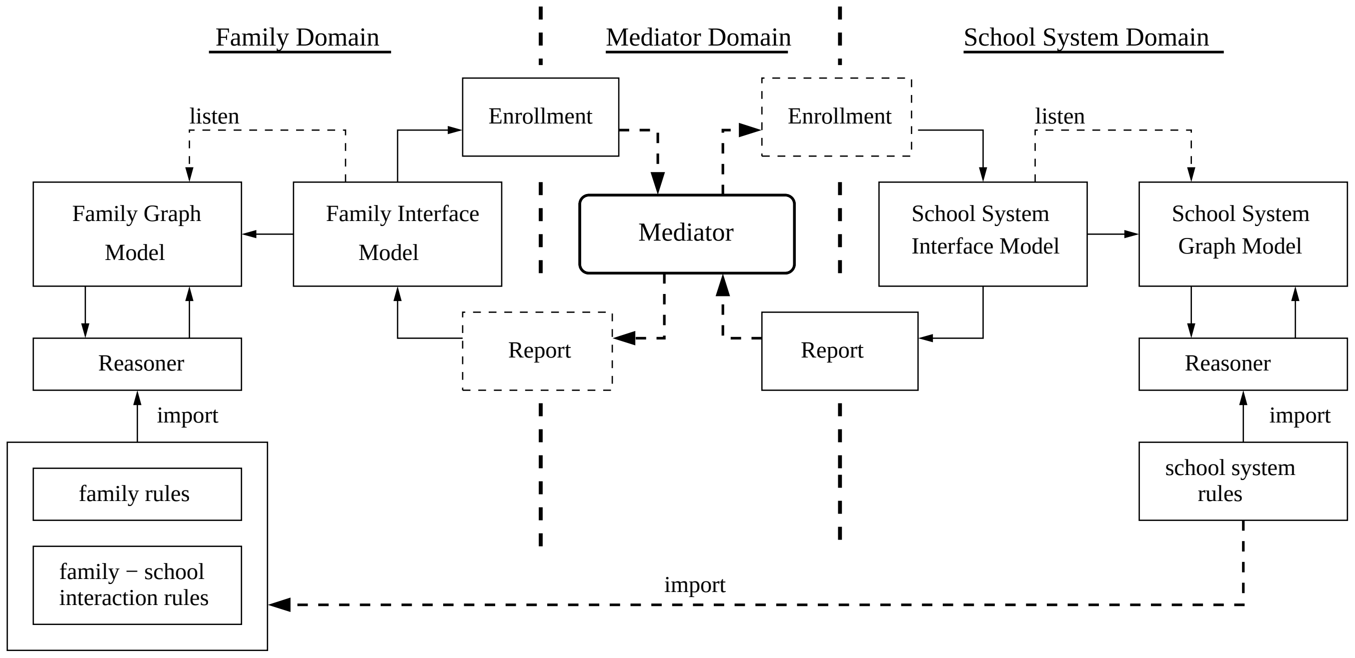

Figure 6. Software architecture for distributed behavior modeling in the family-school case study.

date, home address and social security number of each child. Then, once the child is accepted the school system takes over. They figure out what grade level is appropriate for each child, what classroom the child will be in, the schedule of learning activities, and when school reports will be sent home.

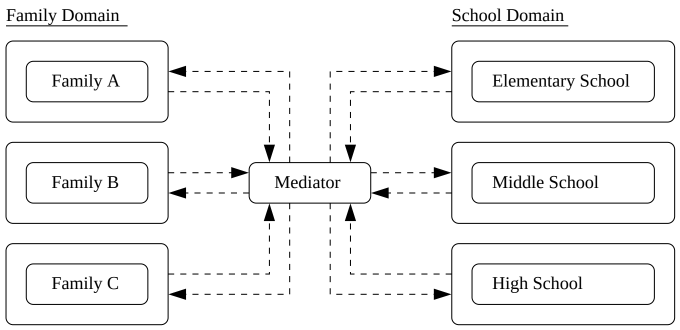

Figure 7. Framework for communication among multiple families and schools enabled by a mediator.

Communication among the family and school communities is handled by a mediator. Every component of the system (i.e., families and schools) register with the mediator as listeners. 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 a message, with source and destination properties. The mediator logic loops through all of its registered listeners to find a match with the message destination, and then destination listener is notified. 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 students 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.

V. 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 1, 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 1). 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 using Apache Camel technology in this context, especially as problem sizes and the number of participating domains scale up. A second important topic for future work is linkage of our simulation framework to tools for optimization and tradeoff 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.

References

[1] 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.

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

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

[4] 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.

[5] 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.

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

[7] 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.

[8] ——, “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.

[9] 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.

[10] 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.

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

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

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