Distributed System Behavior Modeling of Urban Systems with Ontologies, Rules and Message Passing Mechanisms

Maria Coelho

University of Maryland

mecoelho@terpmail.umd.edu Master Thesis Presentation

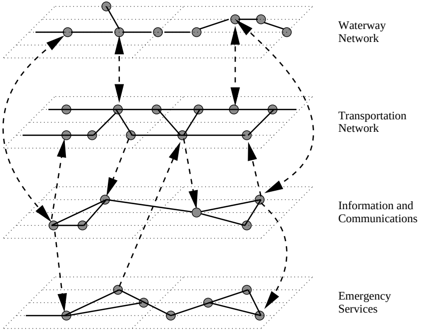
April 13, 2017

Overview

Problem Statement

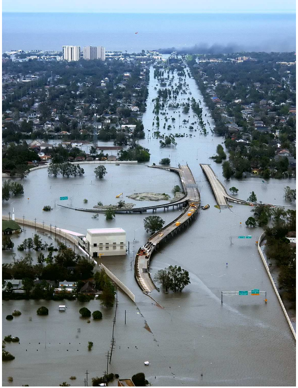
Interdependent Urban Networks

- Networks are heterogeneous, interwoven, dynamic.
- Disciplines want to operate independently in their domain.
- Achieving target levels of performance and correctness of functionality requires disciplines to coordinate activities at key points in system operation.

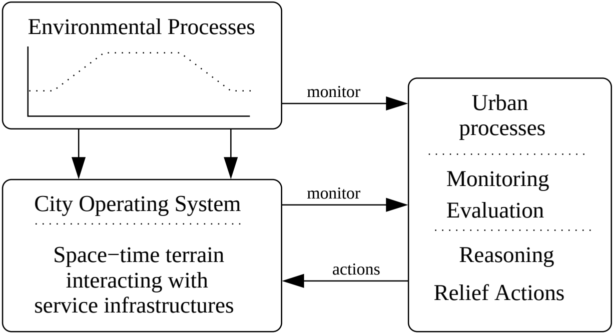


Cascading Failures

- Disturbance in one system can impact other networks in unexpected, undesirable and costly ways.
- Often, infrastructure management systems do not allow manager of one system to access operations and conditions of another system.
- Decision making is complicated by presence of newfound system interactions, incomplete knowledge of system state, and break downs of communication among urban networks.

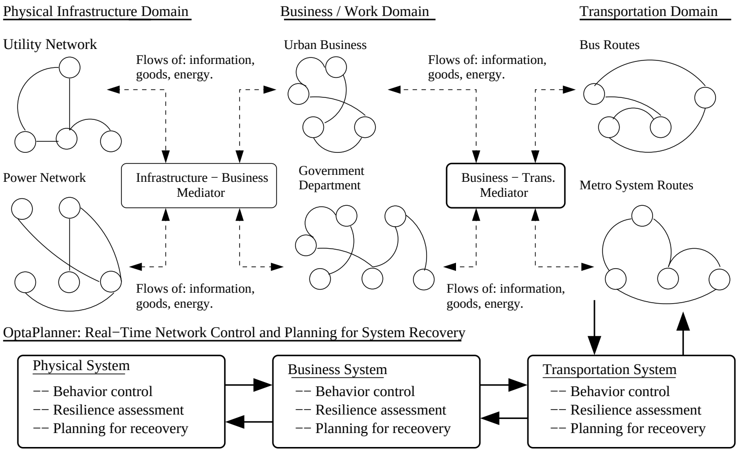


Long-Term Project Objective: City Operating System



Short-Term Project Objective: Behavior Modeling

- Ability to model behavior of city-domain processes, and interactions among distributed system behaviors within a city.



Benefits of Behavior Modeling

Allows decision makers to understand:

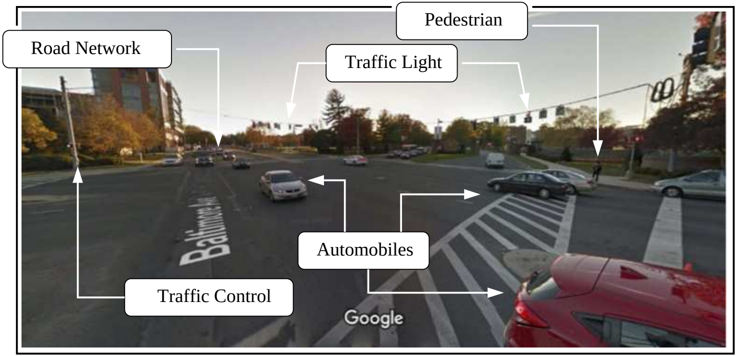
- How failure in one network will impact other networks.
- What parts of a system are most vulnerable.

Allows decision makers to assess:

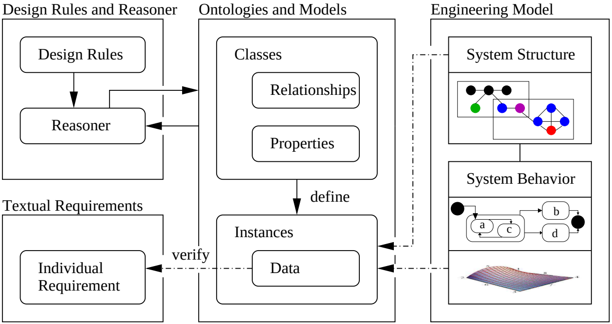
- Sensitivity of systems to model parameter choices.
- Influence of resource constraints.
- Potential emergent interactions among systems.

Solution Approach

Observing Urban Behavior



Ontologies, Rules, and Reasoning Mechanisms



Remarks

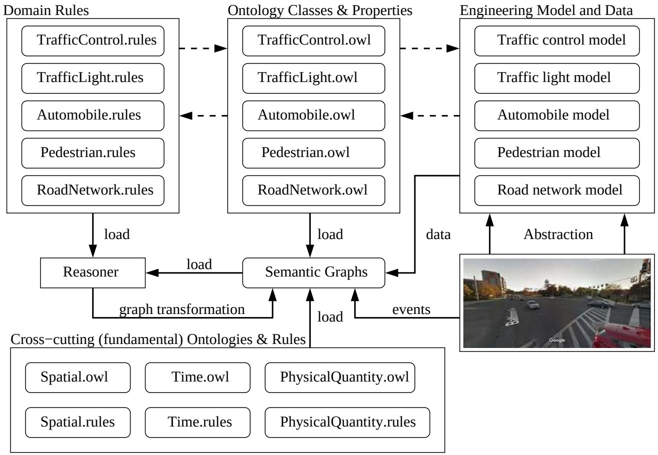
System structures are modeled as networks and composite hierarchies of components.

Behaviors will be associated with components.

Discrete behavior will be modeled with finite state machines.

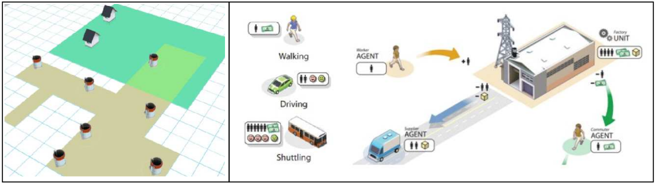
Continuous behavior will be represented by partial differential equations.

Ontologies, Rules, and Reasoning Mechanisms



Related Work

Glassbox Simulation Engine



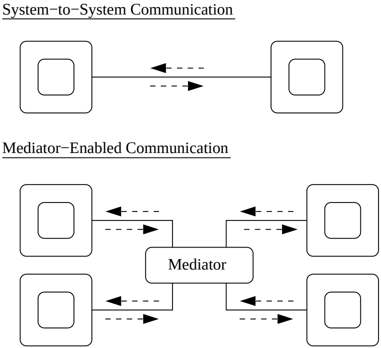
Graphs, Cellular Automata, and Ontologies

- Numerous researchers have studied the topology of urban environments from a graph theoretic standpoint.
- Other studies capture the temporal dynamics of cities with cellular automata, agent-based models, and fractals.
- Extensive studies have been conducted on the development of ontologies for the geographic information sector.
- Researchers have proposed so called smart city ontologies.
- A notable effort in the direction of ontologies developed alongside rules is the DogOnt ontology model.

Contributions

Contributions

- Framework for modeling concurrent, directed communication between all entities composing a system.

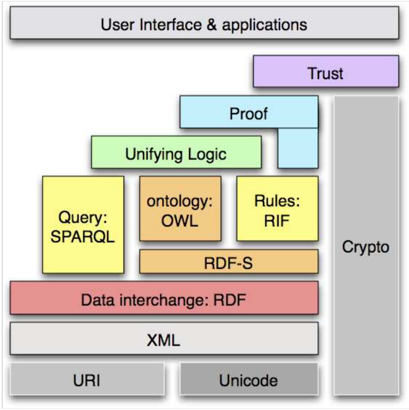


- Mechanisms for incorporating notions of space and time in the reasoning process.

Semantic Modeling

Introduction to the Semantic Web

- Extension to the World Wide Web
- Allows machines to access and share information.
- Relies on technical infrastructure below.



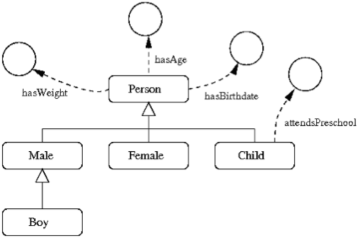
Working with Jena and Jena Rules

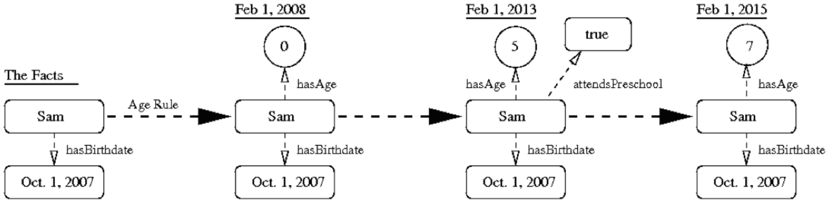
Fact. Sam is a boy. He was born October 1, 2007.

Rule 1: For a given date of birth, a built-in function getAge() computes a person's age.

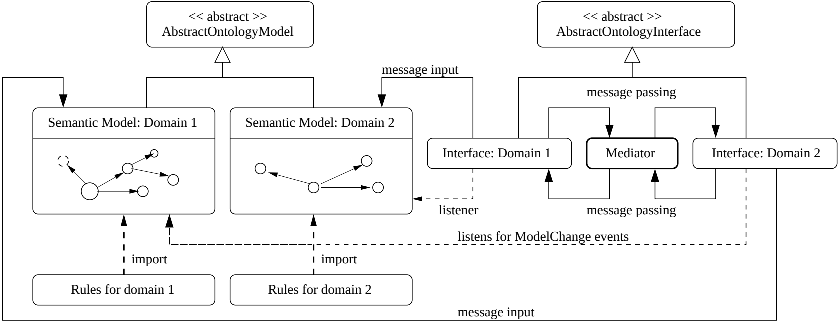
Rule 2: A child is a person with age < 18.

Rule 3: Children who are age 5 attend preschool.

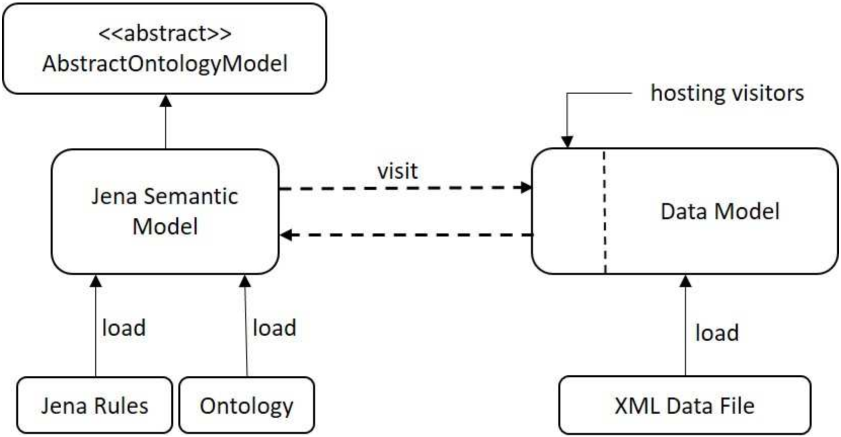




Distributed Behavior Modeling

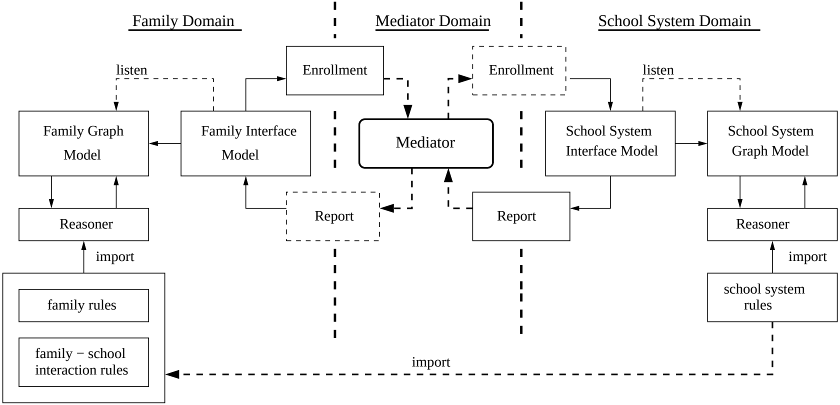


Generation of Semantic Models

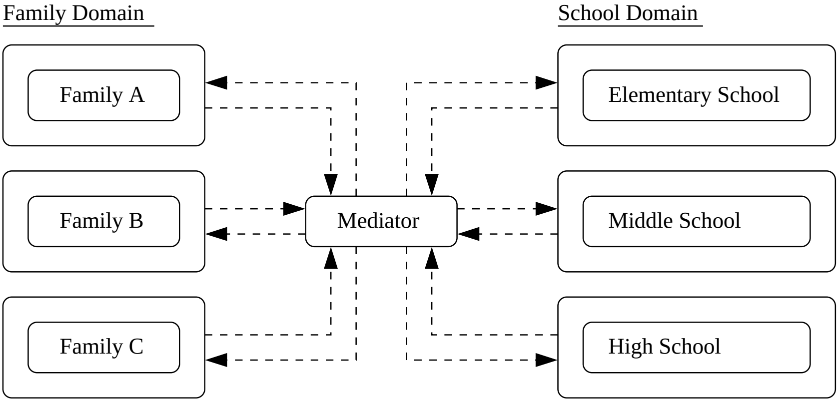


Case Study 1

Family-School System Dynamics



Framework for Communication



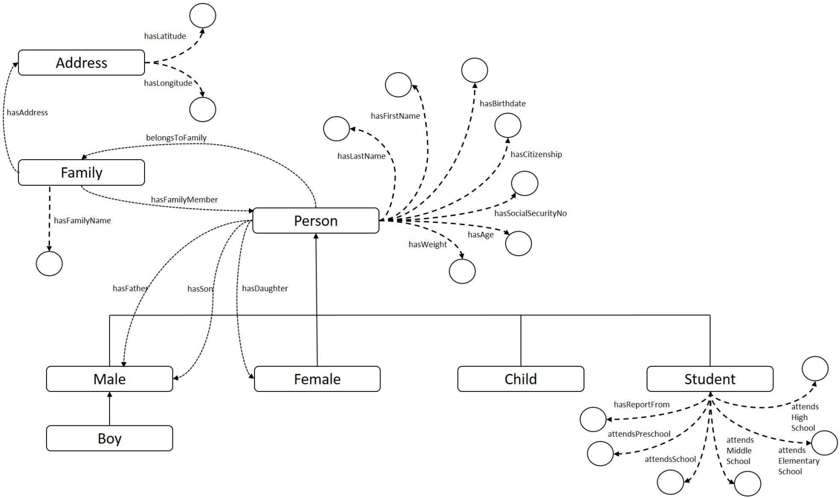
Family XML Datafile


```xml
<?xml version="1.0" encoding="UTF-8"?>
<FamilyModel author="Maria Coelho" date="2017" source="UMD">
<Family>
    <attribute text="FamilyName" value="Austin"/>
    <attribute text="Address" value="6242 Heather Glen Way, Clarksville, MD 21029"/>
    <Person>
        <attribute text="Type" value="Male"/>
        <attribute text="FirstName" value="Mark"/>
        <attribute text="MiddleName" value="William"/>
        <attribute text="LastName" value="Austin"/>
        <attribute text="BirthDate" value="1704-06-10"/>
        <attribute text="Weight" value="170.0"/>
        <attribute text="Citizenship" value="New Zealand"/>
        <attribute text="SocialSecurity" value="111"/>
    </Person>
    <Person>
        ... description of other Austin family members ....
    </Person>
</Family>
<Family>
    <attribute text="FamilyName" value="Jones"/>
    <attribute text="Address" value="5807 Laurel Leaves Ln, Clarksville, MD 21029"/>
    <Person>
        ... description of Jones family members....
    </Person>
</Family>
</FamilyModel>
```

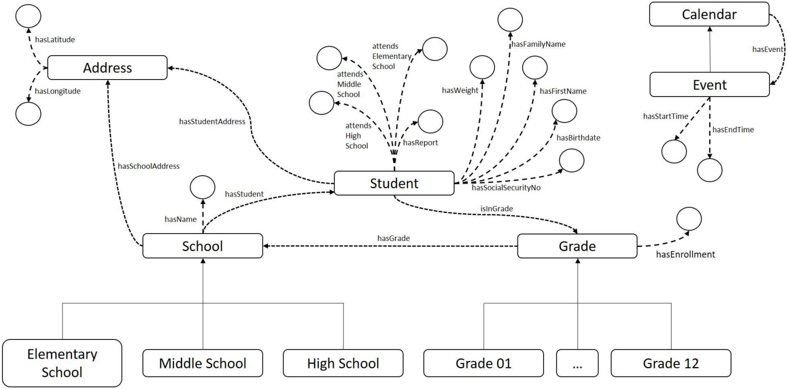
School XML Datafile


```xml
<?xml version="1.0" encoding="UTF-8"?>
<SchoolSystemModel author="Maria Coelho" date="2017" source="UMD">
   <School>
       <attribute text="Type" value="High School"/>
       <attribute text="Name" value="River Hill High School"/>
       <attribute text="Grade" value="Grade09"/>
       <attribute text="Grade" value="Grade10"/>
       <attribute text="Grade" value="Grade11"/>
       <attribute text="Grade" value="Grade12"/>
       <attribute text="Report Period Start Time" value="2016-09-01T00:00:00"/>
       <attribute text="Report Period End Time" value="2020-10-20T00:00:00"/>
   </School>
   <School>
       ... description of Clarksville Middle School ...
   </School>
   <School>
       ... description of Pointers Run Elementary School ...
   </School>
</SchoolSystemModel>
```

Family Ontology



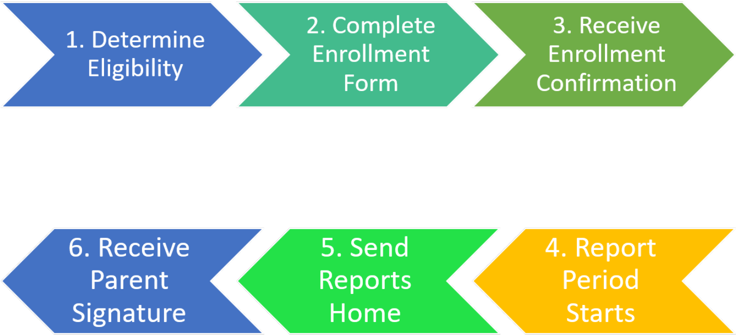
School Ontology



Family Rules - A Sample

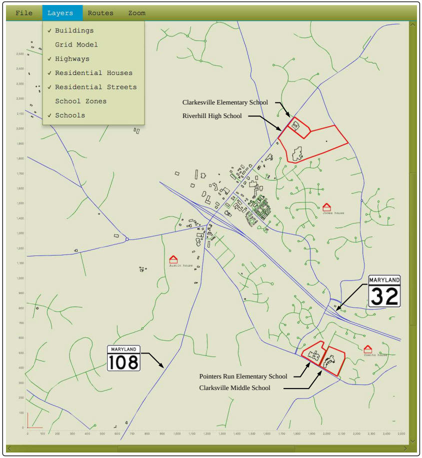
```
@prefix af:  <http://austin.org/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
// Rule 01: Propagate class hierarchy relationships ....
// Rule 02: Family rules ....

// Rule 03: Identify a person who is also a student ...

[ Student: (?x rdf:type af:Person) (?x af:hasAge ?y)
           greaterThan(?y, 4)
           lessThan(?y, 18) -> (?x rdf:type af:Student) ]

[ UpdateStudent: (?x rdf:type af:Student) (?x af:hasBirthDate ?y)
           getAge(?y,?b) ge(?b, 18) -> remove(0) ]

// Rule 04: 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) ]
```

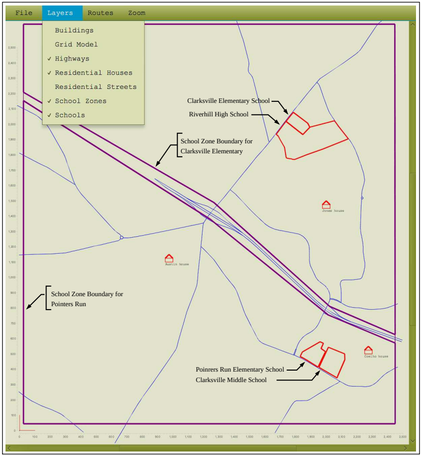
Behavior Modeling Use Cases



Case Study 2

Family-School-Urban-Geography System Dynamics

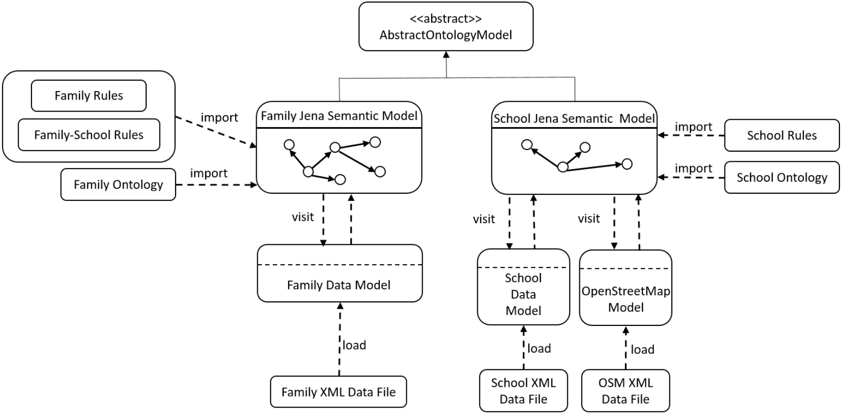




Extensions to Ontologies and Rules

- Certain ontology properties are added to the framework in order to allow modeling spatial behavior (e.g. livesInSchoolZoneOf, isEligibleForSchoolBus)
- Additional rule determines whether or not a person is eligible to the school bus service
- Additonal rule only allows students to enroll when they live within the school zone jurisdiction.
- Graph transformations in the school system model can now occur due not only to input or time, but also space.

Accessing Spatial Data from OpenStreetMap



Conclusions

and

Future Work

Conclusion and Future Work

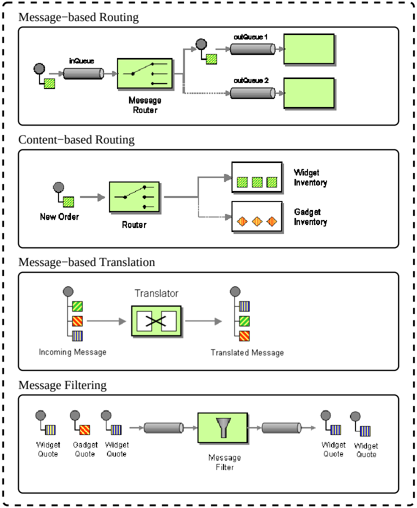
- Project focused on design and preliminary implementation of message passing infrastructure needed to support communication in many-to many association relationships connecting domain-specic networks.
- Long-term objective is to build upon family-school distributed behavior model and create models of distributed behavior of urban infrastructure multi level systems, and simulate cascading system failures that occur due to extreme external events.
- Domain interfaces have been assumed to be homogeneous, but will not always be the case.
- Need for new approaches to the construction and operation of message passing mechanisms.

Future Work

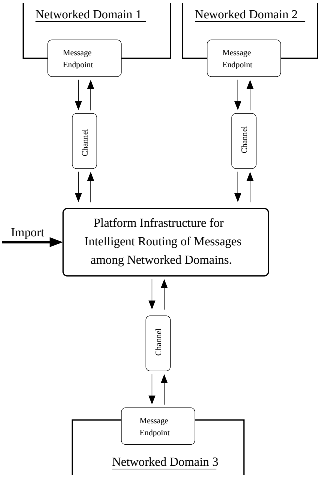
- Apache Camel is an open source Java framework that focuses on making Enterprise Integration Patterns (EIP) accessible through carefully designed interfaces, the base objects, commonly needed implementations, debugging tools and a configuration system.
- In addition to basic content-based routing, Apache Camel provides support for filtering and transformation of messages.

Future Work

Mechanisms for Message Transmisson and Processing in Apache Camel.



Distributed System Behavior Modeling



Future Work

- Linkage of presented simulation framework to tools for optimization and tradeoff analysis.
- Implement natural language processing for the identification of knowledge provided by the datafiles when constructing ontologies.

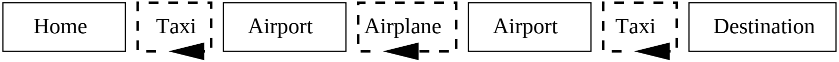
Extra Slides

Solution Approach: Systems of Systems Perspective

- A **System of Systems** is a collection of independently operational systems which have been glued together to achieve further emergent properties.
- The component systems operate for their own purposes rather than the purposes of the combined system.
- Yet, they also function to resolve the purposes of the whole which are generally unachievable by the individual systems acting independently.
- The system of systems will change over time as constituent system are replaced.

Solution Approach: Systems of Systems Perspective

- Cities are System of Systems.
- City subsystems may have a preference to operating as independently as possible from the other subsystems.
- Strategic collaboration among subsystems is often needed to either avoid cascading failures across systems and/or recover from a loss of functionality.



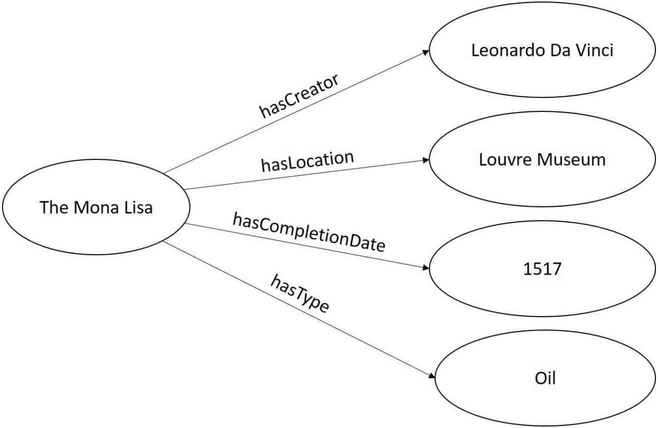
Solution Approach: Model Based Systems Engineering

- Understanding the relationships among the networks and their combined behaviors can be very challenging.
- City systems are being upgraded from industrial-age capability to information-age capability.
- Challenges can be mitigated through the systematic application of model-based systems engineering (MBSE) procedures.
- State-of-the-art MBSE procedures fall short is in the systematic consideration of interactions among many concurrent behaviors.

Working with RDF

- RDF is a graph-based assertional data model for describing the relationships between objects and classes.
- Assertions are transformed into RDF triples consisting of a subject, a predicate and an object.
- A set of related triples constitute an RDF graph

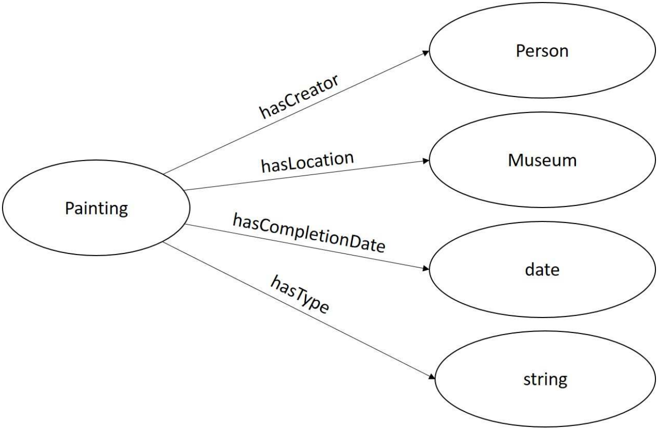
Working with RDF



Working with OWL

- RDF is unable to capture existence,cardinality, localized range, domain constraints, transitivity, inverse or symmetrical properties.
- OWL was developed to address the weaknesses of RDF.
- The additional capabilities allow ontological systems to use reasoning to infer new triples from existing ones.

Working with OWL



Working with OWL

```
// Define Classes ...

<owl:Class rdf:about="http://example.org/monaLisa#Painting">
</owl:Class>

<owl:Class rdf:about="http://example.org/monaLisa#Person">
</owl:Class>

<owl:Class rdf:about="http://example.org/monaLisa#Museum">
</owl:Class>

// Define Datatype Properties ...

<owl:DatatypeProperty rdf:about="http://example.org/monaLisa#hasType">
   <rdfs:domain rdf:resource="http://example.org/monaLisa#Painting"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://example.org/monaLisa#hasCompletionDate">
   <rdfs:domain rdf:resource="http://example.org/monaLisa#Painting"/>
   <rdfs:range rdf:resource="&xsd;date"/>
</owl:DatatypeProperty>
```

Working with OWL

```
// Define Object Properties ...

<owl:ObjectProperty rdf:about="http://example.org/monaLisa#hasCreator">
   <rdfs:domain rdf:resource="http://example.org/monaLisa#Painting"/>
   <rdfs:range rdf:resource="http://example.org/monaLisa#Person"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://example.org/monaLisa#hasLocation">
   <rdfs:domain rdf:resource="http://example.org/monaLisa#Painting"/>
   <rdfs:range rdf:resource="http://example.org/monaLisa#Museum"/>
</owl:ObjectProperty>
```

Working with Jena and Jena Rules

- Apache Jena is an open source Java framework for building Semantic Web and linked data applications.
- Jena provides APIs for developing code that handles RDF, RDFS, OWL and SPARQL.
- Jena inference subsystem is designed to allow a range of inference engines or reasoners to be plugged into Jena (e.g. Jena Rules).
- Jena Rules use facts and assertions described in OWL to infer additional facts from instance data and class descriptions.
- Such inferences result in structural transformations to the semantic graph model.

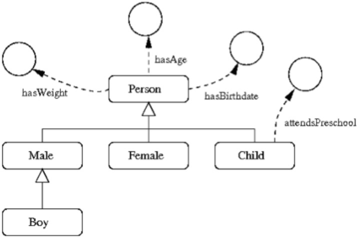
Working with Jena and Jena Rules

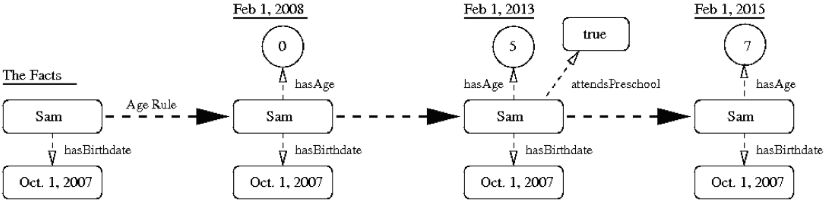
Fact. Sam is a boy. He was born October 1, 2007.

Rule 1: For a given date of birth, a built-in function getAge() computes a person's age.

Rule 2: A child is a person with age < 18.

Rule 3: Children who are age 5 attend preschool.





System Modeling Assumptions

- All of the models execute under a single continuous thread of computation, with the only interaction among domains being exchange of messages.
- Delays in communication between domains are ignored.
- Behavior models are deterministic; uncertainties in behavior are ignored.
- Support for fault-tolerant communication among domains is ignored. We do, however, send confirmation messages back to the sender.

Family Rules

```
@prefix af:  <http://austin.org/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 02: 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 03: Identify a person who is also a student ...
[ Student: (?x rdf:type af:Person) (?x af:hasAge ?y)
      greaterThan(?y, 4) lessThan(?y, 18) -> (?x rdf:type af:Student) ]
[ UpdateStudent: (?x rdf:type af:Student) (?x af:hasBirthDate ?y)
      getAge(?y,?b) ge(?b, 18) -> remove(0) ]
// Rule 04: 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)]
```

School Rules

```
@prefix af:  <http://austin.org/school#>.
@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)]
// 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 03: Middle school rules ...
... Middle school rules removed ...
// Rules 04: High school rules ...
...High school rules 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)  ]
```

Accessing XML Data from Data Models