Integrated Modeling, Mapping, and Simulation (IMMS) Framework for Planning Exercises

Todd Plantenga, Ernest Friedman-Hill
Sandia National Laboratories
PO Box 969, Livermore, CA 94550
tplante@sandia.gov, ejfried@sandia.gov

ABSTRACT

Emergency management personnel at federal, state, and local levels can benefit from the increased use of simulation and modeling for emergency preparedness, including planning, exercises and training. The Department of Homeland Security (DHS) is chartered to identify and address modeling opportunities for national preparedness. Dozens of modeling efforts relevant for emergency response have been identified, but knowledge of existing resources and the expertise needed to provision a simulation with data, execute it, and synthesize results are not uniformly available to all. To meet these needs, the DHS Science and Technology Directorate has spearheaded the Integrated Modeling, Mapping, and Simulation (IMMS) program to create a framework that brings relevant tools to the emergency response community.

IMMS defines a software framework that brings together distributed codes using metadata, heuristic domain knowledge and a uniform interface, to provide integration capability and automated execution. A fundamental goal is to connect users such as training and exercise planners with modeling resources. A fundamental challenge is to bridge the gap in expertise and technical skills between these two communities, a gap which hinders model discovery, provisioning, execution, and interpretation.

We present a platform-neutral, distributed computing framework that connects users (logged in as clients) with models (located on servers). Similar frameworks have been proposed or prototyped, including FAIT for a small set of infrastructure models, *iCAV* and *Palanterra* for GIS modeling, and DIAS for discrete event simulations. IMMS contributes two distinct innovations that bring together users and model providers: a discovery process based on web ontology taxonomy trees, and an abstraction for modeling emergency planning scenarios. The discovery process allows subject matter experts to contribute metadata, while enabling less sophisticated users to find relevant models. The abstraction uses "simulation templates" to group models according to function, and then combine functionalities to address hazard and threat scenarios.

The paper will describe the framework architecture, the innovative discovery and simulation template components, and an initial reference implementation to be used in FEMA National Level Exercises (NLE 2010 and NLE 2011).

ABOUT THE AUTHORS

Todd Plantenga is a Principal Member of the technical staff at Sandia National Laboratories. He has a PhD in EE/CS and over 15 years industry experience developing scientific computing models and algorithms. His research interests include mathematical optimization, data mining, and distributed computing.

Ernest Friedman-Hill is a Principal Member of the technical staff at Sandia National Laboratories. He is the creator of the Jess rules engine and author of *Jess in Action* from Manning Publications. His research interests include machine intelligence, distributed computing, and simulation.

Integrated Modeling, Mapping, and Simulation (IMMS) Framework for Planning Exercises

Todd Plantenga, Ernest Friedman-Hill
Sandia National Laboratories[1]
PO Box 969, Livermore, CA 94550
tplante@sandia.gov, ejfried@sandia.gov

INTRODUCTION

Emergency management personnel at federal, state, and local levels can benefit from the increased use of simulation and modeling for emergency preparedness, including planning, exercises and training. The Department of Homeland Security Science and Technology Directorate (DHS/S&T) has identified dozens of potentially useful models across academia, commercial industry, national laboratories, and government. For example, the National Center for the Study of Preparedness and Catastrophic Event Response (PACER), a DHS Center of Excellence, compiled a list of more than 110 simulation models of relevance to catastrophic event planning and response (Coolahan, 2007). DHS/S&T has brought together key stakeholders such as the Federal Emergency Management Agency (FEMA) and the Office of Emergency Communications to develop requirements for making models available and useful to the emergency response community (DHSST, 2008). Access to and economical use of modeling and simulation tools were recognized as a significant capability gap at national, state, and local levels.

DHS/S&T is addressing this gap by funding the Integrated Modeling, Mapping, and Simulation (IMMS) program. IMMS is architecting a framework that will enable analysts, emergency planners, and incident managers to more effectively prepare for, analyze, and respond to real or potential catastrophic events. The program vision is to create a capability for linking together modeling and simulation tools for easier use.

To realize this vision, a software architecture called SUMMIT – the Standard Unified Modeling and Mapping Integration Toolkit – is being iteratively designed and prototyped. SUMMIT provides a platform-neutral framework that brings together distributed modeling codes and a wide range of users. The framework makes it easier to find and integrate models, provision them for a specific scenario, execute models on available resources, and deliver results to a collaborating set of users.

This paper briefly describes the architectural design of SUMMIT, and then highlights two innovative features: an abstraction that we call *simulation templates* which organizes models according to planning needs, and a discovery process that allows novice and experienced users to find the most appropriate modeling tools.

Contributions of the Paper

Software integration frameworks for modeling and simulation are not a new idea (Jain, 2006) (Pederson, 2006). McLean and Jain recognized the need for a framework and proposed a classification scheme to enable model integration for emergency response training (McClean, 2007). The Dynamic Information Architecture System (DIAS) provides an object oriented software framework for integrating discrete event simulation models (Simunich, 2005). The US Army Corps of Engineers “Fort Future” project brings together systems of models to validate proposed infrastructure installations (Case, 2008). The Fast Analysis Infrastructure Tool (FAIT) integrates a small set of infrastructure models for rapid scenario analysis (FAIT, 2010). Geospatial Information System (GIS) visualization tools provide a loose sort of integration for models that operate on geospatial data; for example, the Department of Defense *Palanterra* (Beaulieu, 2004) and DHS *iCAV* (iCAV, 2010) tools. The Department of Defense’s High-Level Architecture (HLA) is a well-known contributor in this space (Kuhl, 1999). We note that SUMMIT is not intended as an alternative to HLA; it operates at a higher level by identifying appropriate resources and relying on execution frameworks like HLA for code-level integration.

[1]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

The contribution of SUMMIT is a framework whose prime goal is to bring together users and modeling tools from traditionally separate communities. The contribution of this paper is to discuss two innovations that further this goal.

ARCHITECTURE OVERVIEW

The SUMMIT architecture is designed to allow for maximum flexibility, placing few restrictions on federated models but still providing necessary capabilities for integration. Model owners decide who has permission to use their code, and where the executing code is hosted. Emergency planners and other users link models as needed to address specific emergency scenarios. As an example, suppose the emergency incident is release of chlorine gas from a railcar in an urban setting, and the emergency planner wants to know if there are sufficient medical supplies for first response. This scenario might incorporate a finite element model that computes the chemical gas dispersion plume given current weather conditions, another model that quantifies casualties in the population at risk, and a third model that tallies available medical resources. SUMMIT makes it possible for the planner to link these three models, execute them, and view results, all from a single client interface.

Software components are divided between a central server, user clients, and executable models that are usually hosted on the machines of model owners. Figure 1 shows the main components of SUMMIT. The “data” and “model” symbols in the figure are components owned by model contributors. All other symbols are part of the SUMMIT framework. A SUMMIT Client component allows for interaction with the user, and a SUMMIT Software Development Kit (SDK) provides tools for model owners that ease the process of model integration. Components in the SUMMIT server provide system functionality through a set of distributed core services.

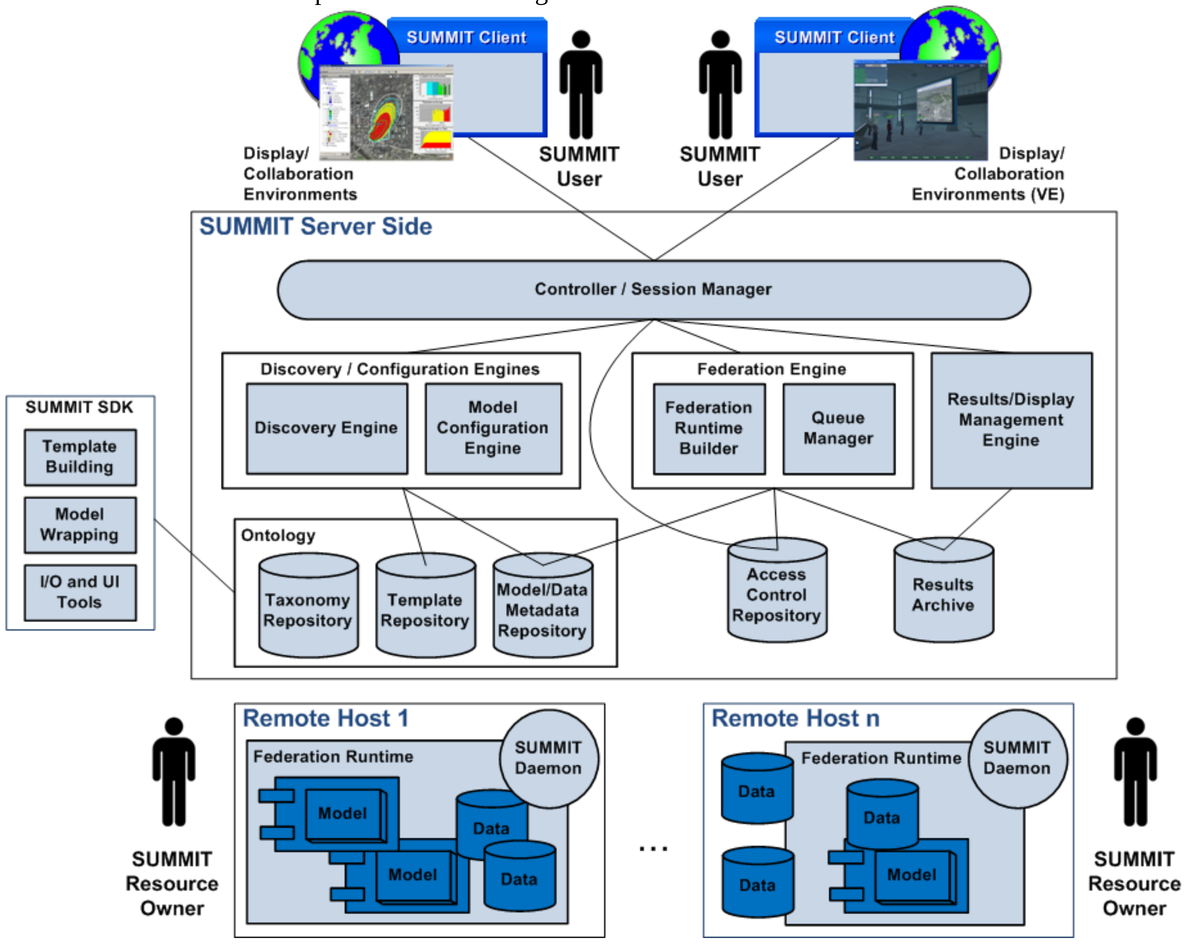


Figure 1. SUMMIT Architecture

SUMMIT provides for three types of users: model owners, scenario planners, and emergency responders who are the primary end user. *Model owners* use SUMMIT SDK tools to create a software wrapper that enables execution as part of a SUMMIT-mediated federation of models. (Note that in this paper the term "federation of models" does not refer to an HLA type of federation, and the Federation Runtime Builder in Figure 1 is not related to HLA Runtime Infrastructure. A SUMMIT federation assumes each model executes in isolation, given a correct set of inputs, and executes only once during a scenario simulation. Future versions of SUMMIT will relax this requirement, allowing dynamic interaction between models, including models that run in an HLA RTI.) For example, the three models in the chlorine gas scenario described earlier might be contributed by three different model owners and hosted at three different remote sites. *Scenario planners* use SUMMIT SDK tools to create simulation templates that bring together models for a specific incident scenario. In the chlorine gas example a scenario planner created the simulation template by linking three models at a conceptual level. These two user groups work offline to add content into the SUMMIT framework. The final *end users* access content through a SUMMIT Client. They log into the system, discover an appropriate simulation template, configure inputs, and then view results after SUMMIT automatically executes the models that compose the simulation template.

Further details on the coordination and execution of models in the SUMMIT framework are discussed in (Friedman-Hill, 2010). This paper now turns its attention to simulation templates and how they are discovered by end users.

SIMULATION TEMPLATE ABSTRACTION

The simulation template concept is fundamental to the SUMMIT architecture. Simulation templates provide an abstract representation of a hazard or incident defined by a scenario planner. A template also defines the components and parameters needed to form an executable simulation, which typically comprises multiple models. Figure 2 shows a simulation template whose purpose is to quantify medical risks to a population subjected to an aerosolized chemical agent release. The specific example of chlorine gas release discussed earlier fits into this abstraction. The template consists of 3 slots (rectangles), data flow connections between slots (thin arrows), template-level inputs (thick arrows pointing in), and template-level outputs (thick arrows pointing out). The end user supplies template-level inputs and views template-level outputs.

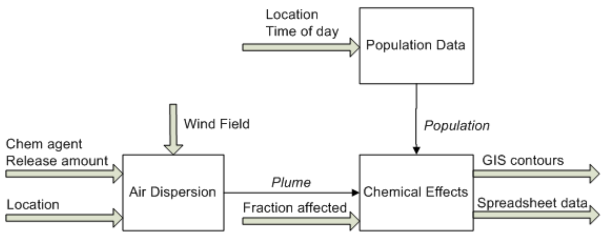


Figure 2. Example Simulation Template

Each slot in a template is an abstract functional description filled by an actual model or simulation tool. To be a candidate for a slot, a model must perform the function, accept all inputs, and produce all outputs. For example, in Figure 2 the *Air Dispersion* slot can accept any model that produces a "plume" output (geospatial contours of chemical agent at a fixed concentration) in response to the four inputs: wind field, chemical agent type, release location, and release amount.

The simulation template and slot abstractions promote reuse of models. For instance, the *Population Data* slot outputs a table of numbers describing inhabitants in a particular urban region at a particular time. The same slot can easily be used in simulation templates that assess populations at risk from other hazards (biological weapons, flood, etc.).

Slots are an abstraction for models to support the idea of alternate models with similar functionality. The *Air Dispersion* slot might be filled by one of several candidate models (EPA, 2010 provides one list), perhaps differentiated by their resolution, or special features like wind tunnel effect in downtown cities. Any model wrapped to fit the slot specifications is capable of executing as part of the simulation template. The end user decides on a specific model for each slot after choosing a simulation template.

Software Integration of Models

Simulation templates greatly simplify the process of provisioning and executing models from the viewpoint of a scenario planner or end user. To achieve this, the SUMMIT framework precisely defines interfaces for each slot, and an execution scheme for each template. Together, these enable automatic execution of a simulation once inputs are provided by a user.

The execution scheme is generated automatically from the connection topology of slots within a simulation template. SUMMIT currently assumes that any model in a slot can be executed independently. SUMMIT supplies inputs, invokes the model based on stored instructions from the model owner, and collects outputs of interest. The SUMMIT Federation Runtime Builder determines an order of execution based on the layout of slots in a template, and then directs a cascade of executables, linking outputs to inputs until all models have run. The independence of models implies they can be run on remotely hosted machines; thus, model owners can host their resource and SUMMIT acts more as a broker that coordinates multiple models.

Inputs and outputs of a slot define a software interface for candidate models. Combined with execution instructions, these define an application programming interface (API) for each slot. When a model is wrapped to implement the API, it becomes a *SUMMIT-compliant* model that can be invoked by the Federation Runtime Builder. The API for a slot is generated by SUMMIT SDK tools in the Java programming language as a Java interface class. Java is chosen because it is platform-neutral, supports a broad range of communication services, and is extremely flexible for wrapping codes (in any programming language). A model is wrapped for a slot by implementing the methods in the Java API. The model is declared SUMMIT-compliant when it executes successfully from a test template supplied by the SDK tool.

SUMMIT DataTypes

Input and output definitions in a software interface are nontrivial. Figure 2 shows the end user view, which has simple descriptions like “Location” and “Plume”, but the Java API hidden from end users is much more detailed. SUMMIT employs Google Protocol Buffers (Google, 2010) to define data containers, and then generates Java classes with Protocol Buffer tools. For instance, “Location” is defined as container with latitude and longitude, each a double precision number containing decimal degrees, the first ranging between -90 and +90, the second between -180 and +180.

DataType definitions are stored in a SUMMIT metadata repository and can be reused freely in any number of simulation templates. Each time a DataType appears in a template, its visible display name can be customized. For example, the “Location” DataType in Figure 2 that designates the point of chemical release might appear in a different template with the name “Hospital Location”.

Metadata definitions for data containers have been considered elsewhere, most notably the National Information Exchange Model (NIEM), a comprehensive effort led by the U.S. Department of Homeland Security and U.S. Department of Justice (NIEM, 2010). The XML schemas of NIEM provide a good reference point for an initial set of DataTypes in SUMMIT. For example, *TwoDimensionalGeographic-CoordinateType* and *ThreeDimensionalGeographic-CoordinateType* provide basic definitions for a “Location” DataType. However, NIEM XML is not used as the data content in SUMMIT because the schemas are not designed for modeling and simulation tools. Definitions sometimes have too much formal abstraction, and many topics in emergency management are not covered at all. Instead, SUMMIT uses the NIEM schemas as guidance for specifying the content of Google Protocol Buffers.

One of the goals of SUMMIT is to bring together existing models from a variety of sources. It is unlikely that models will follow a common set of DataTypes, and inefficient to insist that models be rewritten to accommodate a different API. Instead, SUMMIT supports the idea of Adapters to make simple data translations at the level of the wrapped model. For example, if a model takes “Location” latitude in degree-minute-second format instead of decimal degree, then an Adapter inside the Java implementation class makes the translation, while the slot API still uses the “Location” DataType with decimal degree contents.

For more complex data format differences, SUMMIT allows alternate DataTypes to appear on the link between slots. For instance, “Plume” contour data can be stored in SHP format or KML format (these are alternate GIS formats that cannot always be transformed into one another). SUMMIT might define “Plume-SHP” and “Plume-KML” DataTypes instead of the single “Plume”. The *Air Dispersion* slot could then have a single output labeled “Plume-SHP | Plume-KML”, meaning that either DataType is allowed. In this case, a model is compatible with the slot if it uses one or both output DataTypes. The process of selecting models to populate a simulation template becomes a little more complicated, because a compatible DataType must exist between two populated slots.

DISCOVERY PROCESS

The SUMMIT system will serve end users with different goals (planning, training, operations), from different perspectives (federal, regional, local), with different levels of experience. A mature SUMMIT system may broker hundreds of modeling and simulation tools with a broad range of capabilities. A basic problem is to connect users with relevant models. The previous sections described how emergency response scenarios are captured in simulation templates, and how models are abstracted by functional capability into template slots. This simplifies the end user process of discovery by dividing it into a sequence of decisions: (1) find a simulation template that matches the scenario of interest, (2) configure each slot in the template with a compatible model, and (3) provide specific inputs for execution of the template. Step (1) is called *discovery* and is the subject of this section.

SUMMIT will contain simulation templates that span a broad range of hazards (chemical, biological, nuclear, natural disaster, cyber incident, etc.) and outcomes of interest (casualties, economic impact, infrastructure damage, etc.). Users will have different interests and areas of expertise, and all aspects of the framework (templates and users) can be expected to evolve over time. To address this situation, SUMMIT uses "semantic ontology" technology to power a discovery process that presents the user with a series of "intelligent" queries. Semantic ontology has many interpretations in the computer science community. SUMMIT uses *taxonomy trees* captured with RDFS+ (Allemang, 2008).

A taxonomy tree defines a hierarchical organization of objects in the shape of an inverted tree, with a general "root" object at the top and more specific "branch" or "leaf" nodes beneath. A simple example is a family tree with a parent as the root and child branches. An example in SUMMIT is the organization of hazards into a tree; for instance, in Figure 3 a path is traced from *Hazard Incident* (the root node) through *Chemical Incident*, *CDC Taxonomy*, *Chemical Pulmonary Agent*, and *Chlorine* (a leaf node).

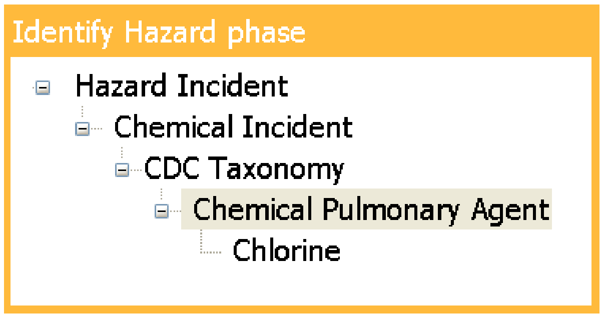


Figure 3. Example Taxonomy Tree

The user traverses a taxonomy tree in the discovery process to provide the right level of specificity. A tree is always most general at its root, and more specific at a leaf. For example, if the user is interested in any *Chemical Pulmonary Agent*, then the search stops at this level. Traversing further down a tree creates a more specific request.

The discovery process uses the taxonomy tree level as one of the search criteria for related simulation templates. When a user chooses a certain node in a tree, it indicates interest in templates matching the node or its children, and indicates that "sibling" items should be ruled out. In the example above, both *Chlorine* (shown) and *Phosgene* (not shown) are at the level below *Chemical Pulmonary Agent*. If the user chooses *Chemical Pulmonary Agent*, then simulation templates tagged for pulmonary agents, *Chlorine*, or *Phosgene* are matched. However, if *Chlorine* is chosen then simulation templates tagged for *Chlorine* are matched, but templates specific to *Phosgene* are not.

The Discovery process uses a collection of taxonomy trees organized into "phases". The first phase is *Hazard Incident*. The choice of hazards suggests other relevant questions to ask under subsequent phases such as the *Hazard Actions* phase. The suggestions are logical links, shown with arrows in Figure 4. The user is thus presented with a set of trees and chooses nodes of interest from each one. The Discovery Engine in the SUMMIT architecture (Figure 1) examines these choices and decides whether to continue with additional phases or to stop and return matching simulation templates.

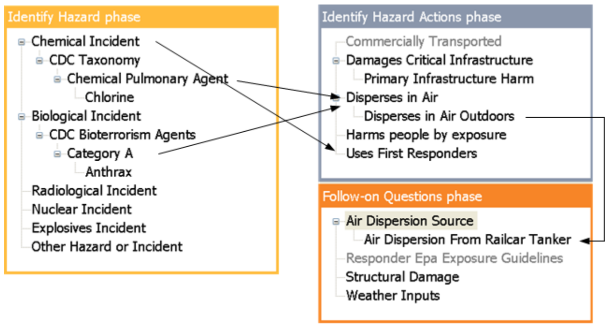
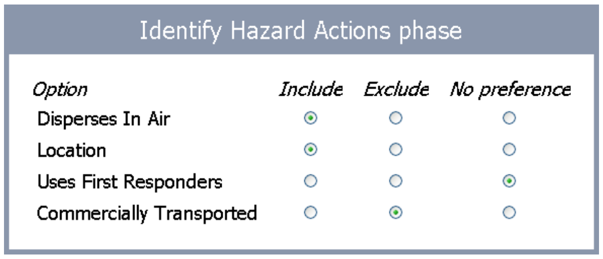


Figure 4. Linked Taxonomy Trees

All nodes chosen by the user contribute as search criteria for finding a simulation template. The SUMMIT discovery process also allows users to exclude a node. For example, Figure 5 shows a user choosing "Exclude" for the *Commercially Transported* tree, which means any simulation template containing a slot that models commercial transportation is ruled out. The figure also shows the user stating "No preference" for the tree *Uses First Responders*. This means the discovery process will not inquire further about first responders or use this as part of the search criteria for simulation templates.

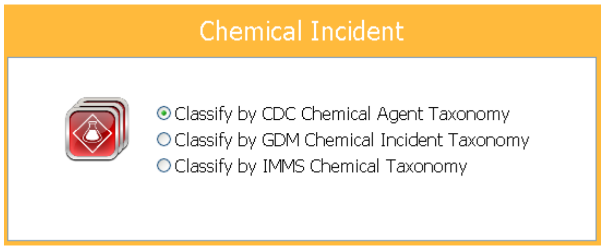


Identify Hazard Actions phase

Option	*Include*	*Exclude*	*No preference*
Disperses In Air	●	○	○
Location	●	○	○
Uses First Responders	○	○	●
Commercially Transported	○	●	○

Figure 5. Tree Preferences

The basic taxonomy tree idea is extended in SUMMIT to allow alternate views of the same tree. For example, *Chemical Incident* is a subtree that holds many specific chemical hazards: *Chlorine*, *Phosgene*, etc. The Center for Disease Control (CDC, 2010) provides a taxonomy that organizes chemicals according to their effects on the human body: *Pulmonary Agent*, *Blister Agent*, etc. The Geospatial Data Model (GDM, 2010) organizes the same chemicals by their physical properties. The IMMS project consulted with a chemical hazard subject matter expert who proposed an organization based on the types of models required: *Dispersed By Air*, *Spread By Contact*, etc. SUMMIT allows all three trees to exist at the same time. The user sees the three *Chemical Incident* groupings as shown in Figure 6, chooses the one they understand best, and follows that organizational tree in specifying the hazard of interest. In this way the discovery process accommodates multiple points of view.



Chemical Incident

● Classify by CDC Chemical Agent Taxonomy
○ Classify by GDM Chemical Incident Taxonomy
○ Classify by IMMS Chemical Taxonomy

Figure 6. Alternate Taxonomy Groupings

Summarizing, there are three main advantages with using a taxonomy tree ontology: (1) users are presented with a uniform interface, a series of taxonomy trees, that is easier to grasp than an unstructured set of questions, but still allows flexibility in the discovery process; (2) subject areas are encapsulated, because subject matter experts can define their knowledge in a tree with minimal dependence on other parts of the discovery knowledge base; (3) subject matter is logically arranged from simple, general descriptions down to detailed, specific leaf nodes.

Software Implementation

Taxonomy trees in SUMMIT are expressed as RDFS+ triples (Allemang, 2008). More powerful set restriction concepts modeled in OWL are not needed. The full set of trees is created and managed using the open source Protégé tool (Protégé, 2010). A collection of triples holds the SUMMIT ontology, but an inference engine is required to reason from this knowledge in response to user requests. SUMMIT uses *Jess*, an established product in the rules engine community licensed by Sandia National Laboratories (Friedman-Hill, 2003). *Jess* provides inferencing capability and has a powerful Java API that makes it easy to embed in Java applications.

Jess inferencing rules examine user choices from taxonomy trees and deduce the next set of relevant taxonomy trees. This is based on logical links as in Figure 4, and the location of those links with respect to the selected tree node.

When discovery is completed, *Jess* implements logic to return simulation templates that match user choices. Every slot in a simulation template is tagged with classes of the ontology. For example, the *Air Dispersion* slot of Figure 2 is tagged with *Disperses In Air* (there can be more than one tag per slot). If the user chooses *Disperses in Air* during the *Hazards Action* phase, then any simulation template containing an *Air Dispersion* slot is considered a match and "scores" one point. Each simulation template has a total possible score equal to the number of distinct tags on its constituent slots. Currently, the Discovery Engine returns all simulation templates with at least one match, ranked by the proportion of total possible matches that were realized.

SUMMARY

In this paper we provided an overview of the SUMMIT framework for integrating modeling and simulation tools for emergency response. We focused on two important features in SUMMIT: simulation templates, an abstraction for organizing models in terms of their relevance to emergency response scenarios, and the discovery process, which provides a broad range of end users with the capability to find relevant simulation templates.

The SUMMIT reference implementation is currently under development at Sandia National Laboratories as part of the IMMS program. A preliminary version of SUMMIT was deployed to support exercise planning in the FEMA National Level Exercise 2010, conducted May 17-18 in Washington, D.C. Threat, casualty, infrastructure, and medical surge models were integrated into SUMMIT to estimate health care resource requirements for the exercise ground truth. The models were contributed by different agencies, linked with special DataTypes, and executed on demand. SUMMIT has been designated as the integrating architecture for modeling and simulation at the FEMA National Level Exercise in 2011.

The SUMMIT framework and project information is hosted at http://dhs-summit.com.

ACKNOWLEDGEMENTS

This work was funded by the Infrastructure and Geophysical Division of the Department of Homeland Security's Science and Technology Directorate (DHS/S&T/IGD) and managed by the National Aeronautics and Space Administration (NASA).

REFERENCES

Allemang, D. & Hendler, J. (2008), *Semantic Web for the Working Ontologist*, Elsevier.

Beaulieu, B.R. (2004). "Security through The Palanterrs", *GeoIntelligence*, Aug 2004.

Case, M.P. (2008). "Fort Future – Ongoing Research", US Army Corps of Engineers, Engineer Research and Development Center. Retrieved May 2010 from http://www.erdc.usace.army.mil/pls/erdcpub/docs/erdc/images/ERDCFactSheet_Research_FortFuture.pdf.

CDC: Centers for Disease Control and Prevention, Chemical Emergencies (2010). Retrieved May 2010 from http://www.bt.cdc.gov/chemical.

Coolahan, K.E. (2007). "Planning for an Integrated M&S Framework for Catastrophic Event Response", in *Proceedings 2007 Spring Simulation Interoperability Workshop*, Norfolk, VA, Apr 2007. Also see http://pacer-ms-catalog.jhuapl.edu.

DHSST (2008). "High-Priority Technology Needs", Department of Homeland Security Science and Technology Directorate, Version 2.0, June 2008.

FAIT: Fast Analysis Infrastructure Tool (2010). Retrieved May 2010 from http://sandia.gov/nisac/fait.html.

GDM: Geospatial Data Model Version 2.7 (2010). Retrieved May 2010 from http://www.fgdc.gov.

EPA Support Center for Regulatory Atmospheric Modeling (2010). Retrieved May 2010 from http://www.epa.gov/scram001/dispersionindex.htm.

Friedman-Hill, E., (2003), *Jess in Action*, Greenwich, Manning Publications.

Friedman-Hill, E. Plantenga, T., & Ammerlahn, H. (2010). "Simulation Templates in the SUMMIT System", in *2010 SISO Spring Interoperability Workshop*, Orlando, FL, Apr 2010.

Google Protocol Buffers (2010). Retrieved May 2010 from http://code.google.com/apis/protocolbuffers.

iCAV: Integrated Common Analytical Viewer (2010). Retrieved May 2010 from https://icav.dhs.gov.

Jain, S. & McLean, C.R. (2006). "An Integrating Framework for Modeling and Simulation for Incident Management", *Journal of Homeland Security and Emergency Management*, *Vol 3*, 2006.

Kuhl, F., Weatherly, R., & Dahman, J. (1999). *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Upper Saddle River: Prentice-Hall.

McClean, C.R., Jain, S., Lee, Y.T., & Shao, G. (2007). "An Integrated Simulation Environment For Incident Management Training", National Institute of Standards and Technology, Gaithersburg, MD.

NIEM: National Information Exchange Model (2010). Retrieved May 2010 from http://www.niem.gov.

Pederson, P., Dudenhoeffer, D., Hartley, S., & Permann, M. (2006). "Critical Infrastructure Interdependency Modeling: A Survey of U.S. and International Research", INL/EXT-06-11464, Idaho National Laboratory, Idaho Falls, ID, Aug 2006.

Protégé: Stanford Center for Biomedical Informatics Research (2010). Retrieved May 2010 from http://protege.stanford.edu.

Simunich, K.L. (2005). "Dynamic Information Architecture System (DIAS): Developer's Guide", Argonne National Laboratory, ANL/DIS-0501, February 2005.