ON USING META-MODELING AND MULTI-MODELING TO ADDRESS COMPLEX PROBLEMS

by

Ahmed Abu Jbara
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____ Dr. Alexander H. Levis, Dissertation Director

_____ Dr. Larry Kerschberg, Committee Member

_____ Dr. Gabor Karsai, Committee Member

_____ Dr. Abbas Zaidi, Committee Member

_____ Dr. Sam Malek, Committee Member

_____ Dr. Stephen Nash, Senior Associate Dean

_____ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Summer Semester 2013
George Mason University
Fairfax, VA

On Using Meta-Modeling and Multi-Modeling to Address Complex Problems

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Ahmed Abu Jbara
Master of Science
George Mason University, 2006
Bachelor of Science
Islamic University - Gaza, 2002

Director: Alexander H. Levis, Professor
Department of Electrical and Computer Engineering

Summer Semester 2013
George Mason University
Fairfax, VA

DEDICATION

This is dedicated to my mother Seham, my wife Shrouq, my children and the rest of my lovely family. During the course of my graduate study they have all provided continuous support and love. Without their help and understanding I could never have finished my graduate studies.

ACKNOWLEDGEMENTS

I would like to thank my dissertation director, Dr. Alexander H. Levis, for giving me the opportunity to work under his supervision. During my graduate studies Dr. Levis has been of great help and provided continuous support and mentorship.

I would also like to thank the members of my committee, Dr. Larry Kerschberg, Dr. Gabor Karsai, Dr. Abbas Zaidi and Dr. Sam Malek for their guidance throughout my Ph.D. studies.

Last but not least I would like to thank the members of the System Architectures Laboratory for the great experience I had working with them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BPEL ..... Business Process Execution Language

BPMN ..... Business Process Modeling and Notation

C2WT ..... Command and Control Wind Tunnel

COA ..... Courses of Action

DSML ..... Domain Specific Modeling Language

DSMWL ..... Domain Specific Multi-Modeling Workflow Language

GIS ..... Geographic Information Systems

GME ..... Generic Modeling Environment

GPL ..... Generic Purpose Language

HLA ..... High Level Architecture

JADE ..... Java Agent DEvelopment Framework

JIATF-South ..... Joint Interagency Task Force - South

OWL ..... Web Ontology Language

SME ..... Subject Matter Expert

SOA ..... Service Oriented Architecture

SORASCS ..... Service Oriented Architectures for Socio-Cultural Systems

SPARQL ..... Simple Protocol And RDF Query Language

ABSTRACT

ON USING META-MODELING AND MULTI-MODELING TO ADDRESS COMPLEX PROBLEMS

Ahmed Abu Jbara, Ph.D.

George Mason University, 2013

Dissertation Director: Dr. Alexander H. Levis

Models, created using different modeling techniques, usually serve different purposes and provide unique insights. While each modeling technique might be capable of answering specific questions, complex problems require multiple models interoperating to complement/supplement each other; we call this Multi-Modeling. To address the syntactic and semantic challenges of this multi-modeling approach for solving complex problems, a systematic methodology for developing multi-modeling workflows is presented. The approach is domain specific: Identification of the domain and the supporting modeling techniques is the first step. Then a Domain Specific Multi-Modeling Workflow Language (DSMWL), supported by a Domain Ontology, is developed and then used to construct workflows that capture interoperations between various models. The domain ontology provides semantic guidance to effect valid model interoperation. The approach is illustrated using a case study from the Drug Interdiction and Intelligence

domain. The Joint Inter-Agency Task Force (JIATF) - South, an agency well known for interagency cooperation and intelligence fusion, receives large amounts of disparate data regarding drug smuggling efforts. Analysis of such data using various modeling techniques is essential in identifying best Courses of Action (COAs). The proposed methodology is applied to the Drug Interdiction domain by performing domain analysis, developing a Domain Specific Multi-Modeling Workflow Language (DSMWL) and a Domain Ontology, and then using the DSMWL and the Domain Ontology to create workflows of model interoperations involving Social Networks, Timed Influence Nets, and Geospatial models.

CHAPTER 1: INTRODUCTION

Traditionally, Modeling and Simulation (M&S) systems were designed with the assumption that a single type of model would be developed and analyzed. A model in such an environment is developed using some known modeling techniques to address a certain class of problems. While single modeling techniques might be capable of answering specific questions, solving complex problems usually requires multiple models interoperating together (Multi-Modeling). The move towards supporting multi-modeling in various modeling and simulation platforms is already taking place. The Command and Control Wind Tunnel (C2WT) [1] developed by Vanderbilt University and the Service Oriented Architecture for Socio-Cultural Systems (SORASCS) [2] developed by Carnegie Mellon University are examples of Multi-Modeling capable platforms. The first provides a federated approach, utilizing the High Level Architecture (HLA) [3] standard and the meta-programmable Generic Modeling Environment (GME) [4]; the second employs Service Oriented Architecture (SOA) techniques in providing model interoperation capabilities.

In achieving Multi-Modeling and to provide powerful supporting platforms, many challenges have to be faced. Besides the technical issues that usually arise in allowing interoperations between models through their modeling tools, there is also a major

challenge of improving the human interface to the Multi-Modeling process itself [5]. This includes addressing both syntactic and semantic aspects of interoperation.

In this dissertation, a systematic methodology for addressing both syntactic and semantic issues in developing multi-modeling workflows to solve complex problems is presented. The focus of our approach is on helping users of multi-modeling platforms in designing workflows of multi-modeling activities that guarantee both syntactic and semantic correctness of the interoperations across models. Our approach is domain specific; the rationale behind this is twofold: first, problems to be solved by employing multi-modeling techniques are usually domain specific themselves; second, it narrows down the scope of meaningful interoperations among several modeling techniques where each technique offers unique insights and makes specific assumptions about the domain being modeled. We begin with the identification and characterization of a domain of interest and its supporting modeling techniques. A Domain Analysis (DA) step follows aiming to provide formal representations of syntactic and semantic aspects of the domain. A new Domain Specific Multi-Modeling Workflow Language (DSMWL) is then developed to construct workflows that capture multi-modeling activities in the selected domain. A Domain Ontology resulting from the Domain Analysis step is utilized to provide semantic guidance that effects valid model interoperation. Resulting workflows that represent multi-modeling activities and solve a specific problem in the domain of interest are then transformed into formats that multi-modeling capable platforms can implement.

Providing multi-modeling capabilities in Modeling & Simulation platforms can be achieved by employing different technologies and techniques. The SOA based approach, as in the SORASCS platform, is one approach in which concepts of loosely coupled, standards-based, and protocol independent services are utilized to allow for interoperations between models and integration of various modeling techniques. Given the fact that a service is the basic building block of any SOA based system, features and capabilities of participating modeling techniques should be available as services. Exposing features and capabilities of modeling techniques as services is not an easy task especially when dealing with legacy tools and techniques, however, once required services exist, the development of a SOA based platform can be achieved.

A major focus of our approach is to provide means for automating the multi-modeling process. The use of a formal Domain Specific Multi-Modeling Workflow Language (DSMWL) supported by a Domain Ontology to capture multi-modeling activities provides some means of automation. Value can be added to this approach with the use of advanced automation techniques like Intelligent and Semantic Agents. Intelligent Agents are entities that act on behalf of human users and require only enough knowledge to solve specific complex problems [6]. Our approach can be extended so that intelligent agents that represent Subject Matter Experts (SMEs) can reason about solving problems in a specific domain, select appropriate modeling techniques, and construct semantically correct multi-modeling workflows using available modeling-techniques' services. Each one of the represented SMEs is basically an expert in using a specific modeling-technique. Implementing such vision would require that multi-modeling

capable platforms include an implementation of a Multi-Agent component like the Java Agent DEvelopment Framework (JADE) [7].

Our approach is illustrated in an application from the Drug Interdiction and Intelligence domain. The Joint Interagency Task Force - South (JIATF-South), an agency well known for interagency cooperation and intelligence fusion [8], receives large amounts of disparate data regarding drug smuggling activities. Analysis of such data using various modeling techniques is essential in identifying best Courses of Action (COAs). We apply our methodology to solve a class of problems in this domain by creating workflows of model interoperations involving Social Networks, Timed Influence Nets, Organization Structures, and Geospatial models.

1.1 Motivation

The main purpose of the research is to develop a methodology for creating multi-modeling workflows that allows for capturing multi-modeling activities in a formal way. The methodology involves the development of a Domain Specific Multi-Modeling Workflow Language (DSMWL) and a supporting Domain Ontology. The domain ontology is used to guide the creation of semantically correct workflows. Users of the proposed approach should be capable of visually designing workflows that address complex modeling problems by employing multiple interoperating models.

1.2 Motivating Example

The use of Modeling & Simulation to allow for better understanding of various application domains has been widely adopted. An example of such is the decision making

process in drug interdiction activities performed by the Joint Interagency Task Force - South (JIATF-South) [8]. The agency usually receives diverse types of data regarding drug trafficking and drug cartels from a wide variety of sources. Quick and accurate analysis of data is essential in addressing drug trafficking threats effectively. The agents working at the agency to analyze the data and to support the decision making process by providing feasible Courses of Action (COAs) use various modeling techniques. Traditionally, they create and analyze separate models including Social Networks, Geographic Information Systems (GIS) models and Influence Nets. While each model provides certain insights, employing multi-modeling into the analysis process allows these models to complement each other and therefore provides a more comprehensive and effective analysis. Figure 1 represents an overview of how a number of modeling techniques can interoperate to support the decision making at the JIATF-South.

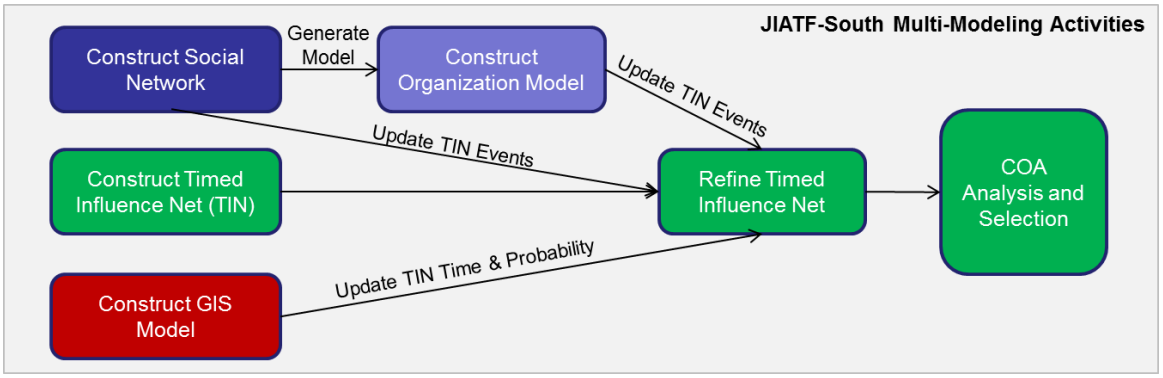

Figure 1: Motivating Example

The use of the methodology presented in this dissertation should help the JIATF-South agents in capturing multi-modeling activities using a Domain specific Multi-Modeling Workflow Language (DSMWL) and a Domain Ontology developed for the drug interdiction application domain. Following the proposed methodology to develop and use the DSMWL and the Domain Ontology is an iterative process in which the JIATF-South agents need help from experts in developing domain specific languages and multi-modeling platforms in general. Once this is accomplished, the agents can perform multi-modeling based analysis by first creating multi-modeling workflows and then implementing these workflows on appropriate platforms. Figure 2 shows the sequence of activities required in applying the methodology to the JIATF-South example and the major roles associated with this.

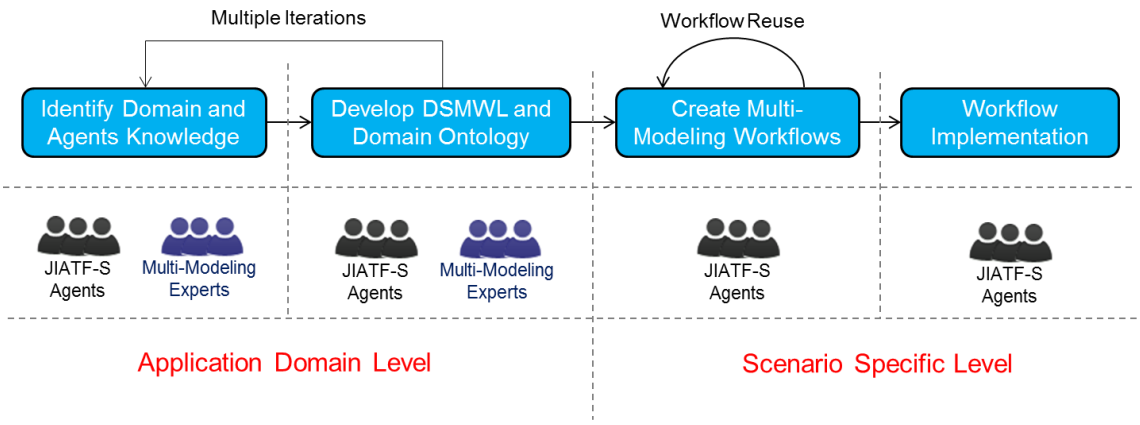

Figure 2: Application of Methodology and Associated Roles

In general, each agent at the JIATF-South is specialized in a specific area, knows part of the problem and uses a set of modeling techniques. The process of implementing the methodology requires that all agents and supporting experts come together to share knowledge and develop the DSMWL and the Domain Ontology. This form of cooperation helps in identifying the relationships between various modeling techniques used in the application domain. A side product to such effort is the increase level of understanding of the overall analysis process by each of the stockholders.

The motivating example explained briefly in this section will be expanded into a full case study in Chapters Seven and Eight.

1.3 Problem Statement

Given a domain specific problem that requires the use of multiple interoperating models created using various modeling techniques, create a multi-modeling workflow (and its Meta-Model) that is syntactically and semantically correct with reference to the domain of interest and the problem being addressed.

An expert modeler needs to have the capability of creating high-level multi-modeling workflows that can capture the interoperations between interconnected models. These interoperations between the models have to be both syntactically and semantically correct with respect to the domain and situation of interest. Creating such workflows requires an appropriate workflow design platform and an environment that is capable of implementing this workflow. Since problems to be solved by employing multi-modeling techniques are usually domain specific themselves, and to narrow down the scope of

meaningful interoperations among several modeling techniques, a domain specific approach is preferable to a generic one. Similarly, a domain specific multi-modeling workflow language is preferable to a Generic Purpose Language (GPL) that lacks modeling attributes required by the specific domain. Creating a meta-model of such a domain specific multi-modeling workflow language is a major task that precedes the use of the language. The meta-model of the language defines the language's constructs and rules. Furthermore, the domain specific multi-modeling workflow language should take into account the semantic aspects of the domain. A Domain Ontology is a perfect fit for capturing the semantic aspects of the domain; however, there is still a gap in terms of providing the domain specific multi-modeling workflow language with the capability of utilizing the Domain Ontology in order to guide the workflow creation process.

1.4 Research Statement and Hypotheses

This research aims to provide a systematic methodology for developing and using a domain specific multi-modeling workflow. The methodology includes the utilization of a domain ontology to guide the use of the workflow language in creating semantically correct workflows. The proposed domain specific language can be used to visually design workflows of multi-modeling activities. Once a workflow for a specific application domain is created, it can be implemented on a platform that provides multi-modeling capabilities.

Hypothesis #1: A Domain Specific Multi-Modeling Workflow Language can be developed using a systematic methodology.

Hypothesis #2: A Domain Ontology can be utilized to guide the creation of semantically correct multi-modeling workflows using a Domain Specific Multi-Modeling Workflow Language.

1.5 Research Questions

- How can we create a domain specific multi-modeling workflow?
- What are the elements of a domain specific multi-modeling workflow language?
- How can we create a domain specific modeling language that is semantically guided?
- How can a multi-modeling workflow guarantee the semantic correctness of model interoperations?

1.6 Contributions

In this dissertation, our approach of developing and using a domain specific multi-modeling workflow language provides contributions to the multi-modeling research area. It provides a systematic methodology for developing domain specific multi-modeling workflows that can be used to visually create workflows of multi-modeling activities. The methodology includes the use of a domain ontology for semantic guidance. The following lists the key contributions of this research effort:

1. A five step methodology for developing and using a domain specific multi-modeling workflow language. This includes the use of the existing Generic Modeling Environment (GME) to design and use such language.

2. A detailed methodology for using a domain ontology to semantically guide the use of the workflow language.

3. Outlining a process for extending the GME to allow for ontology support. This includes the use of an ontology query language and an ontology query engine to query a domain specific ontology associated with a domain specific multi-modeling workflow language.

4. A domain specific multi-modeling workflow language for the selected case study.

5. A domain specific ontology used to semantically guide the domain specific multi-modeling workflow language for the selected case study.

6. A GME workflow interpreter that interprets workflows to the format required for the selected multi-modeling platform.

1.7 Organization of the Dissertation

The rest of the dissertation is organized as follows: Background and related work are presented in Chapter two. Chapter three presents an overview of our approach and discusses the five major steps of our systematic methodology for developing and using domain specific multi-Modeling workflows. Chapter four describes in detail the process of defining a domain specific multi-modeling workflow language. In Chapter five the semantic guidance aspect of the approach is presented and discussed. The implementation of a multi-modeling workflow using a multi-modeling capable platform is discussed in Chapter six. Chapters seven and eight present an application of our approach to a case

study in the Drug Interdiction and Intelligence domain. Chapter nine discusses conclusions, limitations and directions for future research.

CHAPTER TWO: BACKGROUND AND RELATED WORK

Model interoperation has been addressed repeatedly, and from different perspectives, in the Modeling and Simulation research community. A variety of approaches and techniques have been proposed over the past decade [1][2][5]. We begin this chapter by presenting some corner-stone concepts and then proceed with discussing some related work. This includes concepts and techniques that explain the basics of modeling and multi-modeling in addition to concepts essential for our new approach.

2.1 Background

In section 2.1.1 the general concept of modeling is discussed. The rationale behind multi-modeling and its basics are presented in section 2.1.2. Sections 2.1.3 and 2.1.4 discuss meta-models hierarchy, meta-modeling and their relation to multi-modeling. The multi-modeling workflow concept and mapping it to the meta-modeling hierarchy is explained in section 2.1.5. In sections 2.1.6 and 2.1.7 a brief discussion about ontologies and the importance of addressing semantic relationships in multi-modeling workflows are presented. Section 2.1.8 includes a discussion on the history of domain identification techniques and their relation to multi-modeling.

2.1.1 Modeling

Modeling and Simulation helps to develop a level of understanding of the interactions of parts of a specific situation, or a situation as a whole. A simulation is

usually a software implementation of a model that runs over time to study the behavior of the modeled system [9].

Modeling is the process of producing a model; a model is a representation of the construction and working of some situation of interest. A model is similar to, but simpler than, the situation it represents. One purpose of a model is to enable the analyst to predict the effect of changes to the situation. On one hand, a model should be a close representation to the real system. On the other hand, it should not be so complex that it is impossible to understand and conduct an experiment. Generally, a model intended for an analysis or simulation study is a mathematical model developed with the help of a modeling tool and conforms to a specific modeling language. [9]

Figure 3 represents the Modeling Hierarchy in which a modeling language is used to create a model that represents a specific situation of interest. A modeling language generally consists of syntax, symbols, semantics and a procedure that determines how to use the language to create models. A model is created using a modeling tool that implements a modeling language to represent a specific situation. In the context of our research we refer to the combination of a modeling language, a modeling tool, and the procedures applied to create models as a Modeling Technique.

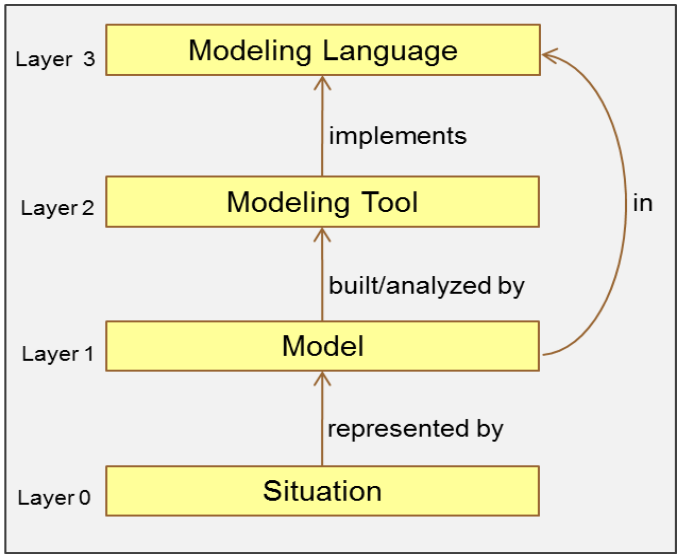

Figure 3 Modeling Hierarchy

2.1.2 Multi-Modeling

No single model can capture the complexities of any domain of interest. Each Modeling Language provides certain capabilities and makes specific assumptions about the domain being modeled. For example, Timed Influence Net [10] models describe cause and effect relationships among groups at a high level but have no capability of capturing social aspects between the groups of interest. Social Networks [11], on the other hand can describe the interactions among groups and members of the groups. In this context, a multi-modeling approach addresses a complex problem through the use of a number of interconnected domain-specific models where each model contributes insights to the overall problem. The interoperations between the interconnected models could serve different purposes and can happen in various forms. [12]

- A model x uses another model y, where each model is constructed using a different modeling language, to complete a computational or analysis task.
- Two or more models run concurrently and supply a complementary part of a solution.
- A model x supplements computational or analysis tasks of another model y with results and/or parameter values.

2.1.3 Meta-Models

A meta-model is an abstraction layer above the actual model and it describes the modeling language used to create the model; the model conforms to its meta-model. The meta-model itself conforms to a higher meta-model (meta^2-model) which describes the meta modeling language as described in Figure 4

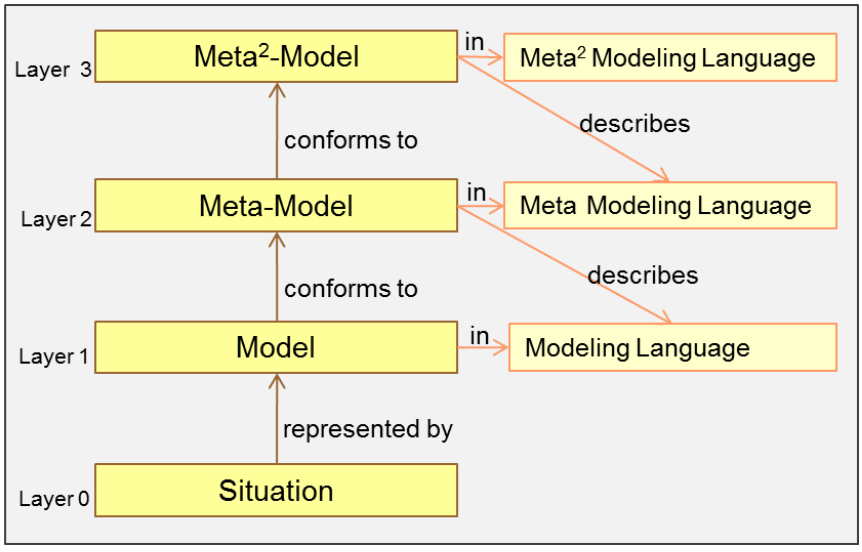

Figure 4: Meta-Models Hierarchy

The typical role of a meta-model is to define how model elements are instantiated. Consider an example of a meta-model for a simplistic Finite State Machine (FSM) as shown in Figure 5. Layer 2 shows the meta-model of the FSM modeled using the UML notation. Layer 1 represents a FSM Model that conforms to the Meta-Model in Layer 2 and that captures a specific situation in Layer 0. [13]

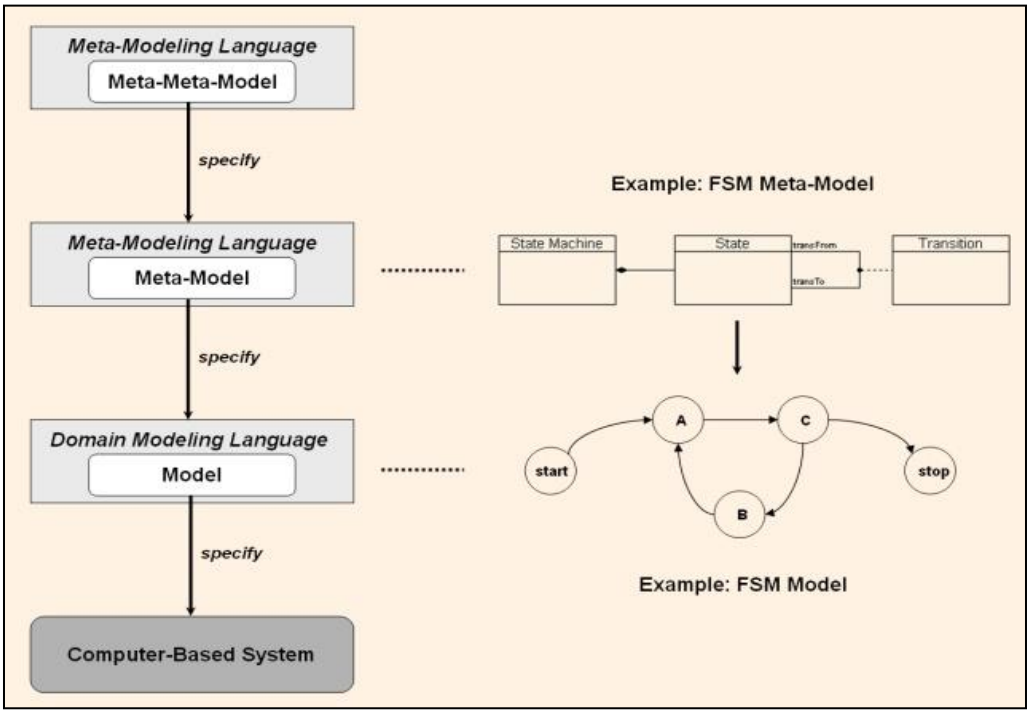

Figure 5: FSM Meta-Model [13]

2.1.4 Meta-Modeling

In general, meta-modeling is defined to be the process of constructing a meta-model in order to model a specific problem within a certain domain. In the context of this multi-modeling research effort, the meta-modeling concept is extended to include the

analysis of the conceptual foundations of a model ensemble so that individual models, constructed to address a specific problem in a domain of interest, can interoperate as part of a workflow developed to address a specific problem [12]. Meta-modeling then becomes a process of constructing a meta-model which guarantees that the models of a multi-modeling activity can interoperate correctly with respect to their meta-models.

In [12] different types of meta-modeling operations for performing multi-modeling have been identified. So far, this effort has resulted in identifying the following operations:

- Concatenation: models share representations and can get instances from each other.
- Amplification: model adds or augments class representation from another.
- Parameter Discovery: one model provides parameters for algorithms to another model's method.
- Model Construction: one model is used to construct models of another type.

2.1.5 Multi-Modeling Workflows

Four layers need to be addressed in order to achieve Multi-Modeling as shown in Figure 6 [14]. The first layer, Physical, i.e., Hardware and Software, is a platform that enables the concurrent execution of multiple models expressed in different modeling languages and provides the ability to exchange data and also to schedule the events across the different models. The second layer is the syntactic layer which ascertains that the right data are exchanged among the models. The Physical and Syntactic layers have been

addressed through the development of some environments like the Command & Control Wind Tunnel (C2WT) [1] by Vanderbilt University and the Service Oriented Architecture for Socio-Cultural Systems (SORASCS) [2] developed by CASOS at Carnegie Mellon University.

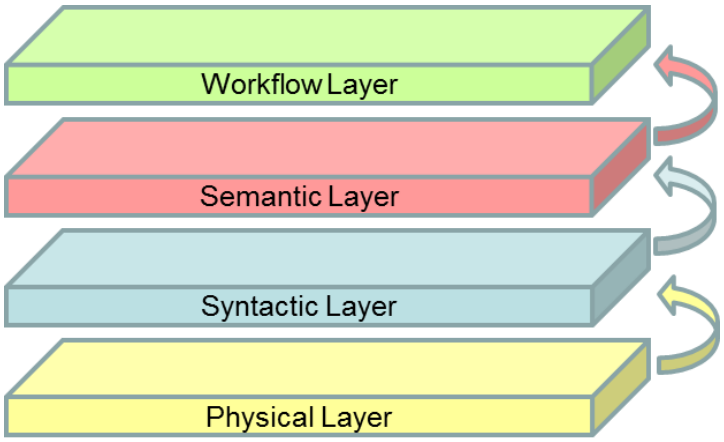

Figure 6: The Four Layers of Multi-Modeling [14]

A third problem needs to be addressed at the Semantic layer, where the interoperation of different models is examined to ensure that conflicting assumptions in different modeling languages are recognized and form constraints to the exchange of data. In the Workflow layer, valid combinations of interoperating models are considered to address specific issues. Different issues require different workflows [14].

In the multi-modeling workflow level, analysts can design a workflow of a multi-modeling activity (Interoperation). A multi-modeling workflow should be able to capture

the different analysis activities associated with each model and the interconnections between models. A multi-modeling workflow is itself a model of an analysis process.

A formal approach to capture a multi-modeling workflow requires a formal modeling language with its own rules. Developing workflows using such an approach allows for translating visual views of model interoperation into an executable implementation. There already exist generic techniques for designing and implementing workflows such as Business Process Model and Notation (BPMN) [15] and Business Process Execution Language (BPEL) [16]. A domain-specific approach requires a domain specific multi-modeling workflow language for the selected domain of interest. Such a language would be tailored to a problem domain and would offer a high level of expressiveness and ease of use compared with a General Purpose Language (GPL) [17] and can be a specific profile of an existing GPL, i.e., BPMN. Figure 7 shows the mapping between a Domain Specific Multi-Modeling Workflow Language (and its Meta-Model) to the Meta-Models Hierarchy.

Defining the meta-model of the workflow language in Layer 2 of Figure 7 is a meta-modeling process itself. To capture those constructs of the meta-model that define the new language, a meta-modeling language that conforms to a higher meta-model, meta^2-model, is also required. The research community in this area has addressed such hierarchy from different perspectives and many approaches were developed. One of these approaches is the Generic Modeling Environment (GME) [4], a configurable toolkit for creating domain-specific languages and program synthesis environments, developed by

Vanderbilt University. The configuration is accomplished through meta-models specifying the modeling paradigm (modeling language) of the application domain.

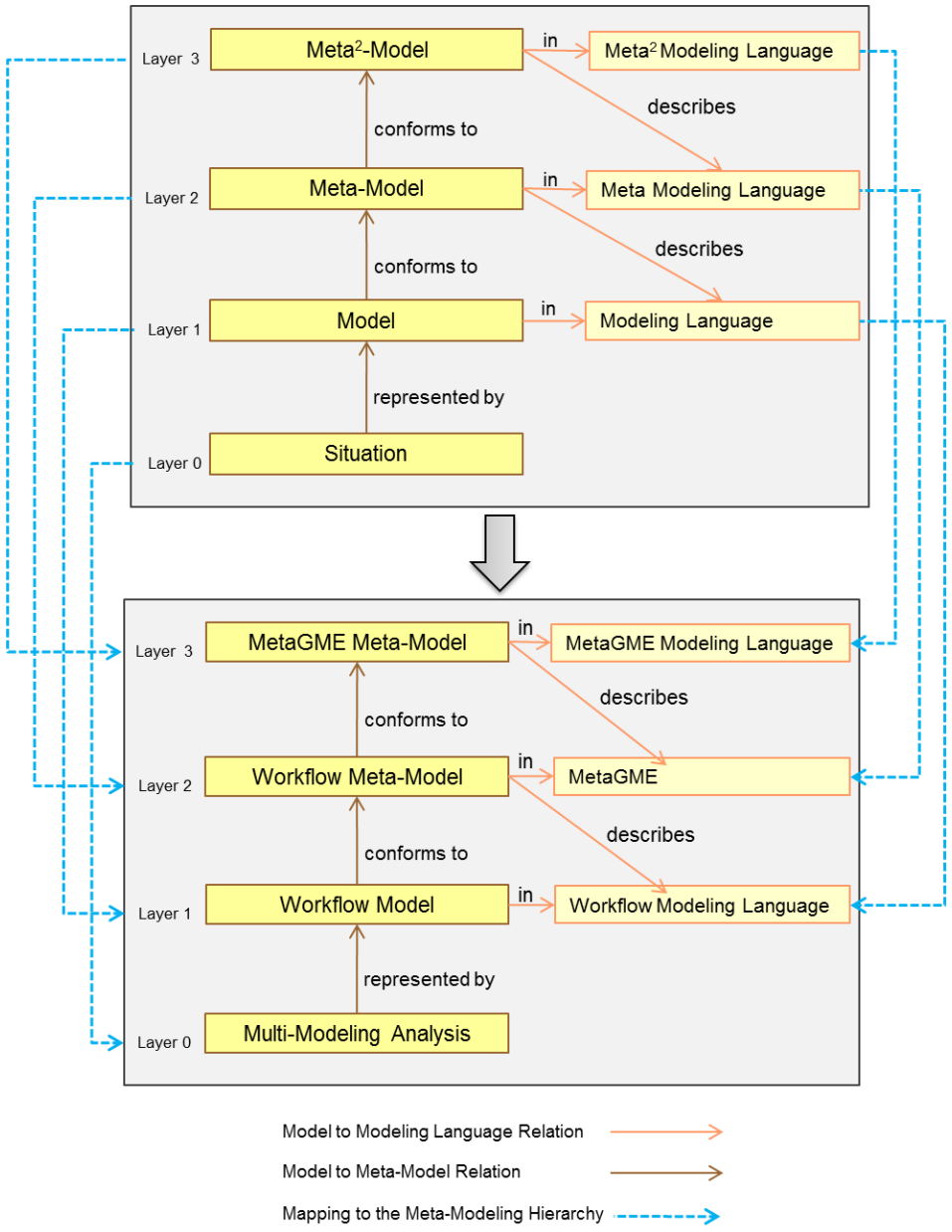

Figure 7: Mapping to the Meta-Models Hierarchy

The modeling paradigm contains all the syntactic and presentation information regarding the domain including the concepts used to construct models, relationships between concepts, different views and organizations of the concepts, and rules governing the modeling process. Defining the modeling paradigm is a modeling activity itself. GME has a meta-modeling paradigm that configures the environment for creating meta-models of the domain of interest. These models of the meta-models are then automatically translated into GME configuration through model interpretation. The meta-modeling paradigm is based on the Unified Modeling Language (UML).

2.1.6 Workflow Languages

In general, workflow languages aim at capturing workflow-relevant information of application processes with the intent of executing them by a workflow management system. They are considered as another species of languages for human computer interaction. In contrast to other General Purpose Languages (GPLs), workflow languages are highly domain specific. Workflow languages are used to specify workflow models which represent processes to be used by a workflow management system. [18]

Workflows can be considered from different aspects as presented in [18]. A description of each of these aspects is as follows:

- Functional Aspect: The functional aspect covers the functional decomposition of activities; it specifies which activities have to be executed within a workflow. To deal with high levels of complexity, the concept of nesting is used. In particular, workflows are composed of a number of complex or atomic workflows. Due to their

relative position, the components of a complex workflow are known as sub workflows.

- Behavioral Aspect: Workflows consist of a set of interrelated activities. Hence, the execution of a workflow has to take into account the complex interrelationships of workflows and sub workflows. This aspect specifies under which conditions the activities of a workflow and its sub workflows are executed during workflow execution. Important components of this aspect are control flow constraints.

- Informational Aspect: An important aspect of a workflow language is the information aspect. It allows for capturing the transfer of data as generated or processed by workflow activities. The transfer of data between workflow activities is known as data flow.

- Organizational Aspect: Information on the organization and the technical environment in which the workflow is to be implemented has to be provided. In general, workflow activities can be automatic or manual. Manual activities are handled by humans who may use application programs. Automatic activities are executed by systems. Since a strict assignment of an activity to humans is not always feasible, the role concept is used.

- Operational Aspect: The integration of existing tools into a workflow through activities is an important feature. The information required for such integration is specified in the operational aspect.

- Flexibility Aspect: Providing flexibility to workflows is based on the understanding that during workflow modeling not all aspects can be specified completely. There may be unforeseen situations which require flexible reactions by the workflow modeler or user.

Workflow languages can be classified according to their underlying methodologies and meta-models. A workflow language meta-model describes the constructs and their relationships for any workflow model of a particular workflow language. An important class of workflow languages is graph-based languages that allow the specification of workflows using different forms of directed graphs. It is possible to specify functional and behavioral aspects using graph notation. However, the informational and operational aspects require additional specifications.

Another class of workflow languages is the set of Net-based languages such as Petri nets. They are more widely used to specify the behavior of a dynamic system with a fixed structure. Besides these classes of workflow languages, script languages are widely used. Often, these languages are closely related to workflow management system development. Workflow languages can also have multiple representations. For instance, there may be a graphical language for the specification of workflow models, which is translated into a script language, to be processed by a workflow management system. [18]

The Business Process Modeling and Notation (BPMN), developed by the Object Management Group, is one example of a graph-based business process "workflow"

generic language. It provides a notation that is readily understandable by a wide spectrum of users, from the analysts that create initial drafts of workflow models, to the technical developers responsible for developing the technology that will implement these workflows. [19] In BPMN, workflow business process models are expressed in business process diagrams. Each diagram consists of a set of modeling elements. These elements are partitioned into a core element set and a complete element set. The core element set allows expressing simple structures in business processes, while expressive power is added by the complete element set. The BPMN has the flavor of a framework more than of a concrete language, because some topics are disregarded and left to the process designer or the designer of the modeling tool. The notional elements in business process diagrams are divided into four categories as shown in Figure 8. [18]

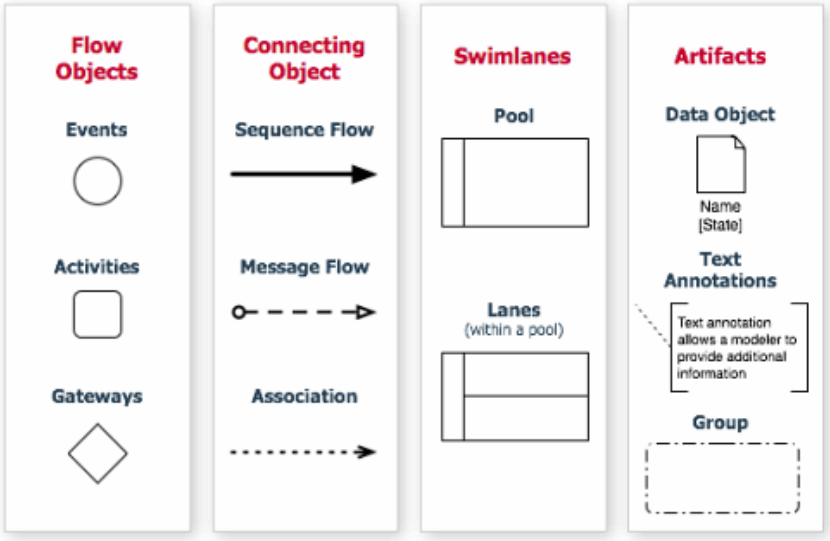

Figure 8: Core Set of BPMN Elements [18]

Flow objects are the building blocks of BPMN diagrams; they include events, activities and gateways. States in the real world are represented by events. Activities represent work performed during the process. Gateways are used to represent the splits and joints of the flow of control between flow objects. Organizational aspects are represented by swim lanes which are restricted to a two-level hierarchy: pools and lanes. Artifacts are used to show additional information about a process that is not relevant for sequence flow. Data objects are supported as artifacts. Connecting objects connect flow objects, swim lanes or artifacts. Sequence flow is used to specify the ordering of flow objects, while message flow describes the flow of messages. Associations are used to link artifacts to other elements. The core element set provides a small set of concepts and its graphical representation for modeling processes. The diagram shown in

Figure 9 represents a process of analyzing data for decision making purposes using supporting modeling techniques. [18]

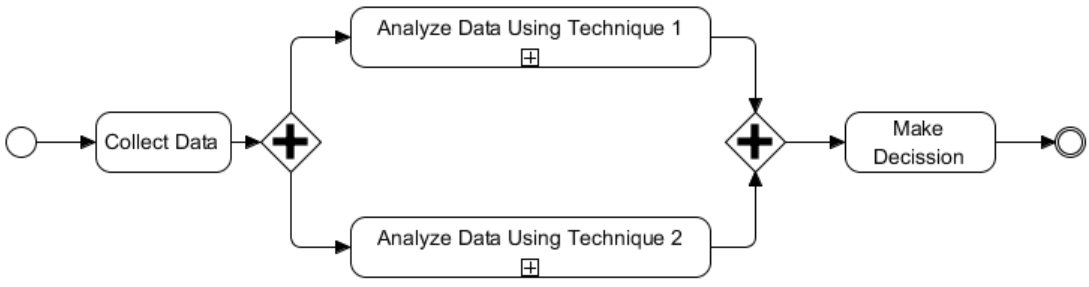

Figure 9: BPMN Example

2.1.7 Ontologies

Ontology is the term used to refer to the shared understanding of a domain of interest. It entails some sort of a global view of a specific domain. This view is conceived as a set of concepts, their definitions and their inter-relationships; this view is referred to as conceptualization. Ontologies are usually intended to be used for different purposes. Some purposes reflect various interpretations of the word 'Ontology' such as a meta-level specification of a logical theory. In computer systems ontologies can be thought of as a means to structure a knowledge base. [20]

In reference to the Meta-Modeling concept addressed in this research, an ontology can be used to guide the interoperation between models based on their meta-models. A multi-modeling workflow language that conforms to a higher meta-model needs a mechanism to guarantee the semantic correctness of model interoperation. A domain specific ontology can serve this purpose.

Over the last two decades, a huge number of ontologies representing different domains and on different levels of abstraction have been created. On the other hand, the trend of systems integration is continuously challenged by the fact of not having ontologies that can capture the interconnections between domains. This motivates the need for Upper Ontologies that can facilitate the definition of interconnections between domains. In [21] an upper ontology, the Suggested Upper Merged Ontology (SUMO), has been proposed as a starter document for a Standard Upper Ontology Working Group. The SUMO defines general purpose terms and acts as a foundation for more specific

domain ontologies. The SUMO was created by merging publicly available ontological content into a single, comprehensive, and cohesive structure. The knowledge representation language for SUMO is a version of KIF (Knowledge Interchange Format) called SUO-KIF. The merging of the collected and identified ontological content occurred in two steps. A syntactic merge, in which the ontological content was translated to SUO-KIF, was followed by a difficult semantic merge. In the semantic merge step the ontologies were first divided into two classes, those defining very high level concepts and those defining lower level notions. The first class ontologies were merged easily since they were compact and contained a significant amount of overlapping content. This upper level was then used as the foundation for aligning all the content from the lower level class.

Ontology matching on the other hand is the process that produces a set of semantic correspondences between entities of two distinct ontologies [22]. Ontology matching is a critical operation in many domains. Many approaches and solutions for the ontological matching problem have been proposed throughout the research community. These approaches range from manual to automatic and semi-automatic in between. In [23] a set of algorithms is described that exploits upper ontologies as semantic bridges in the ontology matching process. In [24] a classification of Ontology "Alignment" mapping techniques has been presented.

2.1.8 Domain Identification

In general applying a multi-modeling approach for solving complex problems is domain specific; and hence, domain characterization becomes an essential task to be conducted prior to any other activity.

In the context of software and systems engineering, a domain is most often understood as an applications area, a field for which systems are developed [25]. It is also defined to be a class of problems, where the types of problems to be solved and the context in which the system elements can be used are clearly identified [26]. In the context of our research, we consider a domain to be a specific class of problems to be solved using a set of modeling techniques and appropriate data.

The domain identification process itself has been approached in many research efforts, especially the research on software reusability in late 80's and early 90's. In [26] a comparison of Domain Analysis (DA) approaches for software reuse purposes was presented. Domain identification was pointed out as a first and essential step prior to any DA activities. Domain identification methods in those approaches include informal description in the form of statements, use of object oriented techniques, employing classification schemes, determining domain boundaries and collecting examples of similar problems.

2.2 Related Work

In this section we discuss related work. A number of multi-modeling platforms is presented in 2.2.1. A related research on Meta-Modeling for Multi-Modeling is discussed

in 2.2.2. In 2.2.3 an approach for using Ontologies and Meta-Models to address model interoperability is presented.

2.2.1 Multi-Modeling Platforms

There exist a number of environments that support multi-modeling based Modeling and Simulation. Among those environments, three are of special interest to this research effort: the Command & Control Wind Tunnel (C2WT) [1] developed by Vanderbilt University, the Service Oriented Architecture for Socio-Cultural Systems (SORASCS) [2] developed by Carnegie Mellon University and NAOMI [27] developed by University of California - Berkeley.

2.2.1.1 Command & Control Wind Tunnel (C2WT)

In [1], an environment for designing and deploying heterogeneous Command and Control (C2) simulation federations, the Command and Control Wind Tunnel (C2WT), is presented. Its primary contribution is to facilitate the rapid development of 'integration models', and to utilize these models throughout the lifecycle of the simulated environment. An integration model defines all the interactions between federated models and captures other design intent, such as simulation engine-specific parameters and deployment information. This information can be leveraged to streamline and automate significant portions of the simulation lifecycle. The C2WT uses a discrete event model of computation as the common framework for the precise integration of an extensible range of simulation engines, using the Run-Time Infrastructure (RTI) of the High Level Architecture (HLA) platform. The C2WT offers a solution for Multi-Model simulation by

decomposing the problem into model integration and experiment or simulation integration tasks. Figure 10 shows the overall architecture of the C2WT.

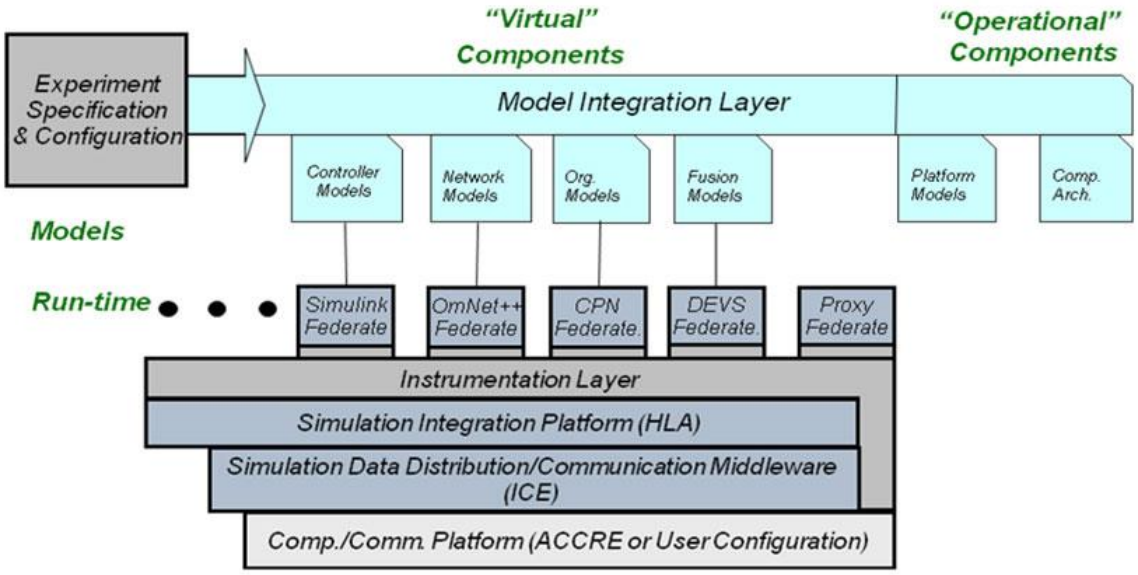

Figure 10: The C2WT Architecture [1]

2.2.1.2 Service Oriented Architecture for Socio-Cultural Systems (SORASCS)

The Center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon University utilized a Service Oriented Architecture (SOA) to build a Modeling and Simulation environment, called SORASCS [2]. SORASCS provides architecture for socio-cultural modeling and analysis to support the socio-cultural modeling and simulation community. The basis of the architecture is the use of a multi-layered system capturing the essential flows of information and providing support for flexible orchestration, coordination, and transformation.

Figure 11 shows the four layers of the SORASCS architecture. The first layer represents the data layer where a set of heterogeneous data sources form the raw inputs to the system. In the second layer, high level models represent information extracted from the first layer. A collection of analysis tools resides in the third layer. These tools interact with models in the second layer to generate input for and analyze results from the tools in the fourth layer, which is the end-user layer. In addition to the capability of presenting analysis and simulation output from the lower layer, the user layer is capable as well of supporting orchestration, allowing users to put together new combinations of processing that determine the nature of model generation and the way in which analysis/simulation services are composed. SORASCS architecture employs some mechanisms for orchestration, coordination, and transformation. It enables users (analysts) to use a combination of graphical and textual inputs to specify and configure a collection of data transformation and analysis services. A registry of transponders and filters helps in finding an optimal chain of transformations. Finally, a registry of the services provided by the suite of existing and newly developed analysis and simulation tools, allows for selecting and composing services.

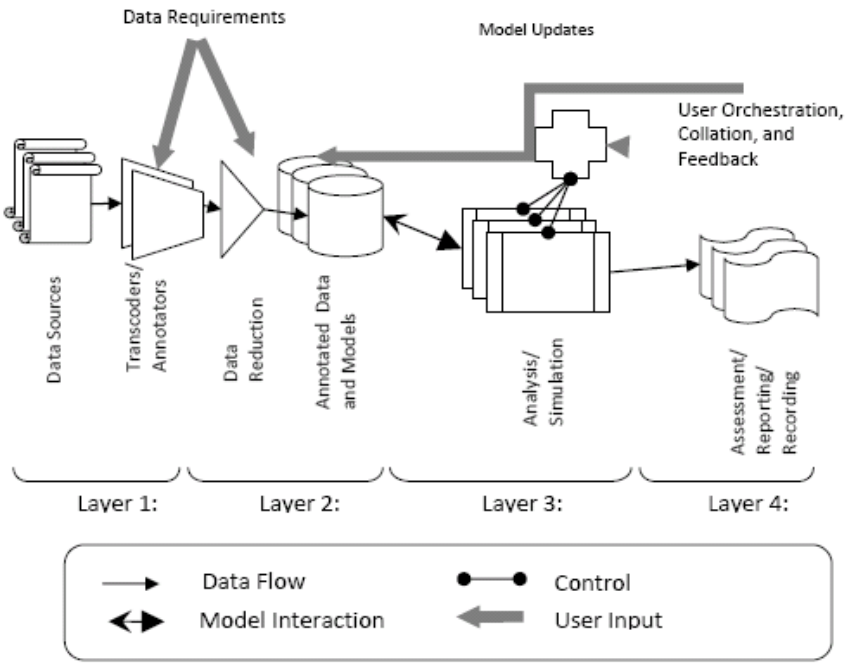

Figure 11: SORASCS Architecture [2]

2.2.1.3 NAOMI

NAOMI is an experimental platform for enabling multiple models, developed in different Domain Specific Modeling Languages (DSMLs), to work together. NAOMI analyzes model dependencies to determine the impact of changes to one model on other dependent models and coordinates the propagation of necessary model changes. NAOMI also serves as a useful testbed for exploring how diverse modeling paradigms can be combined. [27]

NAOMI allows modelers to use any modeling languages they desire, and as new modeling languages and tools are developed, they can be integrated into the NAOMI infrastructure. NAOMI provides a standard way for linking the multiple facets of a

system through a structured mechanism for specifying shared attributes of the system under design. [27]

Figure 12 shows a high level overview of NAOMI. Modelers edit models in DSMLs and use the Multi-Model Manager to construct the multi-model from individual models. Connectors isolate DSML specific interface code and interact with the Multi-Model Manager and the Sandbox, which provides modelers with a local copy of the Multi-Model Repository that holds the multi-model's artifacts under version control. The Execution Automation Engine determines an ordering of execution based on dependencies in the multi-model and orchestrates execution of the multi-model. The functionality provided by the system includes model data exchange, constraint management, change propagation, model execution, and version management. [27]

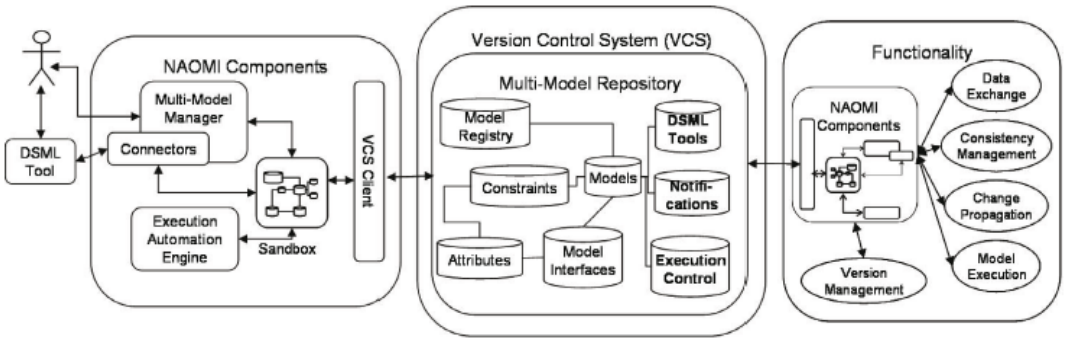

Figure 12: NAOMI [27]

2.2.1.4 AToM³

AToM^3 is a tool for multi-modeling under development at the Modeling, Simulation and Design Lab (MSDL) in the School of Computer Science of McGill University. It allows both meta-modeling and model-transforming. In the context of AToM^3, meta-modeling refers to the description or modeling of different kinds of formalisms used to model systems. Model-transforming refers to the (automatic) process of converting, translating or modifying a model in a given formalism into another model that might or might not be in the same formalism. [28]

In AToM^3, formalisms are described as graphs. From a meta-specification of a formalism, AToM^3 generates a tool to visually manipulate models described in the specified formalism. Model transformations are performed by graph rewriting. The transformations themselves can thus be declaratively expressed as graph-grammar models. [28]

2.2.1.5 OsMoSys

The Object-based multi-formaliSm MOdeling of SYStems (OsMoSys) is a framework that provides capabilities of multi-formalism modeling of systems. Its approach is based on meta-modeling, allowing to easily define and integrate different formalisms, and on some concepts from object orientation. Its main objectives are the interoperability of different formalisms and the definition of mechanisms to guarantee the flexibility and the scalability of the modeling framework. [29]

2.2.1.6 SIMTHESys

Structured Infrastructure for Multiformalism modeling and Testing of Heterogeneous formalisms and Extensions for SYStems (SIMTHESys) is an approach to multiformalism compositional modeling that is based on the possibility of freely specifying the dynamics of the elements of a formal modeling language in an open framework. This is obtained by the application of consolidated metamodeling foundations to the description of models, together with the concept of behavior as a bridge between formalism dynamics and solution techniques. [30]

2.2.1.7 Möbius

Möbius is a software tool for modeling the behavior of complex systems. It supports multiple modeling languages based on either textual or graphical representations. The tool was built under the belief that no one modeling formalism can be the best way to build all models of systems from across the diverse spectrum of application domains. In addition to the fact that many domain-specific modeling languages are needed, we also need many techniques (for example, simulation, state space exploration, and analytical solution) for analyzing models to study important behaviors of the systems being modeled. Möbius addresses those issues by defining a broad framework in which new modeling formalisms and model solution methods can be easily integrated, and populating that framework with multiple, synergistically combined modeling formalisms and model solution methods. [31]

2.2.2 Meta-Modeling for Multi-Modeling

In [14] a theoretical foundation for the use of multiple interacting models in order to determine the valid interaction between different modeling techniques was presented. The proposed approach was based on the use of Concept Maps, Meta-Models, and Ontologies to capture the valid interoperations between interconnected models. It extended earlier research efforts by Kappel et al. [32] and Saeki and Kaiya [33].

Figure 13 shows an overview of the approach phases. The first phase is the Conceptual Mapping level in which a Concept Map for each of the Modeling Languages used in a domain of interest is constructed. A Concept Map for a Modeling Language captures the assumptions, definitions, elements and their properties and relationships relevant to the domain. The resulting Concept Model is a structured representation; however, it is not formal, and hence it can't be used for machine reasoning. The first phase is followed by the Syntactic Modeling level in which Concept Maps representation and domain knowledge are formalized using Syntactic Models. This leads to the creation of a pseudo Ontologies that mirror the Syntactic Models and serve as Foundation Ontologies; they do not contain any semantic concepts related to the Modeling Language or the domain. In the third phase, semantic concepts and relationships are added to Foundation Ontologies to obtain the Refactored Ontologies. Once a refactored ontology is completed for each Modeling Language, mapping of concepts across the ontologies takes place. This results into an Enriched Ontology that contains concepts and relationships within and across multiple Refactored Ontologies.

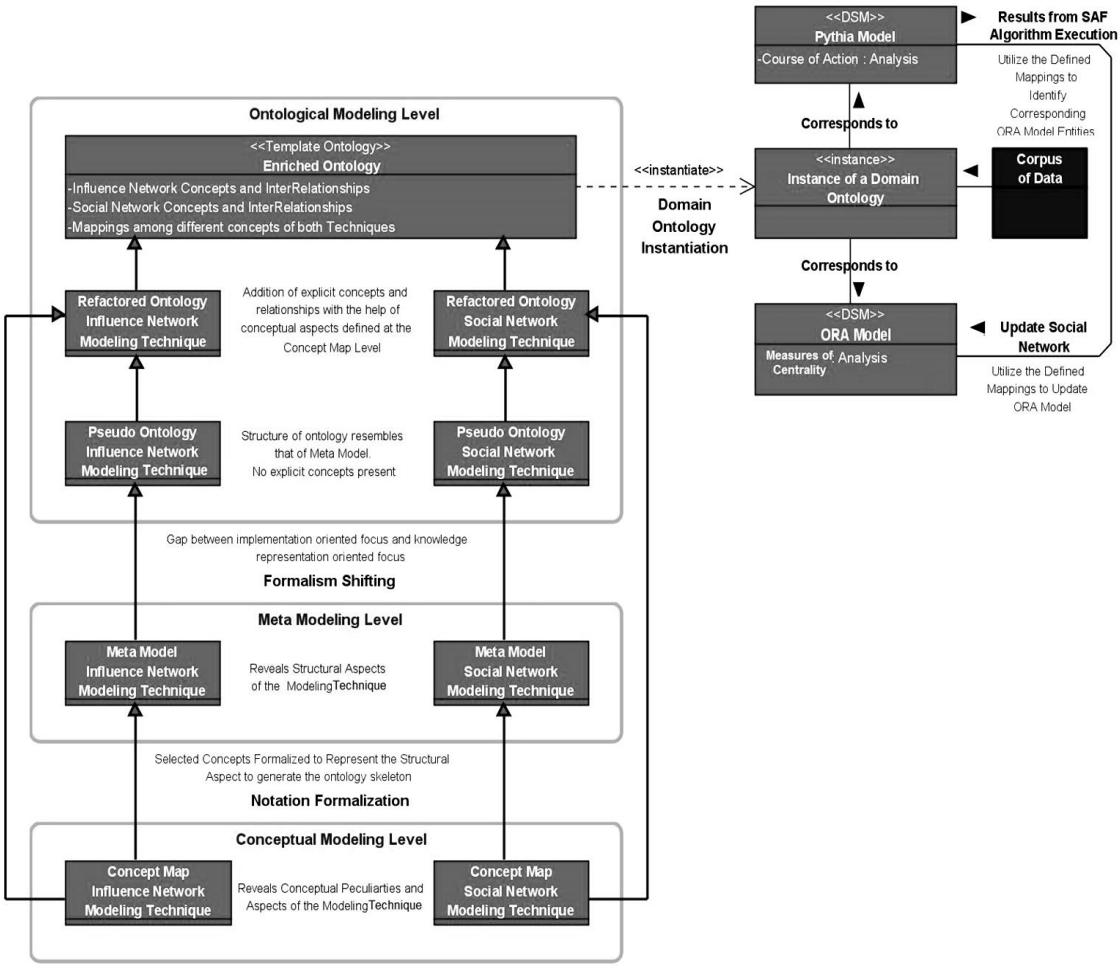

Figure 13: Meta-Modeling Approach [14]

2.2.3 Model Interoperability Using Meta-Models and Ontologies

In [34] a clear distinction between Meta-Models and Ontologies is made; they are different but complementary concepts, and both are needed to allow for model interoperation. Figure 14 presents the proposed architecture for semantic interoperability using both Meta-Models and Ontologies. Different models of the bottom model-layer are created corresponding to different Meta-Models that in turn are created using one common Meta²-Model. With the help of this approach, primarily syntactical and some

semantic aspects of model elements are defined. Ontologies are not seen as being completely independent from the language concepts that are used to create the models that are to be interconnected. The basis for semantic interoperability is provided via linking model elements of arbitrary layers of the Meta-Model hierarchy with ontology concepts.

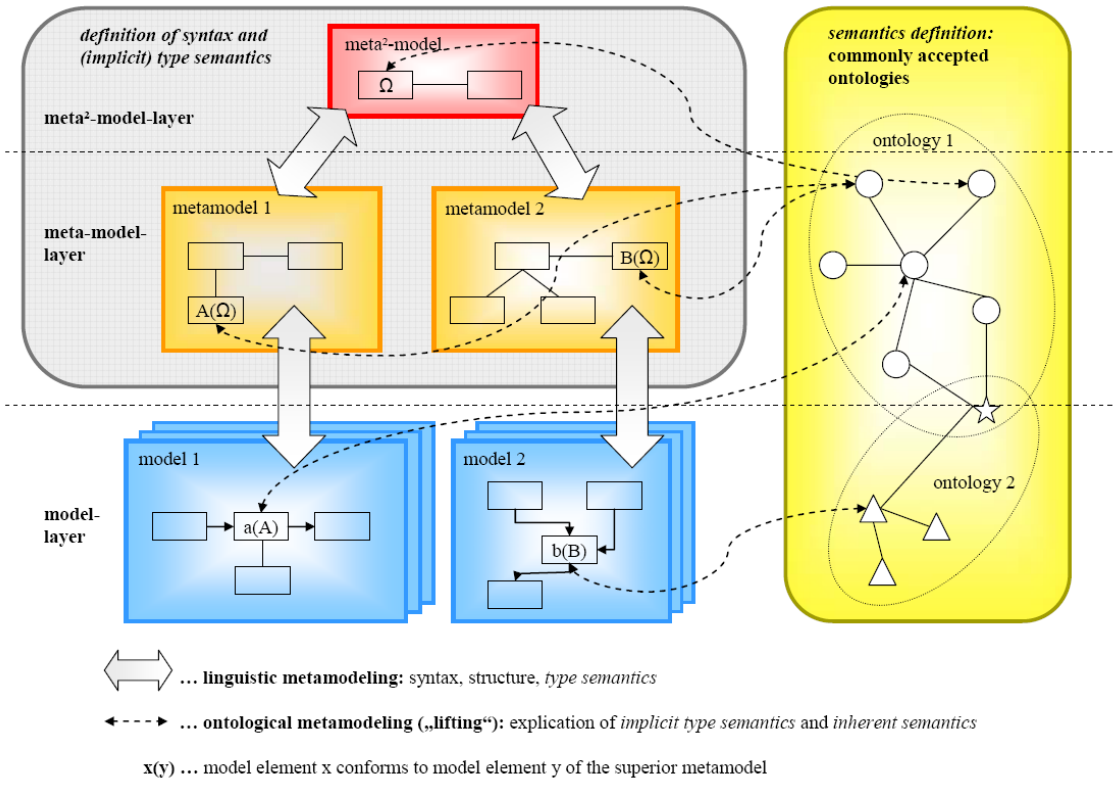

Figure 14: Model Interoperability Using Meta-Models and Ontologies [34]

2.2.4 Domain Specific Modeling Language (DSML)

Domain Specific Modeling Languages (DSMLs) are languages tailored to a specific domain. They offer a high level of expressiveness and ease of use when compared with a General Purpose Language (GPL). Development of a new DSML is not an easy or a straightforward activity. It requires both domain knowledge and language development expertise. DSMLs were developed simply because they can offer domain-specificity in better ways: [17]

- Appropriate or established domain specific notations are usually beyond the limited user-definable operator notation offered by GPLs.
- Appropriate domain-specific constructs and abstractions cannot always be mapped in a straightforward way to functions or objects that can be put in a library.
- Use of a DSL offers possibilities for analysis, verification, optimization, parallelization, and transformation in terms of DSL constructs that would be much harder or infeasible if a GPL had been used because the GPL source code patterns involved are too complex or not well-defined.
- DSLs need not be executable.

According to [17], the development of any new DSML should go in five phases:

- Decision: Deciding in favor of a new DSML is not easy and the decision phase should balance the tradeoffs between the benefits of using an existing general purpose language for developing a new specific language.

- Analysis: In the analysis phase of a new DSML development, the application domain should be identified and domain knowledge should be gathered. The output of domain analysis varies widely but consists basically of domain-specific terminology and semantics in more or less abstract form.

- Design: Basically there are two main approaches for designing new DSMLS. The easiest way is to base a new DSML on an existing Language (Language exploitation). In some cases the domain of the new DSML requires the use of the second approach, the invention of a new language.

- Implementation: The implementation of the newly developed DSML depends on the type of the language. Meta-Modeling environments like the GME can be used to implement a newly designed DSML. An executable language will need to be interpreted to the execution environment and hence an interpreter might be required for the new DSML as well.

- Deployment: Deployment of the newly developed DSML is related to the domain of interest and the toolset used in this domain. A specific tool can be built to allow for using the new language or a generic environment like the GME can also be used.

2.2.5 The Generic Modeling Environment (GME)

The Generic Modeling Environment is a configurable toolkit for creating domain-specific modeling and program synthesis environments (Domain Specific Languages). The configuration is accomplished through Meta-Models specifying the modeling

paradigm (Modeling Language) of the application domain. The modeling paradigm contains all the syntactic and presentation information regarding the domain including the concepts used to construct models, relationships between concepts, different views and organizations of the concepts, and rules governing the modeling process. Defining the modeling paradigm is considered itself as another modeling problem. GME has a Meta-Modeling paradigm that configures the environment for creating Meta-Models of the domain of interest. These models of the Meta-Models are then automatically translated into a GME configuration through model interpretation. The Meta-Modeling paradigm is based on the Unified Modeling Language (UML). Syntactic definitions are modeled using pure UML class diagrams and it is possible to capture static semantics with constraints using the Object Constraint Language (OCL). Figure 15 shows an overview of the GME architecture. [4]

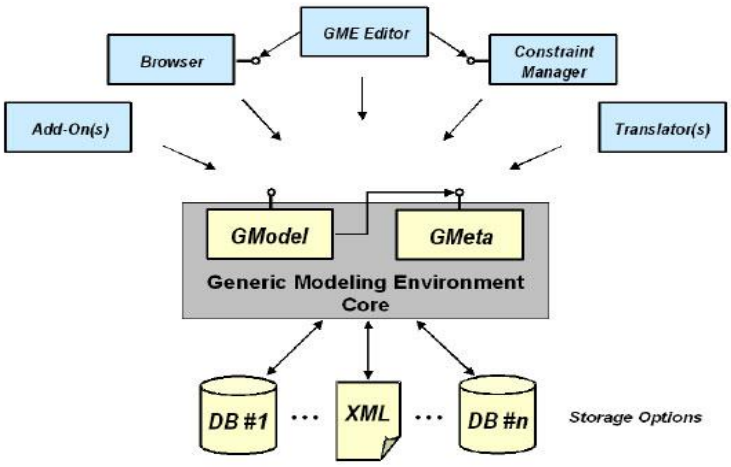

Figure 15: The GME Architecture [4]

GME supports various concepts for building complex models. These include: hierarchy, multiple aspects, sets, references, and explicit constraints. The UML class diagram in Figure 16 depicts the complex relationships among these and other important concepts. [35]

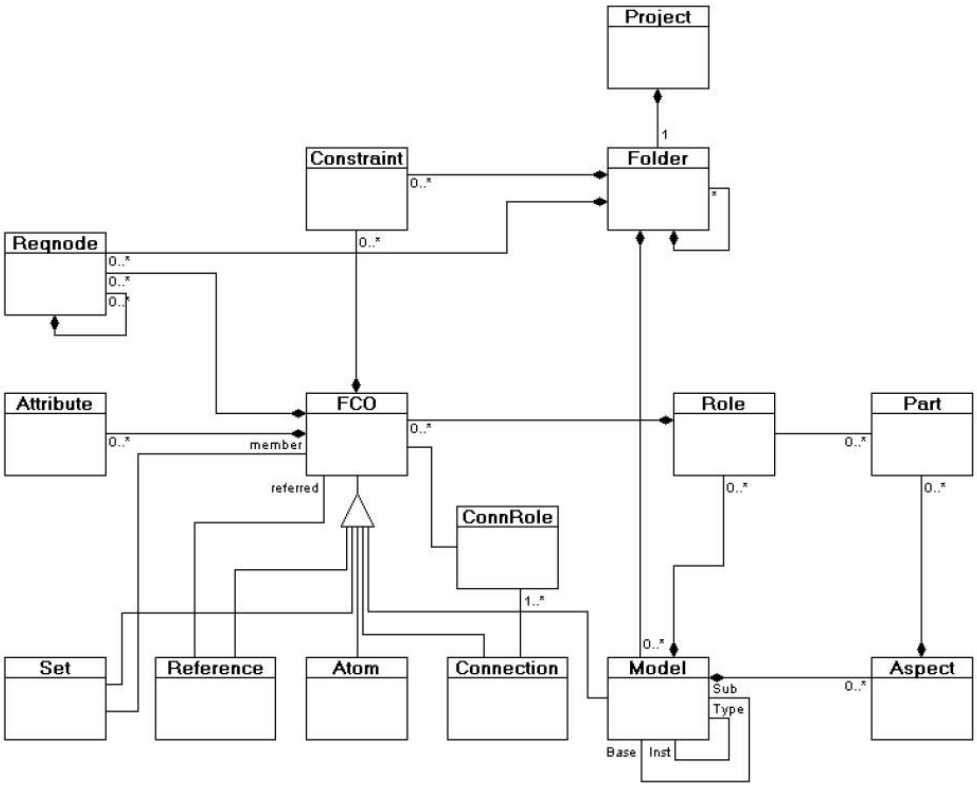

Figure 16: GME Concepts [35]

A Project contains a set of Folders. Folders are containers that help organize models. Atoms, References, Connections and Sets are all first class objects, or FCO-s for short. Atoms are the elementary objects – they cannot contain parts. Each kind of Atom is associated with an icon and can have a predefined set of attributes. The attribute values

are user changeable. Models are the compound objects; they can have parts and inner structure. The modeling paradigm determines what kind of parts are allowed in Models, but the modeler determines the specific instances and number of parts a given model contains. Aspects provide primarily visibility control. Every Model has a predefined set of Aspects. Each part can be visible or hidden in an Aspect. Every part has a set of primary aspects where it can be created or deleted. There are no restrictions on the set of Aspects a Model and its parts can have; a mapping can be defined to specify what Aspects of a part are shown in what Aspect of the parent Model. The simplest way to express a relationship between two objects in GME is with a Connection. Connections can be directed or undirected. Connections can have Attributes themselves. In order to make a Connection between two objects they must have the same parent in the containment. The paradigm specifications can define several kinds of Connections. It is also specified what kind of object can participate in a given kind of Connection. A Connection can only express a relationship between objects contained by the same Model. References are similar to pointers in object oriented programming languages. A reference is not a "real" object, it just refers to (points to) one. In GME, a reference must appear as a part in a Model. This establishes a relationship between the Model that contains the reference and the referred-to object. Connections and References are binary relationships. Some information does not lend itself well to graphical representation. The GME provides the facility to augment the graphical objects with textual attributes. All FCOs can have different sets of Attributes. The kinds of Attributes available are text, integer, real, boolean and enumerated. Folders, FCOs (Models, Atoms, Sets, References,

and Connections), Roles, Constraints and Aspects are the main concepts that are used to define a modeling paradigm. In other words, the modeling language is made up of instances of these concepts.

The Generic Modeling Environment (GME) allows for different types of extensions to the environment using plug-ins or add-ons. Plug-ins can be used to provide some useful additional functionality to ease working in GME. Add-ons on the other hand can be used to react to GME-events. These ways of extending the GME can be utilized to allow for a semantic guidance layer using a domain specific ontology for a domain specific language, i.e. the Multi-Modeling Workflow Language.

2.3 Closure

The background section of this chapter discussed some foundation concepts essential to the approach presented in Chapter Three. In particular, the concept of multi-modeling workflows discussed in section 2.1.5 is what drives this research effort. In addition, previous achievements in related research areas that were discussed in section 2.2 are utilized in our approach. Concepts of the various Multi-Modeling and Meta-Modeling approaches are utilized in different aspects of this research methodology.

CHAPTER THREE: THE MULTI-MODELING APPROACH

The multi-modeling platforms presented in [1] [2] and [27] and discussed in section 2.2.1 focus in general on the lower two layers of the multi-modeling layers shown in Figure 6. While the C2WT [1] utilizes the GME [4] to create the integration model that defines all the interactions between federated models and captures other design intent, its capability of allowing Subject Matter Experts (SMEs) to create high level workflows of multi-modeling activities is limited. The interaction model that it utilizes is more on the syntactic layer. Semantic concepts are implicitly embedded as integration rules or left to the judgment of the modeler. On the other hand, the SORASCS [2] platform focuses on providing a service-based environment for model interoperation. An analyst using the system can define a sequence of model interoperation through service invocation; and then such sequence can be saved as a "Workflow" for future use. This method doesn't provide analysts and modelers with the capability of creating workflows of model interoperations ahead of the multi-modeling based analysis. It also lacks the capability of checking the semantics of model interoperation prior to execution. NAOMI [27] provides a generic platform for multi-model integration through a Multi-Model Manager and Connectors. While it provides advanced capabilities through a comprehensive environment for designing and executing Multi-Model based analysis, it is restricted when it comes to defining workflows of the interoperations between models. Its Multi-

Model Manager (M3) provides an interface for defining a model-dependency graph which captures the sequence of operation. This dependency graph lacks the capability of capturing the interoperations between models on the data and operation level.

In this research, we focus on the top two layers of the multi-modeling layers as shown in Figure 6. Our approach is platform agnostic; we see a multi-modeling workflow as a high level visual and formal model that captures interoperations of multi-modeling activities and that can be implemented on any appropriate multi-modeling capable platform. In addition to capturing model interoperations in a workflow model, the semantic correctness of these interoperations is addressed by employing a supporting Domain Ontology.

In our approach we follow a systematic methodology for creating multi-modeling workflows that are both syntactically and semantically correct. The methodology involves both developing a Domain Specific Multi-Modeling Workflow Language (DSMWL), and then using the new language to create workflows of multi-modeling activities in an application domain. From a meta-modeling perspective, our approach focuses on the middle two layers of the meta-models hierarchy as shown in Figure 17.

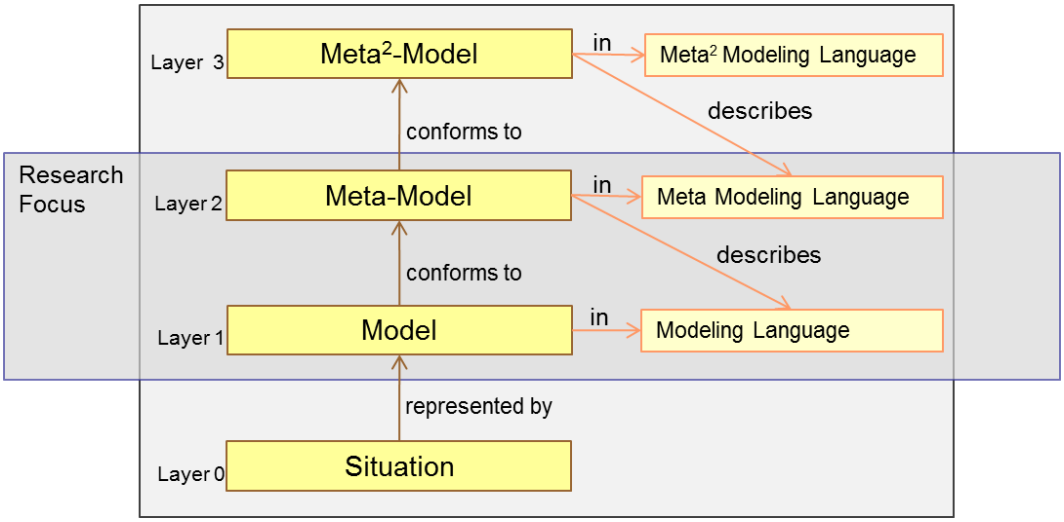

Figure 17: Meta-Modeling Research Focus

The methodology also addresses the implementation of workflows on appropriate platforms. The success of our methodology relies on the existence of application domain Subject Matter Experts (SMEs) who can effectively apply it for their domain/s of interest.

3.1 Overview of the Methodology

Our approach is domain specific; the rationale behind this is twofold: first, problems to be solved by employing multi-modeling techniques are usually domain specific themselves; second, it narrows down the scope of meaningful interoperations among several modeling techniques where each technique offers unique insights and makes specific assumptions about the domain being modeled.

The methodology consists of five major steps as shown in Figure 18. The first step is Domain Identification (DI) and characterization. This includes the identification and characterization of the Modeling Techniques used to analyze problems in this

domain. A Domain Analysis (DA) follows in the second step aiming to provide formal representations of syntactic and semantic aspects of the domain. In the third step a new Domain Specific Multi-Modeling Workflow Language (DSMWL) is developed. This new language is used to construct workflows that capture multi-modeling activities for analyzing problems in the selected domain, the fourth step of our methodology.

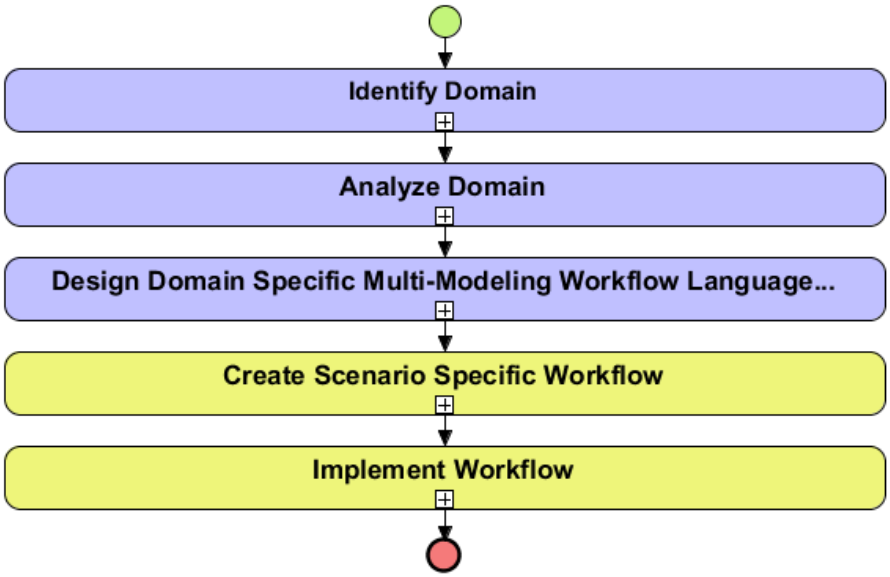

Figure 18: The Multi-Modeling Approach

A domain Ontology resulting from the DA step is utilized to provide semantic guidance that effects valid model interoperation while creating workflows in the fourth step. Once a workflow that represents a multi-modeling activity is created, it can then be implemented in the fifth and final step. The workflow implementation can take place in any appropriate platform that supports multi-modeling i.e., the C2WT [1] or SORASCS [2].

3.1.1 Domain Identification (DI)

This is the first step which deals with characterizing a specific domain of interest in which interoperating models are used to analyze certain problems. We address the domain identification challenge by employing different techniques. As discussed in section 2.1.8, the domain identification problem has been approached repeatedly during the late 80's and early 90's among the software reusability research community. In our DI step we utilize some of the techniques proposed during that era. Informal textual domain description, defining domain boundaries and content, classification of concepts, identifying sources of data, and identifying main actors are among those major techniques proposed in [25] [26] [37] [38] [39] [40] [41] [42] [43] [44] and adopted in our methodology.

As shown in Figure 19, we begin with an informal description of the domain in the form of textual statements that:

- Describe the problems to be analyzed in the domain.
- Identify the Modeling Techniques usually used in analyzing these problems
- Specify data sources and the different types of data used in in the application domain.
- Select the main actors involved including domain experts, modelers and analysts.

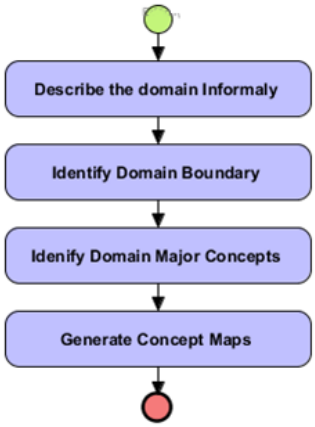

Figure 19: Domain Identification

The initial informal description might contain concepts that are outside the scope of the problem of interest. This leads to the next step in which the domain boundary is decided. This step is important in order to establish what is in and what is outside of the application domain with respect to the problem of interest. It delineates the scope of the modeling effort and helps in focusing on the concepts related to the application domain and the modeling techniques used in this domain.

So far the resulting product of the DI step is still an initial informal description of the domain. Since eventually the concepts captured in this informal description are to be used for formal domain analysis, it would be valuable to transform these concepts into an intermediate semi-formal representation. For this purpose initially a classification of concepts applicable to the domain takes place. In general, two high level classifications of concepts are required, concepts related to the Domain and concepts related to the Modeling Techniques used in the Domain. Table 1 and Table 2 are examples of templates for capturing both Domain and Modeling Technique concepts respectively.

Table 1: Domain Concepts Template

Domain Concepts	
General Concepts	**Specific Concepts**
Data	Type Source
Actors	Roles Groups
Analysis Problems	Analysis Techniques Data Requirements Resources Requirements
Analysis (Modeling Techniques)	Modeling Languages Modeling Tools

Table 2: Modeling Technique Template

Modeling Technique Concepts	
General Concepts	**Specific Concepts**
Model	Elements Constructs Data Types
Modeling Tool	Operations Data Types
Domain	Capabilities Requirements
Actors	Modelers

Once sufficient concepts are identified and classified in the form of tables, concept maps [36] are then created to capture those concepts and the relationships between them. Concept mapping is a representation technique to organize knowledge about a specific domain.

A collection of concept maps captures groups of related concepts and answers specific questions about the domain. Figure 20 shows an example of a template Concept Map that represents some of the concepts captured in Table 1.

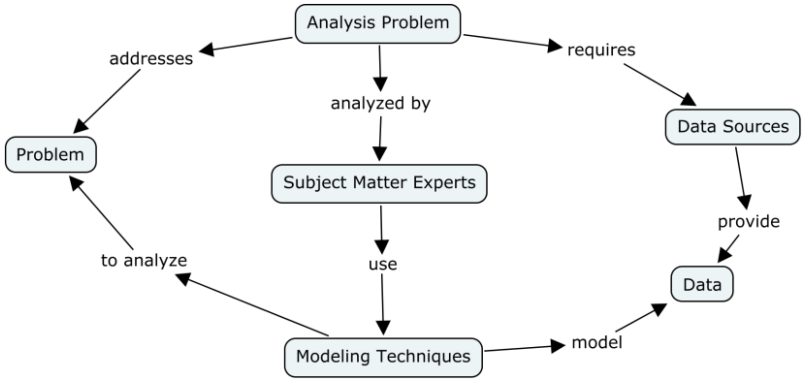

Figure 20: Concept Map Template Example

The Domain Identification step is an iterative process; the refinement of domain concepts and consequently the domain concept maps can be repeated multiple times. This step can be revisited any time a new modeling technique is introduced in solving domain problems.

3.1.2 Domain Analysis (DA)

Once satisfactory concept maps that represent the domain of interest and its supporting modeling techniques are ready, the Domain Analysis (DA) step begins. This process is important in order to transform the semi-formal domain representation in the form of concept maps into more formal representations that can be used in the next steps. The process, as shown in Figure 21, goes into two parallel, but complementary, paths. On the outer path, UML class diagrams derived from the concept maps are produced to capture the structural aspects of the domain including its supporting Modeling Techniques. A domain class diagram captures the major constructs of the domain:

- Problems to be solved and techniques used to analyze and solve them.
- Modeling techniques used in the domain.
- Data types and sources of data.
- Participating actors and their roles.

Concept maps of modeling techniques are also used to create class diagrams that represent each of them. A modeling technique class diagram would include:

- Modeling Language structure.
- Model constructs.
- Modeling Tool operations.

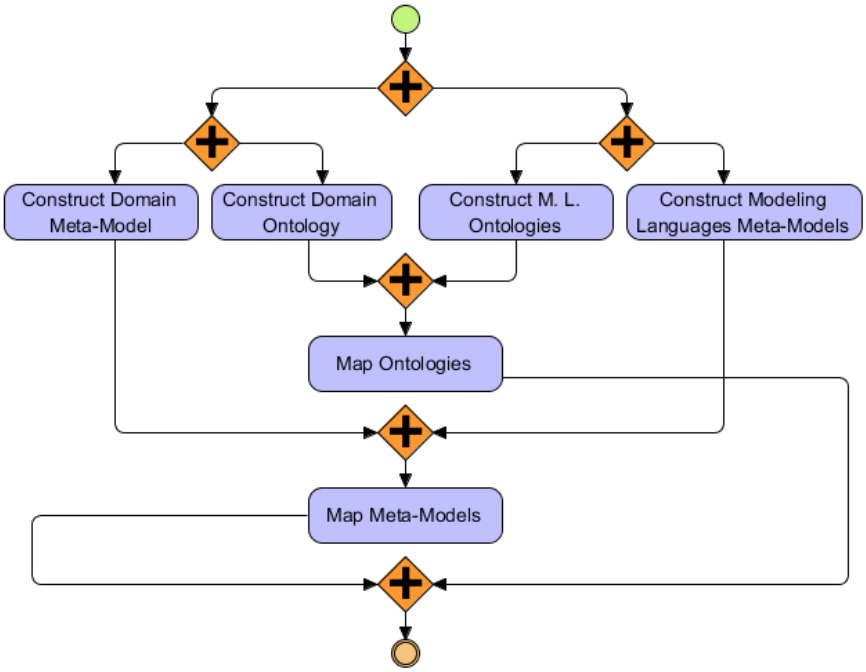

Figure 21: Domain Analysis

Mapping between Domain and Modeling Techniques class diagrams follows to produce a consolidated diagram that captures domain concepts and possible model interoperations. This is a manual step that relies on the expertise of the SMEs.

On the inner path of Figure 21, ontologies based on the concept maps of the domain and the modeling techniques are constructed to capture the semantic aspects of both. These ontologies are represented using the formal Web Ontology Language (OWL) [45] and are created using Protégé [46]. Mapping of these ontologies follows by employing Upper Ontology [21] and Ontology Matching [22] techniques. The utilization of ontologies in our approach is discussed in detail in Chapter 5.

3.1.3 Domain Specific Multi-Modeling Workflow Language (DSMWL)

A meta-model of the new language has to be created and it should include the set of fundamental language constructs that represent the essential concepts of the domain, the set of valid relationships that exists between the domain concepts, and a set of constraints that govern how the language constructs can be combined to produce valid models. Accordingly, in the third step of our methodology, the consolidated UML class diagram obtained from the DA step is used as the basis for the meta-model that defines the domain specific multi-modeling workflow language. The GME is used to create the meta-model of the multi-modeling workflow language. This meta-model is then automatically translated into a GME configuration that allows the use of GME itself to create workflows of specific multi-modeling scenarios. In general, we propose the use of a profile of BPMN as the basis of any domain specific multi-modeling workflow language. In Chapter 4 we discuss this step in detail.

The semantic concepts identified in the domain identification process and then captured in the Ontology in the domain analysis step should be enforced while using the new domain specific multi-modeling workflow language. Since our ontologies are represented in OWL and we are using GME to create multi-modeling workflows, there should be a way to allow OWL ontologies to guide the creation of workflows, that is, to guarantee their semantic correctness.

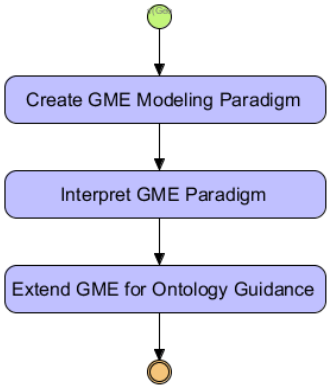

Figure 22: Domain Specific Multi-Modeling Workflow Language

GME allows for different types of extensions to the environment using Plug-ins or Add-ons [4]. Utilizing these GME extensibility features and in order to address the semantic guidance issue, we implemented a GME Add-on extension. This extension reacts to GME events and, in case of any interoperation connection while using our multi-modeling workflow language, the OWL ontology is checked on the semantic validly of this connection. We use SPARQL [47] queries that are passed to a SPARQL Query Server to query the ontology. Based on the query result, our GME extension could allow or disallow the interoperation connection. We discuss the GME Ontology extension in detail in Chapter 5.

3.1.4 Multi-Modeling Workflow Creation

After defining the domain specific multi-modeling workflow language and having its GME modeling paradigm interpreted, GME can be used to create workflows for specific situations of interest. This is the fourth step of our methodology.

In order to solve a specific problem in the selected domain, a SME would capture all available information about the problem, analyze the information and then decide on which suitable modeling techniques can help in solving the problem. The SME would then use the developed multi-modeling workflow language to visually create a workflow model that represents the sequence of operations each modeling technique is contributing to the problem including interoperations among models. Figure 23 shows the 3 major tasks of this step.

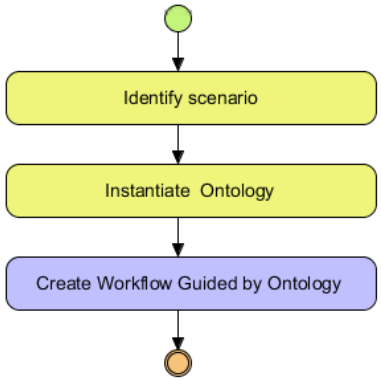

Figure 23: Multi-Modeling Workflow Creation

Since the domain specific multi-modeling workflow language was developed following an extensive domain analysis, it is sufficiently rich with possible operations and interoperations that address the problem of interest in the domain. While constructing the workflow model, the domain ontology is continuously checked to validate the semantic correctness of any interoperation. The GME extension serves the purpose of this domain ontology integration into the workflow construction process.

3.1.5 Workflow Implementation

The final step is to implement our workflows in an appropriate platform. In order to achieve this, an interpretation of the workflow to an executable form is required. For this purpose, a GME interpreter is required. The interpreter transforms the workflow to a format that the implementation platform uses to execute workflows of multi-modeling activities. Chapter Six discusses this step in more detail.

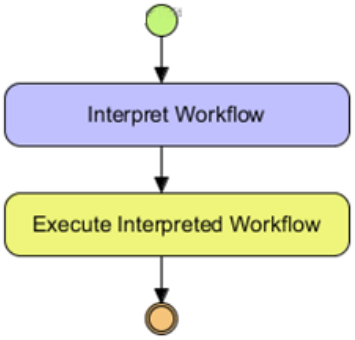

Figure 24: Workflow Implementation

3.2 Two-Phase View of the Approach

The overall process of the methodology can be viewed as a two phase approach as shown in Figure 25 and Figure 26.

Phase 1 is where the first three steps, domain identification, domain analysis, and workflow language definition take place. For a specific domain, this phase goes into multiple iterations until a Multi-Modeling Workflow Language that addresses the domain of interest and is capable of capturing model interoperations is reached.

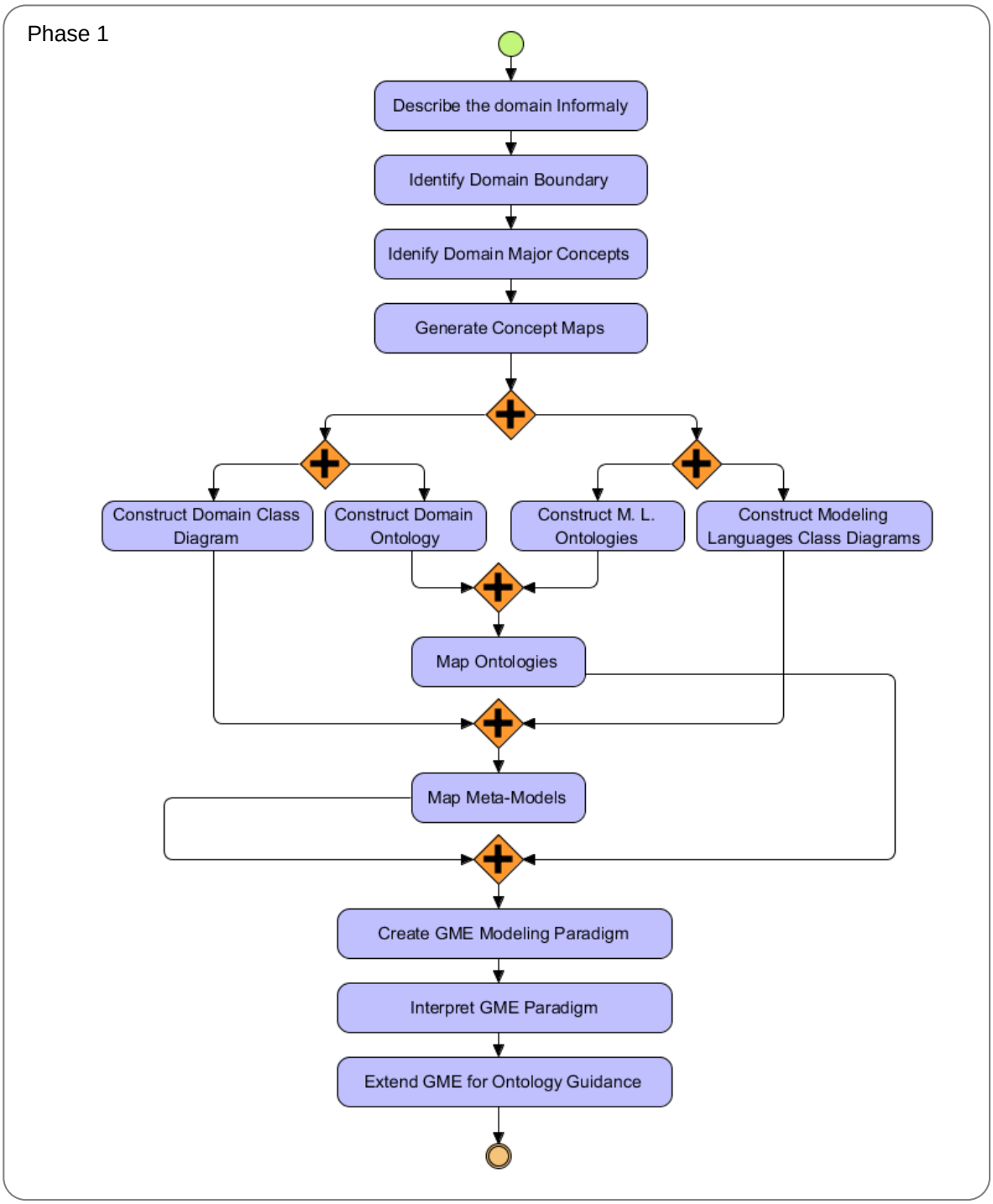

Figure 25: Phase 1 of the Methodology

Phase 2 takes place when the workflow language is used to create workflows for specific scenarios. It is always possible to go back to Phase 1 to refine and enhance the multi-modeling workflow language; this might be the case when a new modeling technique is introduced in the domain of interest.

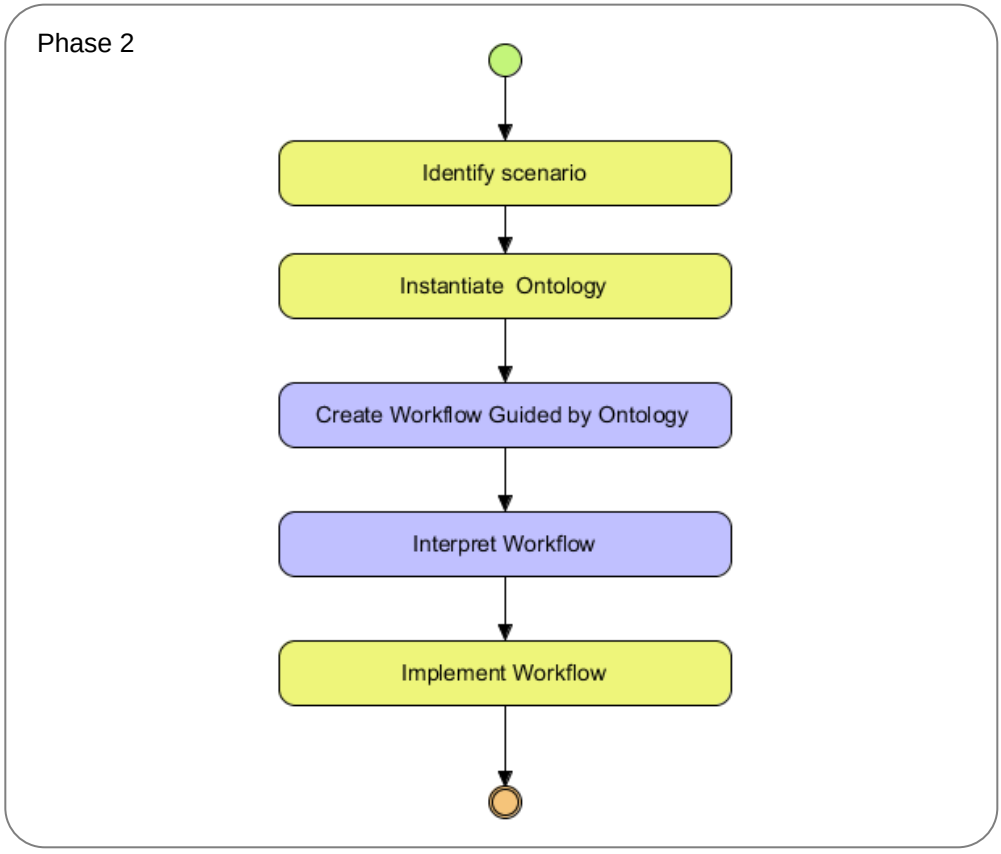

Figure 26: Phase 2 of the Methodology

CHAPTER FOUR: THE DOMAIN SPECIFIC MULTI-MODELING WORKFLOW LANGUAGE

In Chapter Three, a methodology consisting of five steps for addressing multi-modeling through the use of a Domain Specific Multi-Modeling Workflow Language (DSMWL) has been presented. The third step of the methodology, in which the DSMWL has to be developed, is a cornerstone for our approach. In this chapter we discuss this step in detail and show how the Generic Modeling Environment (GME) configuration for the new language is created. However, before we begin the discussion it is worth referring back to section 2.2.4 where the trade-offs between using an existing Generic Purpose Language (GPL) and developing a Domain Specific Modeling Language (DSML) were highlighted.

Table 3: GPL, DSML and DSMWL relationships

General	**GPL**	**UML**	**GME (Generic Tool)**
Specific	DSML	BPMN	DSMWL
Model	Specific Model	Specific Model	Workflow

Table 3 shows the relationships between a GPL, a DSML and the DSMWL. A GPL such as UML is used by GME to create a meta-model of the DSMWL. The DSMWL can be a profile of an existing DSML like BPMN. The DSMWL is then used to create a workflow that addresses a specific situation. In developing the DSMWL, the five phase approach of Mernik et al. [17] for developing a new language is adopted.

4.1 Phase 1: Decision Towards a DSMWL

A decision to develop a DSMWL has to be made. Since we deal with a number of modeling techniques to address problems in a specific application domain, the use of domain specific multi-modeling workflow language is required to create workflows of multi-model interoperations. Each domain has its own characteristics and interoperations between various models that require specific constructs in the workflow language used to capture them. Our decision on this initial phase is to develop a domain specific multi-modeling workflow language for each application domain. The development of a new language for each application domain doesn't necessarily mean that the work for other domains is of no use. As mentioned in section 3.2, the first phase of the approach is iterative. Working on a new domain can be thought of as revisiting the first phase of a previously addressed application domain. This is especially feasible when the new domain employs similar modeling techniques to those in the previously addressed domain.

4.2 Phase 2: Analysis for a New DSMWL

In the second phase, analysis has to take place to study the application domain and identify the constructs of the new DSMWL. The first step of our methodology, the Domain Identification (DI) step, serves this purpose. The DI step provides information about the domain and its main concepts. It results in informal and semi-informal description of the domain and its supporting modeling techniques. This provides preliminary understanding of the DSMWL requirements.

4.3 Phase 3: Designing a New DSMWL

The third phase deals with the design of the DSMWL. In general, there are two main approaches for designing a new domain specific language. One way is to base it on an existing language (language exploitation). The second approach would be the invention of new language. Developing a completely new language is not a trivial task. It requires extensive domain knowledge in addition to expertise in developing modeling languages. In our approach, we favor the first option in which the DSMWL is based on an existing GPL. Our desired DSMWL is intended to capture workflows of multi-modeling interoperations. There already exist GPLs that allow for creating and executing workflows, e.g., the Business Process Modeling and Notation (BPMN) [15] and the Business Process Execution Language (BPEL) [16].

In section 2.1.5 we discussed the different aspects and classes of workflow languages as presented in [18]. The goal of our approach is to provide a methodology that allows for developing and then using a DSMWL. Such a language should be capable of

providing its users with the capability of visually creating workflows that represent multi-modeling activities. This requires the development of a graph-based language since our methodology addresses both creating and implementing workflows. The language should also be capable of capturing other aspects including functional, behavioral, informational, organizational and technical ones, in addition to allowing flexibility. In order for a DSMWL to be able to capture different aspects, graph-based features should be complemented with other supporting techniques. Such techniques, like model attributes, should allow for specifying domain and implementation platform information as part of the workflow model creation process. This leads us to identify three major types of multi-modeling workflows that can be addressed by our methodology:

- Graphical Multi-Modeling Workflow: A workflow model of this type would basically capture a specific situation of interest where multiple models interoperate to provide analysis results. A workflow diagram represents the flow of activities represented by features available through supporting modeling tools. These activities could involve multi-model interoperation. It doesn't specify any implementation related information as part of the workflow.

- Implementation Multi-Modeling Workflow: This type of workflow captures implementation specific information. It deals with low level syntactic/semantic interactions between models. Implementation information can be captured in the form of scripts or attributes.

- Hybrid Graphical & Implementation Multi-Modeling Workflow: A hybrid workflow model would basically allow for creating graph-based diagrams in addition to capturing low level syntactic/semantic information.

Our methodology deals with the third type, the Hybrid Multi-Modeling Workflow that allows for capturing the flow of activities in addition to syntactic/semantic aspects required for workflow implementation.

We have already discussed an example of a graph-based workflow language, the Business Process Modeling and Notation (BPMN), in section 2.1.5. It is an example of a graph-based workflow language that provides a readily understandable notation. Its core and complete element sets allow expressing a wide variety of workflow constructs. In [18], the BPMN has been characterized as having the flavor of a framework more than of a concrete language. The four categories of its core element set allow for creating workflow diagrams that capture different aspects of a workflow model, a multi-modeling workflow model in our case. Also, since we are interested in capturing interoperations between models, the hierarchal features of BPMN are of special importance to our approach.

Due to the aforementioned reasons we propose, in general, the use of a profile of BPMN as the basis of any new Domain Specific Multi-Modeling Workflow Language (DSMWL). In Figure 27 an example of an initial basic set of elements a DSMWL would include is shown. We extended/constrained some of the core elements of BPMN to propose this profile.

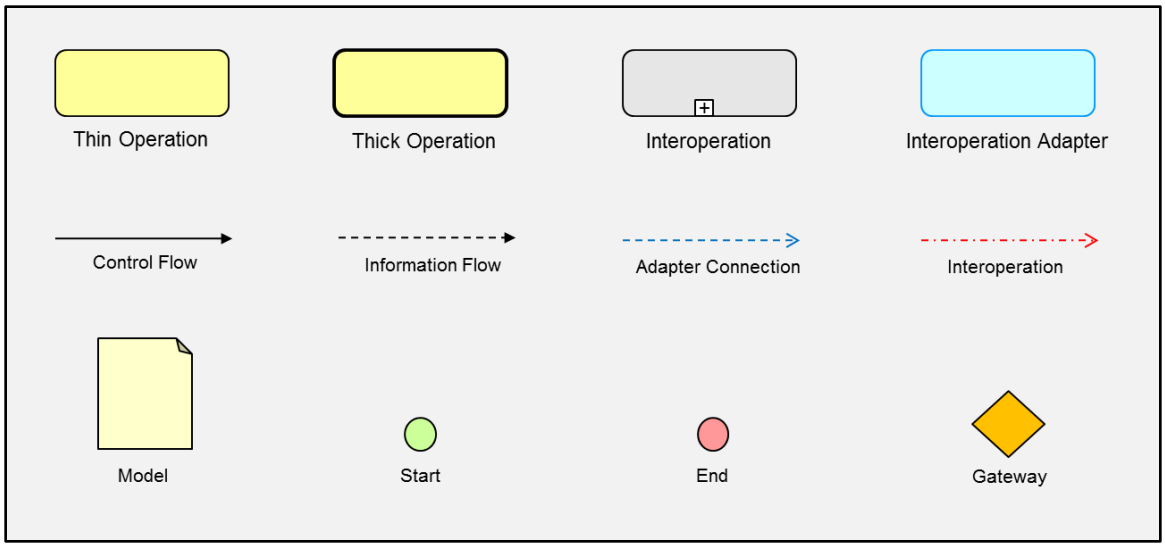

Figure 27: Constructs of a Multi-Modeling Workflow Language Based on BPMN

A multi-modeling workflow model should capture activities performed using various modeling techniques. A modeling technique activity represents a modeling tool operating on a model that conforms to the tool's supported modeling language.

We borrowed the concept of activities from BPMN and classified them into two major categories. A workflow model in our approach has two types of activities, operations and interoperations. Operations are those activities where a single modeling tool is operating on a single model. Operations are themselves classified into two classes:

- **Thin Operations**: Thin operations represent the case when modeling tools of interest have the capability of exposing their functionalities as services. A thin operation is represented graphically using a round-corner yellow rectangle with a thin border.

- **Thick Operations**: Thick operations represent the case in which a legacy modeling tool is used as a package in the overall multi-modeling activity. A thick operation is represented graphically using a round-corner yellow rectangle with a thick border.

The concept of thin and thick tasks was originally used in the SORASCS architecture to distinguish between services exposed by modeling tools and the use of modeling tools as standalone packages [2]. Modeling tools vary, some provide advanced capabilities like exposing their functionalities as services that can be integrated or invoked, and some others are black boxes and have to be used as a package.

Interoperations are those activities that involve multi-model interoperation. This is the case when a modeling tool operating on a model requires or passes information to other tool operating on another model.

- **Interoperations**: An interoperation in our proposed workflow language is a hierarchical container that captures a multi-modeling activity where two models are to be interacting through their supporting modeling tools thin/thick operations. It is represented using a round corner gray rectangle with a plus sign in the middle.

In addition to including thin or thick operations, interoperations also include into their sub diagram interoperation adapters.

- **Interoperation Adapter:** An interoperation adapter is the element that captures the information aspect of exchanging data between models. It is itself a hierarchical

element that includes detailed mapping of data between models in its sub diagram. It is represented using a round-corner blue rectangle.

Since the main purpose of the workflow language is to capture the operations and interoperations between models through their supporting modeling tools, a representation of the model is required.

- **Models:** We borrowed the data element from BPMN and used it as a representation of models in our workflow language. A model in our approach includes attributes that allow for mapping data to other models. We consider these attributes to be of two types, method data attributes and item data attributes. Method attributes are those data values that result from a tool operation on the model. Item attributes are data values that are obtained directly from a model item. A model is represented using one folded corner vertical yellow rectangle.

Events and gateways are also used as in BPMN. We propose the use of a subset of BPMN events and gateway elements:

- **Event:** An event is something that happens during the course of the workflow and can affect its flow. Basically we have two types of events, Start event represented by a green circle and End event represented by a red circle.

- **Gateway:** It controls how sequence flows interact as they converge and diverge. In general, we use a split/merge gateway represented by an orange diamond shape with plus sign inside it.

We use different styles of arrows to represent connections between workflow model elements:

- **Control Flow Arrows:** Control flow arrows connect workflow elements of type operation, interoperation, event and gateway. These control arrows represent the sequence of activities in a workflow model. They are represented by solid black arrows.

- **Information Flow Arrows:** These arrows determine the information flow from and to models. They can link models to operations or interoperations. They are represented by dashed arrows with closed arrowhead.

- **Adapter Connection Arrows:** An adapter connection arrow connects two models that are exchanging data inside an interoperation. It is represented by a dashed arrow with open arrowhead.

- **Interoperation Arrows:** These arrows represent the mapping between data types of interoperating models inside an interoperation adapter. They are represented by dashed doted arrows with open arrowhead.

4.4 Phase 4: Meta-Model of the New DSMWL

A meta-model of the DSMWL has to be created and it should include the set of fundamental language constructs that represent the essential concepts of the domain, the set of valid relationships that exists between the domain concepts, a set of constraints that govern how the language constructs can be combined to produce valid models, and the

concrete syntax or notation of the DSML. We consider this to be the fourth phase in which the new DSMWL is to be implemented.

The second step of our approach, the Domain Analysis (DA) step, forms the basis of this phase. The DA step results in formal representation of the domain and its supporting modeling techniques in the form of UML class diagrams that eventually are consolidated into one comprehensive UML class diagram. This consolidated class diagram together with the basic elements and constructs identified in the previous third phase are used to create a meta-model for the DSMWL. Defining the meta-model of the new language in Layer 2 of Figure 6 is a meta-modeling process itself. To capture those constructs of the meta-model that define the new language, a meta-modeling language that conforms to a higher meta-model, meta^2-model, is also required. We use a UML class diagram to represent the meta-model of the language. Figure 28 shows an example of a meta-model for a new Domain Specific Multi-Modeling Workflow Language.

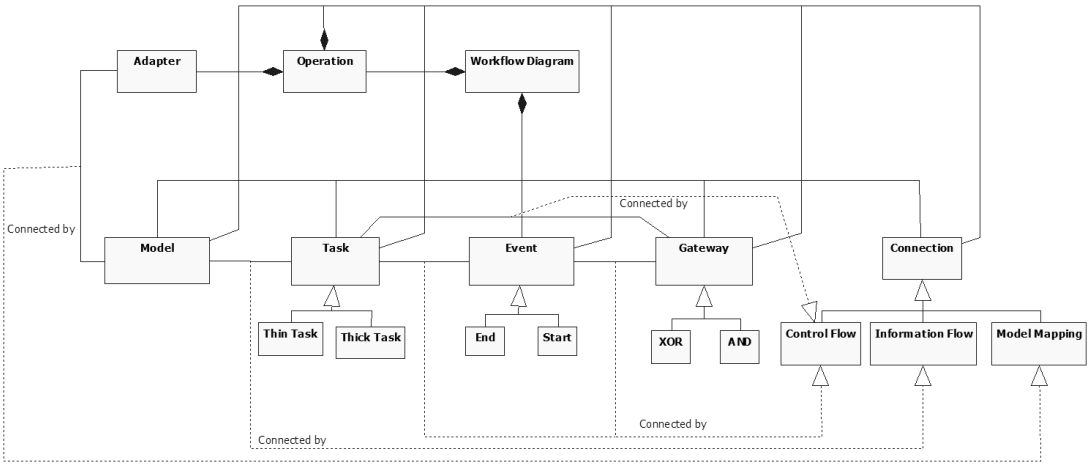

Figure 28: Meta-Model Example

We use the GME platform to develop the DSMWL which will be discussed in the next section. GME bases its meta-modeling language, the MetaGME, on UML. This allows the mapping of the meta-model of our new Domain Specific Multi-Modeling Workflow Language to the GME platform.

4.5 Phase5: Deployment of the New DSMWL using GME

We presented in section 2.2.5 the Generic Modeling Environment (GME) platform. GME provides the capability of configuring its environment through meta-models specifying a new modeling paradigm (new modeling language). Deployment is the term we use to refer to this process.

In our approach, the GME is used to deploy the meta-model of the new Domain Specific Multi-Modeling Workflow Language (DSMWL). This meta-model is then automatically translated into a GME configuration that allows the use of GME itself to create workflows of specific multi-modeling scenarios.

A major advantage of using GME is that it allows the definition of a graph-based language with the flexibility of modifying the meta-model of the language and retranslating it into an updated GME configuration. It also supports various concepts for building large-scale, complex models including: hierarchy, multiple aspects, sets, references, and explicit constraints. The vocabulary of the domain-specific language implemented by different GME configurations is based on a set of generic concepts built into GME itself. The choice of these generic concepts is the most critical design decision.

In section 2.2.5 the main concepts provided by the GME for deploying Meta-models that define new Modeling Languages were discussed. In this phase of defining our new domain Specific Multi-modeling Workflow Language we utilize these concepts to transform our UML based Meta-model of phase 3 into a GME paradigm. We begin the process be creating a new GME project that should contain the definition of the new modeling paradigm. Inside this GME project we create a folder that should host the models of our paradigm. Then, under this folder we create a model that should contain the elements of our paradigm. Figure 29 shows a snapshot of GME interface with the project, folder and model created for our new language paradigm.

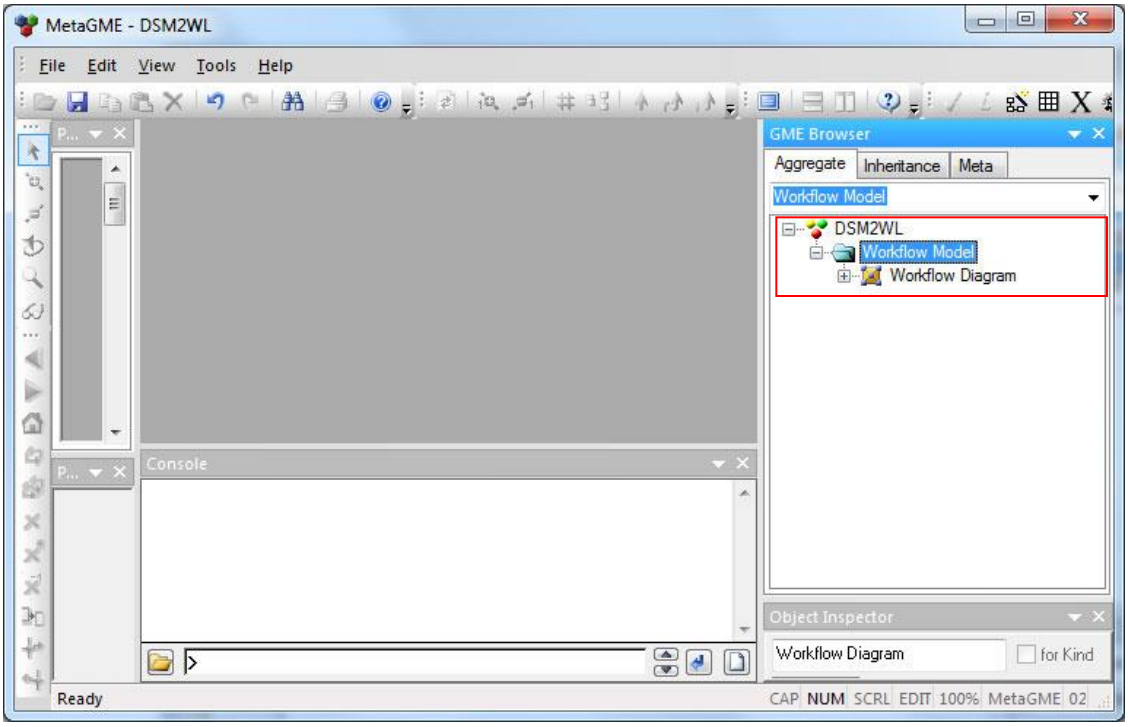

Figure 29: GME Project and Folders

Before we continue with defining the models of the GME paradigm, we first map the constructs of the new language identified in section 4.3 and shown in Figure 27 to the basic concepts of GME.

Table 4 summarizes this mapping. A GME *Model* is a compound structure; it can contain sub-elements and sub-structure. A workflow diagram is a structure of sub-elements and therefore it is represented by a GME *Model*.

Table 4: Mapping of Langauge Constructs to GME Concepts

Language Construct	**GME Concept**
Workflow Diagram Interoperation Interoperation Adapter Analysis Model	Model
Operation (Thin, Thick) Gateway (Split, Join) Event (Start, End)	Atom
Control Flow Connection Information Flow Connection Adapter Connection Interoperation Connection	Connection

Since our language is a hierarchical one in which some of its workflow elements can have inner structure themselves, we represent those elements by a GME *Model* as well. This includes interoperations, interoperation adapters, and analysis models. An interoperation represents the activities between a number of analysis models through their modeling tools. This includes the use of thin and thick operations in addition to

interoperation adapters in its sub-structure. A sub-structure for an interoperation adapter captures the mapping between analysis models data. An analysis model includes atomic substructure to represent its content types.

GME *Atoms* are the elementary objects and cannot contain parts. Thin and thick operations are represented as GME *Atoms.* They are basic elements in our language and can't have any sub-structure. Gateways and Events are of GME *Atom* type as well.

Control flow, information flow, interoperation and interoperation adapter connections are all of type GME *Connection.* A control flow connection can connect elements of type operation, interoperation, event, and gateway. An information flow connection connects an analysis model with an operation or an interoperation. The interoperation connection connects an interoperation adapter with analysis models and has to be part of an interoperation sub-structure. To connect analysis models inside an adapter, an interoperation adapter connection is used.

In addition to the use of Model, Atom and Connection concepts, we use GME *Attributes* to define specific domain related information. An example of the use of attributes would be when domain and modeling technique specific information is to be captured in the language design.

GME supports different aspects when defining the paradigm of a new language. GME *Aspects* provide primarily visibility control and allow for separation of concerns while building the new language paradigm. Figure 30 shows an initial *Class Diagram*

Aspect for the GME paradigm designed for our Domain Specific Multi-Modeling Workflow Language. This is basically a view of all the elements of the language and the relations between them. This is not a complete paradigm since it doesn't have domain specific concepts but rather shows an example of how GME is used.

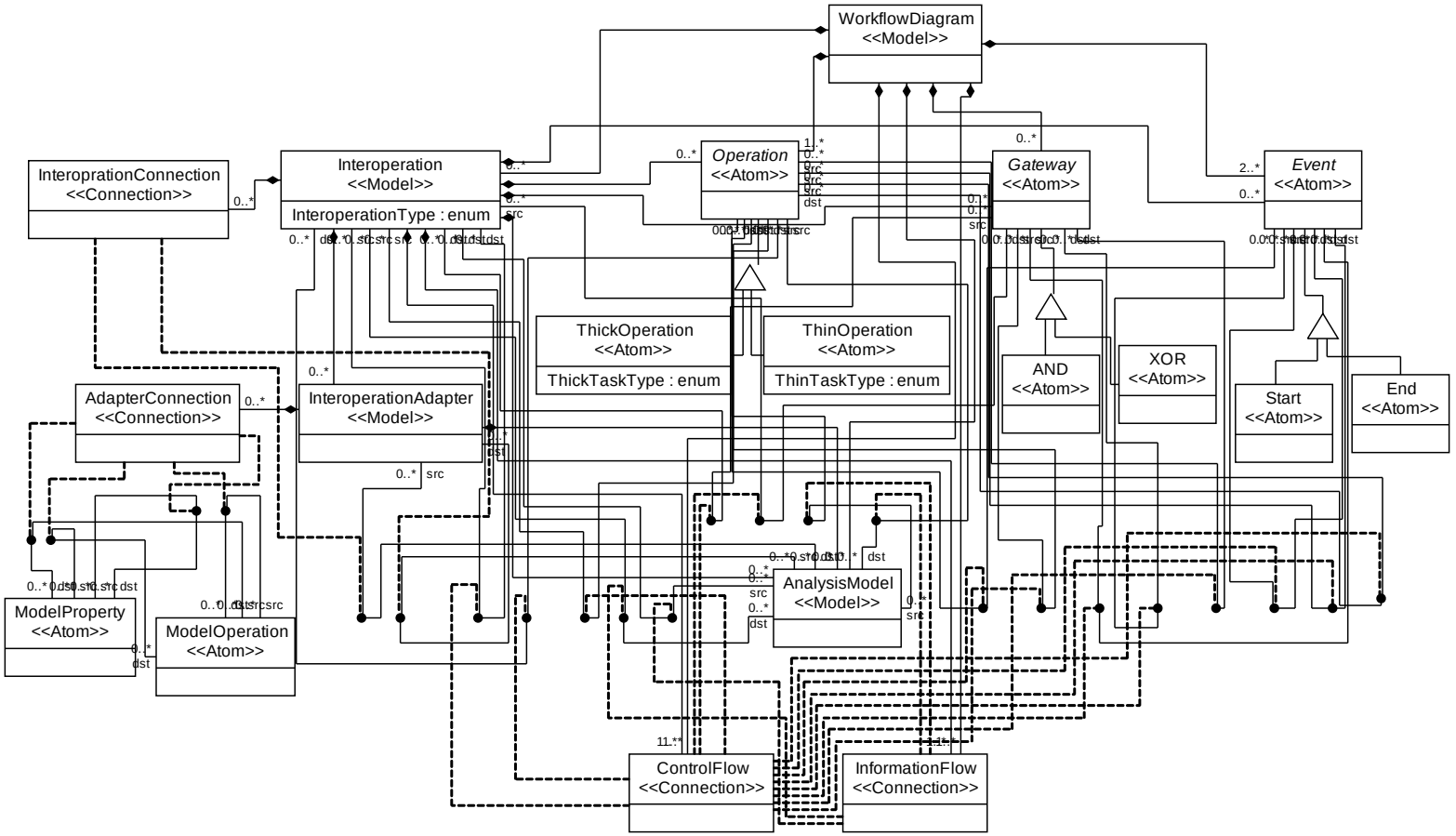

Figure 30: GME Class Diagram Aspect

One other aspect that GME provides is the Visualization Aspect. GME provides the capability of designing and then using a visual graph-based language. Since not all the elements of the class diagram aspect are for visualization purpose, a visualization aspect determines which elements are going to be visualized. In addition, each element can be customized in terms of how it should look including color, position, title, and the use of icons. Figure 31 shows an example of a Visualization Aspect for the new language. The highlighted boxes represent those elements that can be used visually while creating workflows using the new language.

Once the GME meta-model is ready and has all the constructs identified during the domain analysis, it can then be transformed into GME configuration. This is done by using the MetaGME interpreter feature provided by GME. Going back to Figure 4 in section 2.1.5, MetaGME is nothing more than the meta-modeling language shown in Layer 2 and used to define the meta-model of the new language in Layer 1. GME has incorporated into its specification the meta^2-model of the MetaGME language. When the meta-model of the new language is interpreted using the MetaGME interpreter and then registered into GME as a new language, this new configuration allows the use of GME itself to create workflows of specific multi-modeling activities using the new language. Figure 32 shows the GME interface after the paradigm has been registered and used to create a workflow model.

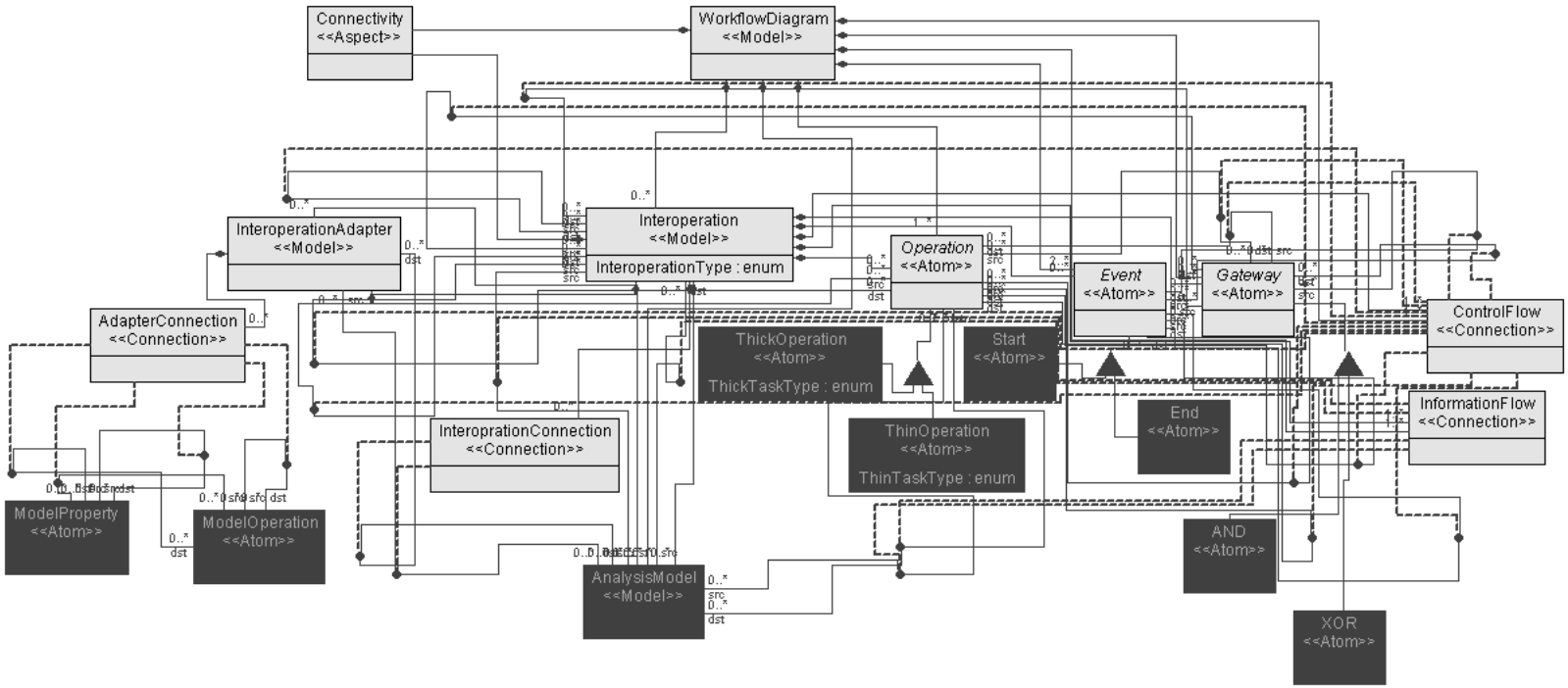

Figure 31: GME Visualization Aspect

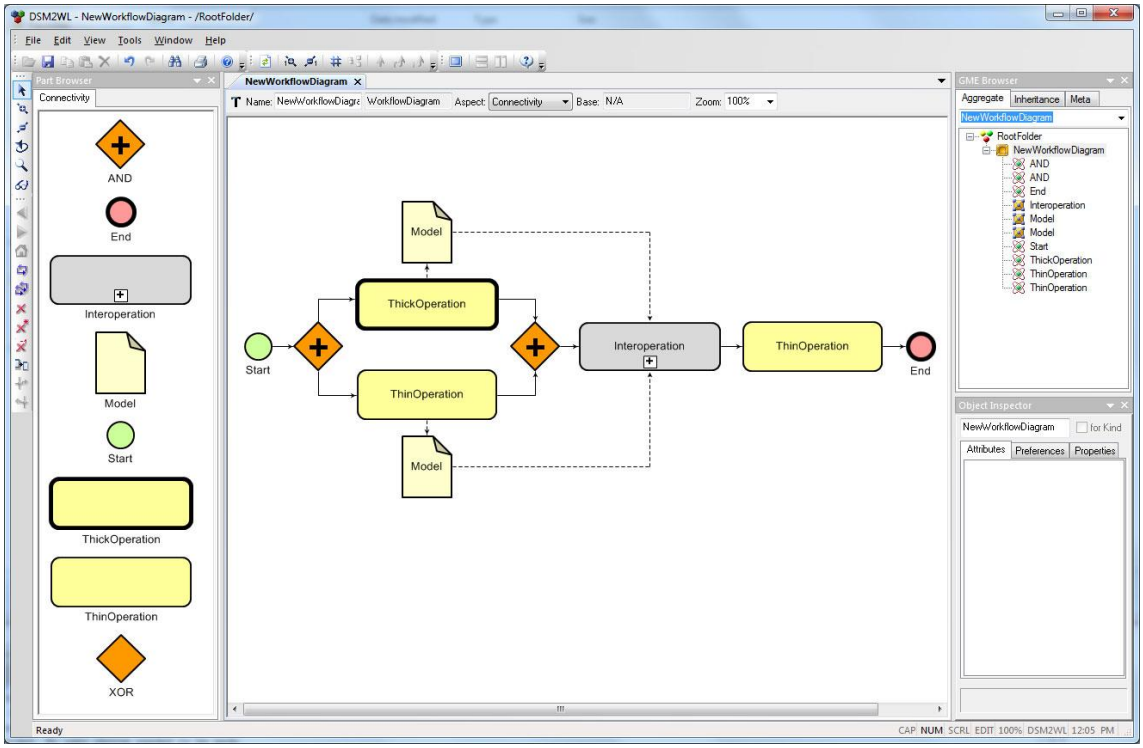

Figure 32: Using GME to Create Workflows

The process presented in this chapter outlines the steps required to configure and use GME to design a Domain Specific Multi-Modeling Workflow Language and then using GME itself to create workflow models. The process and meta-model of the workflow language represented in this chapter can be considered as a template to be used for a specific domain of interest. Domain concepts including modeling techniques and model interoperation concepts should be included in the GME paradigm once identified.

4.6 DSMWL Expressiveness

The expressibility of a domain specific language is a measure of what it can be used for. The more it can model, the greater its expressive power is. Ideally, a language should have the capability to capture all the details of the application domain. [48]

Different approaches can be followed to evaluate the expressiveness of a domain specific language [49]. One of the main drivers behind developing a domain specific language is to provide users of the language with the capability of representing domain specific concepts in a higher level of abstraction compared to other languages. Since our methodology in developing the DSMWL is based on defining a profile of a BPMN, we will assess the expressiveness of the DSMWL by comparing its basic constructs to the constructs of BPMN.

Table 5 provides a comparison between the DSMWL and BPMN. This comparison shows the level of expressiveness the DSMWL offers to its users. The main constructs of BPMN are represented, some are constrained like sup-processes, and some others are extended like the association arrows.

Table 5: DSMWL Expressiveness

DSMWL Construct	**BPMN Construct**	**Comparison**
Operation • Thin • Thick	Activity	The DSMWL defines two types of operations, thin and thick, to represent two different ways a modeling tool provide interaction with a model. In BPMN activities are used in general to represent any type of interaction in a workflow.

Interoperation	Sub-Process	An interoperation in the DSMWL is mapped to a sub-process in BPMN. It captures the interoperation between models. In BPMN sub-processes model any types of activities however in the DSMWL interoperations are required to have an interoperation adapter.
Interoperation Adapter	Sub-Process	An interoperation can be seen a specific type of a sub-process that captures the mapping between of data exchange between models.
Model	Data Object	A data object in BPMN can represent and type of data required in the modeling a business process. A model in the DSMWL represents an actual model created and analyzed by modeling tool used in the application domain.
Event	Event	The concept of events in both the DSMWL and BPMN are similar and used to represent triggers that affect the flow. They include start, end, intermediate and timer, events.
Gateway	Gateway	Gateways in both the DSMWL and BPMN share the same features. They control how sequence flows interact as they converge or diverge. Gateways include split, merge, parallel, …etc.
Control Flow	Sequence Flow	Control flow arrows in the DSMWL are mapped to sequence flows in BPMN. They connect elements of multi-modeling workflow. The use of control flows in the DSMWL is more constrained than BPMN. Some constructs like interoperation adapters are required to use a specific type of connectors.

Information Flow Adapter Connection Interoperation Connection	Association	Associations in BPMN associate flow objects including activities and sub-processes with data and artifacts. In the DSMWL three types of connectors are used to associate workflow elements with models. Information flow arrows connect models to operations or interoperations. Adapter arrows connect interoperating models. Interoperation arrows represent the mapping between data types of interoperating models.

CHAPTER FIVE: SEMANTIC GUIDANCE FOR MULTI-MODELING WORKFLOWS

As discussed earlier, our approach focuses on both the workflow and the semantic layers of multi-modeling as was shown in Figure 6. In Chapter four we discussed in detail how a domain specific multi-modeling workflow language can be developed. A multi-phase process of identifying the constructs of such language and then designing its meta-model has been presented. An important concern that drove our research in the direction of developing a domain specific multi-modeling workflow language is the ability of multi-modeling platforms users to capture model interoperations correctly with respect to the syntactic and semantic requirements of the domain of interest. Modeling techniques used to perform analysis in a specific domain can be used in some other domains. While some model interoperations are applicable in one domain, the same interoperations can have no meaning in another domain. The domain specific multi-modeling workflow language (DSMWL) of our approach aims to guarantee valid interoperations with respect to the domain being addressed.

Capturing semantic concepts of interoperations into the definition of the meta-model of the domain specific multi-modeling workflow language has many limitations. First, to embed semantic constraints in the language definition itself complicates its meta-model and decreases its flexibility. Second, embedding such constraints into the language

definition limits the possibilities of reusing a DSMWL into another application domain. In our approach, we adopt the separation of concern philosophy and use a supporting ontology to capture the semantic aspects of the domain. This ontology is then used to guide the use of the DSMWL in a way that guarantees that all created workflows are semantically correct with respect to the application domain.

In order to construct a domain ontology we utilize approaches from [14] and [34]. We discussed those approaches in detail in sections 2.2.2 and 2.2.5. The approach in [14] is based on the creation of pseudo ontologies that mirror syntactic "meta" models of modeling languages and serve as pseudo ontologies. Then, semantic concepts and relationships are added to pseudo ontologies to obtain refactored ontologies. Once a refactored ontology is completed for each modeling language, mapping of concepts across the ontologies takes place. This results in an enriched ontology that contains concepts and relationships within and across multiple refactored ontologies. In this approach, in addition to using the ontology for representing the semantics, it has also been used to represent the meta-model of the modeling language. This approach leads to a complete dependence on the ontological representation. Meta-models of the modeling languages and the application domain of interest are required to develop the new DSMWL language while the ontological representation is needed to guide the use of the DSMWL to guarantee the semantic correctness of the interconnected models.

The second approach in [34] maintains a clear distinction between meta-models and ontologies; they are different but complementary concepts, and both are needed to

allow for model interoperation. We borrow from [14] the concept of comparing ontologies (for each modeling technique) to help identify the similarities, overlaps, and/or mappings across the models under consideration and then constructing a higher level ontology that determines which sets of models can interoperate. However, we keep a clear distinction between the meta-models of modeling languages that eventually contribute to the development of the DSMWL and the ontologies that guide the creation of semantically correct workflows..

This approach of using a separate ontology to capture the sematic concepts of the domain and then to validate model interoperations in a workflow using such ontology has its own challenges. In addition to the classical problem of identifying and classifying semantic concepts of an application domain, the technical problem of using such ontology in support of a workflow language is yet to be solved.

5.1 Identification of Semantic Concepts

We address the first challenge by integrating the identification of the domain semantic concepts process into the overall domain identification and analysis processes. The semi-informal concepts captured in the form of concept maps, as discussed in section 3.1.1, represent the first step in identifying semantic concepts. In this phase all domain concepts including the concepts related to the supporting modeling techniques and their constructs are captured as well. In the Domain Analysis step that follows we begin to use these concepts to construct UML class diagrams that represent the structural aspect of the application domain and its supporting modeling techniques, and in parallel to that we

construct domain and modeling techniques pseudo ontologies. Eventually the class diagrams constitute the foundation of the DSMWL meta-model and the ontologies get mapped into a refactored upper domain ontology.

5.2 Ontology Construction

Before discussing the steps followed to construct the domain ontology, we begin by highlighting the formalism and tools we use for this purpose. Over the years, many tools and languages have been developed to capture semantic concepts and construct ontologies. In section 2.1.6 we discussed the basics of ontologies and some of the supporting tools, languages and techniques. In our approach, we decided to use the Web Ontology Language (OWL) [45] as the standard formalism to capture semantic concepts. We have also decided to use Protégé as the tool to build and analyze ontologies [46].

5.2.1 Class Hierarchy and Properties (Pseudo Ontologies)

Following the approach of [14] where pseudo ontologies were initially constructed and then refactored into an enriched unified ontology, we begin by creating a number of pseudo ontologies. Basically we create a pseudo domain ontology and pseudo modeling techniques ontologies. The first step towards creating these ontologies is based on the concepts characterized in the Domain Identification step. Concepts qualified from Table 1 and Table 2 as semantic concepts are identified as ontology classes. Classes' hierarchy is inferred from concept maps similar to the one in Figure 20. Basic properties and relations between classes are then defined. Figure 33 represents an OntoViz [50]

diagram of a pseudo domain ontology based on Table 1 and created using Protégé as shown in Figure 34.

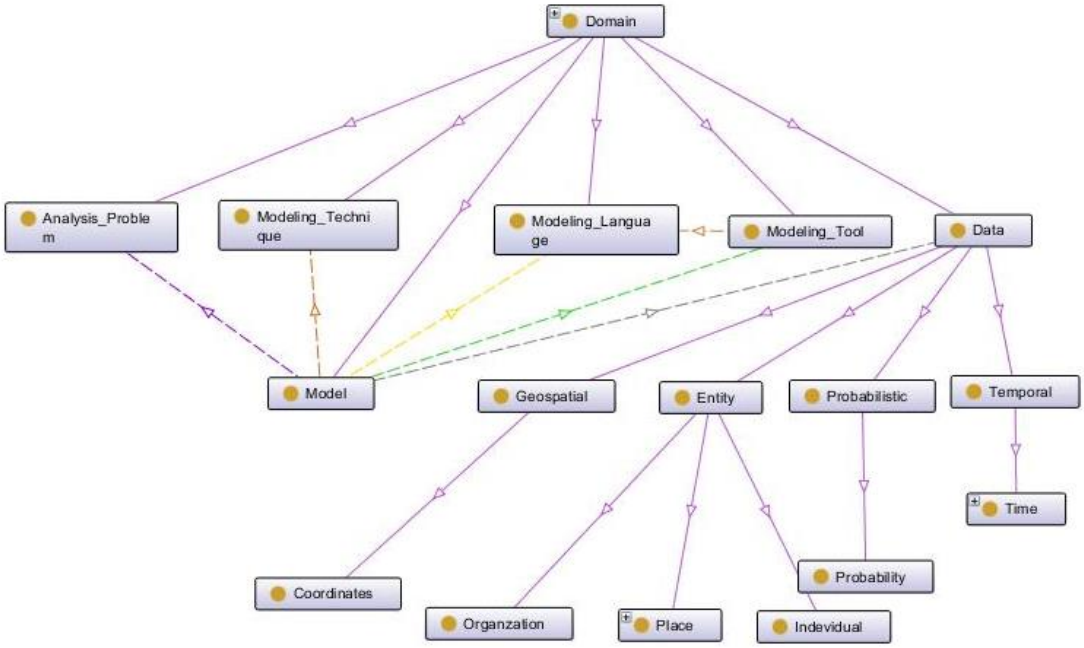

Figure 33: OntoViz Diagram of Pseudo Domain Ontology

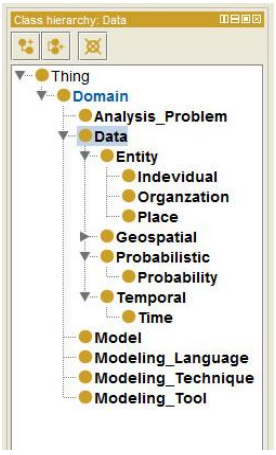

Figure 34: Protégé Class View

An OntoViz diagram of an example modeling technique pseudo ontology and a snapshot of the Protégé classes definition are shown in Figure 35 and Figure 36.

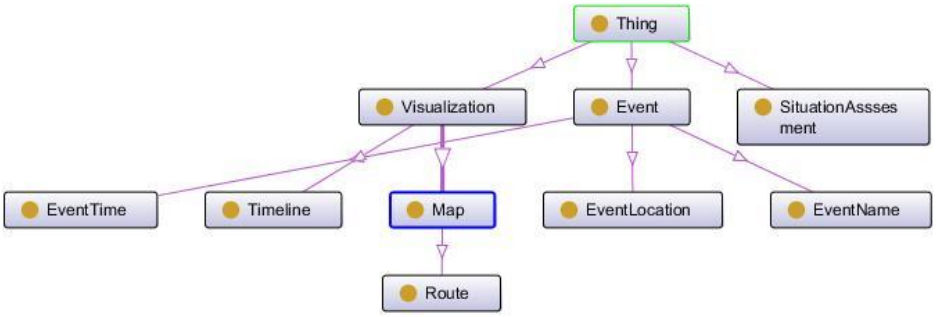

Figure 35: OntoViz Diagram of Example Modeling Technique Ontology

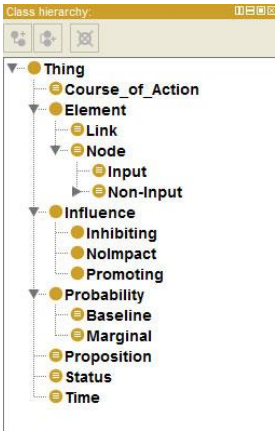

Figure 36: Protégé Class View of Example Modeling Technique Ontology

5.2.2 Refactored "Upper" Domain Ontology

Once pseudo ontologies are identified, the process of refactoring and enriching the domain ontology begins. In this step mapping of concepts from the modeling techniques ontologies to the domain ontology takes place. The aim of this process is to construct an "Upper" domain ontology that captures domain and modeling techniques concepts in addition to concepts related to model interoperation. This step is very domain specific and is focused on the problems we are trying to solve. In this section, we discuss the basics of this step; however, the examples in Chapter Seven and Chapter Eight where we show an application on a case study should help clarify this more.

In order to proceed with constructing our upper domain ontology, we employ techniques from the ontology matching and upper ontologies research domain. We discussed in section 2.1.7 some of these approaches including the classification of ontology "Alignment" mapping techniques presented in [24]. Techniques developed to address the ontology matching problem have been either manual, semi-automatic, or fully automatic. The human involvement in the loop is usually a tradeoff between precision and speed. Our approach employs a semi-automatic technique. We begin with a manual operation conducted by Subject Matter Experts (SMEs) to identify terminological, structural, extensible and semantic mappings between pseudo modeling technique ontologies and the pseudo application domain ontology. This process includes the introduction of new classes and concepts into the pseudo application domain ontology required to map concepts from the pseudo modeling techniques ontologies. This results in a list of mappings as shown in the Table 6 example.

Table 6: Pseudo Ontologies Mapping

Source	Relation	Destination
Domain Ontology Modeling Languages	Has	Modeling Language A Modeling Language B
Model Element	is similar to	Model Element
Modeling Technique Operation	result input to	Modeling Technique Operation
Model Element	value input to	Modeling Technique Operation

Relations captured in the examples provided in Table 6 are then used to refactor the domain ontology by adding classes, relations and properties. In addition to the use of OWL to capture those mappings, the Semantic Web Rule Language (SWRL) [51] is used to capture rules that govern the mappings between concepts. Once all the mapping concepts and relations are captured manually, a reasoner is executed to infer all mappings and relations based on the identified relations and rules. The use of the reasoner represents the automated part of the mapping process. The final resulting ontology is our desired enriched upper domain ontology. Figure 37 represents an OwlViz [52] diagram of an upper domain ontology.

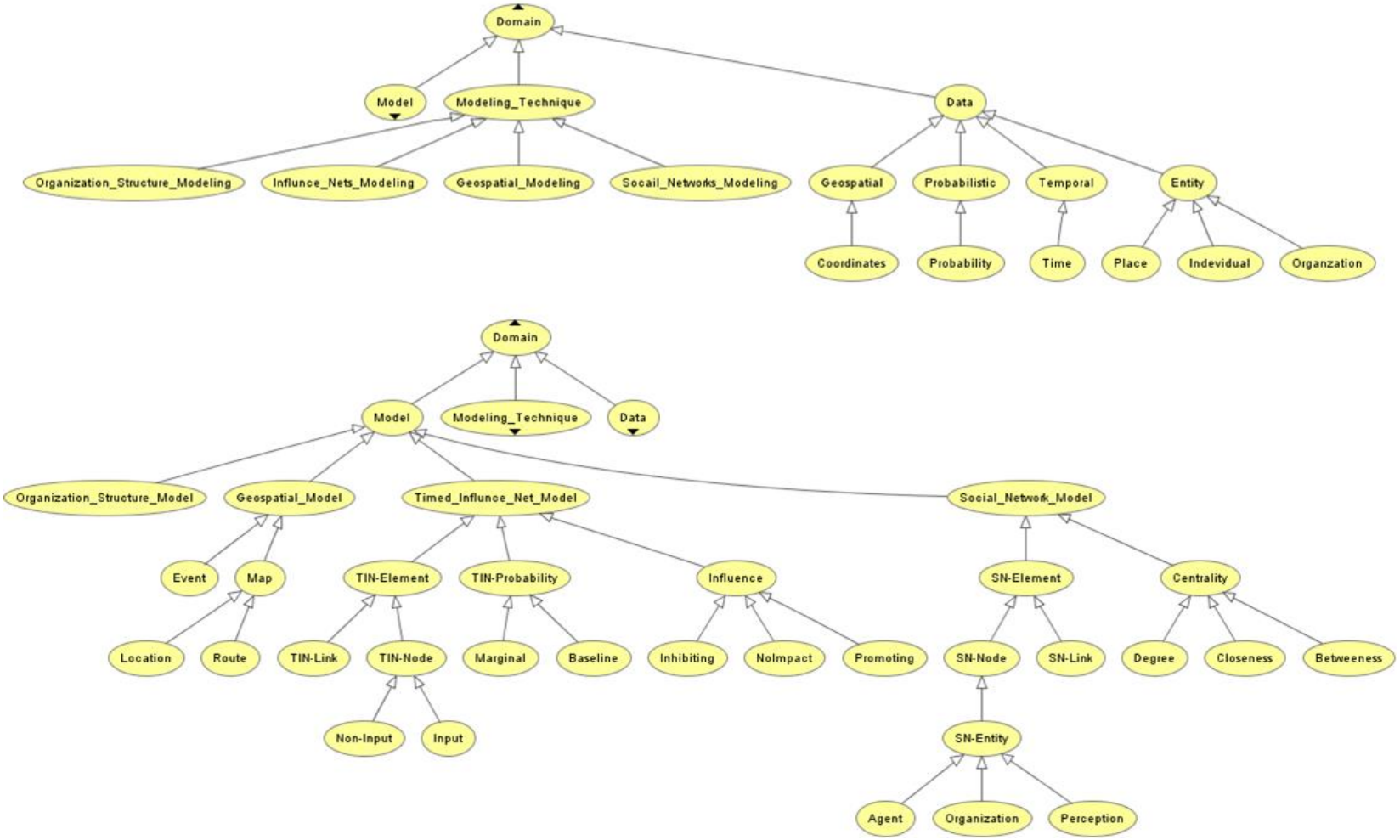

Figure 37: Example Upper Ontology

5.3 Semantic Guidance of Multi-Modeling Workflows

In this section and its sub-sections we discuss the details of how the domain ontology is used to support the creation of semantically correct multi-modeling workflows. The aim of our approach is that the user of the DSMWL should be capable of creating workflows that capture model interoperations while guaranteeing the semantic correctness of the resulting workflows.

Since we are using GME to create multi-modeling workflows, and since our final domain ontology is captured in OWL, there should be a way to allow communication between GME and the domain ontology while creating workflows for specific application domain problems. This communication is required to make sure that interoperations captured in any workflow are semantically correct. Checking the domain ontology for the correctness of workflow interoperations requires the capability of querying the ontology for such information. There exist techniques that allow for executing queries on OWL based ontologies [53]. Among these techniques, the Ontology Query Language (SPARQL) [47] is utilized in our approach.

5.3.1 Querying the Ontology

As any other language, SPARQL requires a platform that allows its use to query ontologies. Our final domain ontology is in OWL and we use Protégé to create and operate on it. There exist Protégé plugins that provide the capability of executing SPARQL queries on its OWL ontologies; however, we still need to integrate this OWL querying capability with the multi-modeling workflow creation platform, GME in our

case. For this reason, we investigated the use of a SPARQL query engine that allows this kind of integration. We found our desired solution in Fuseki [54], a SPARQL query server that utilizes Jena's [55] SPARQL query engine.

Fuseki provides **Re**presentational **S**tate **T**ransfer (ReST) style [56] services over HTTP. It can be seen as a SPARQL server that exposes query functionalities as services. As part of the service request, a path to the OWL ontology to be queried is passed in addition to the query itself. This mechanism fits our approach as it leverages the use of an existing service-based SPRAQL server. It also works with GME, the platform we use to create multi-modeling workflows as described in the following section.

5.3.2 Extending the GME for Semantic Support

As mentioned earlier, in order to utilize the domain ontology while creating multi-modeling workflows to guarantee semantic correctness of interoperations between models, the ontology should be queried on the validity of workflow interoperation connections. In other words, this means that whenever a connection on the interoperation level of a multi-modeling workflow is created a query has to be formulated and passed to the ontology query engine. We have already discussed the use of SPARQL as ontology query language and Fuseki as the query server/engine. We use GME to create multi-modeling workflows. The challenge now is to find a way that GME itself can react to the creation of interoperation connections, formulate a SPARQL query based on the entities participating in that connection, and then have that query executed on the supporting

domain ontology using Fuseki. In addition to that, GME should be capable of parsing the result of the query and decide whether the connection under consideration is valid or not.

The selection of GME as the tool to develop and then use the DSMWL wasn't arbitrary. It's not only powerful in setting up the environment for a new Domain Specific Multi-Modeling Workflow Language including the definition of its meta-model, but it is also an extendable platform that allows for different types of extensions to the environment; basically using Plug-ins or Add-ons [35]. GME Plug-ins are used to provide useful additional functionality in order to ease working in GME. Add-ons on the other hand are used to react to GME-events and provide advanced capabilities that match our need for the purpose of semantically guiding workflow creation.

Utilizing these GME extensibility features and in order to address the semantic guidance issue, we implemented a GME Add-on extension. This extension reacts to GME events and, in the case of creating any connection that represents interoperation between models, it formulates a SPARQL query that is executed on the supporting ontology. The details and code used to develop this extension are shown in Appendix A.

5.3.1 The Overall Ontology Guidance Process

The overall ontology guidance process is shown in Figure 38. The process spans two lanes that represent two integrated platforms. The upper lane represents the GME while the lower lane represents Fuseki, the SPARQL server. Each lane shows the sequence of task associated with it.

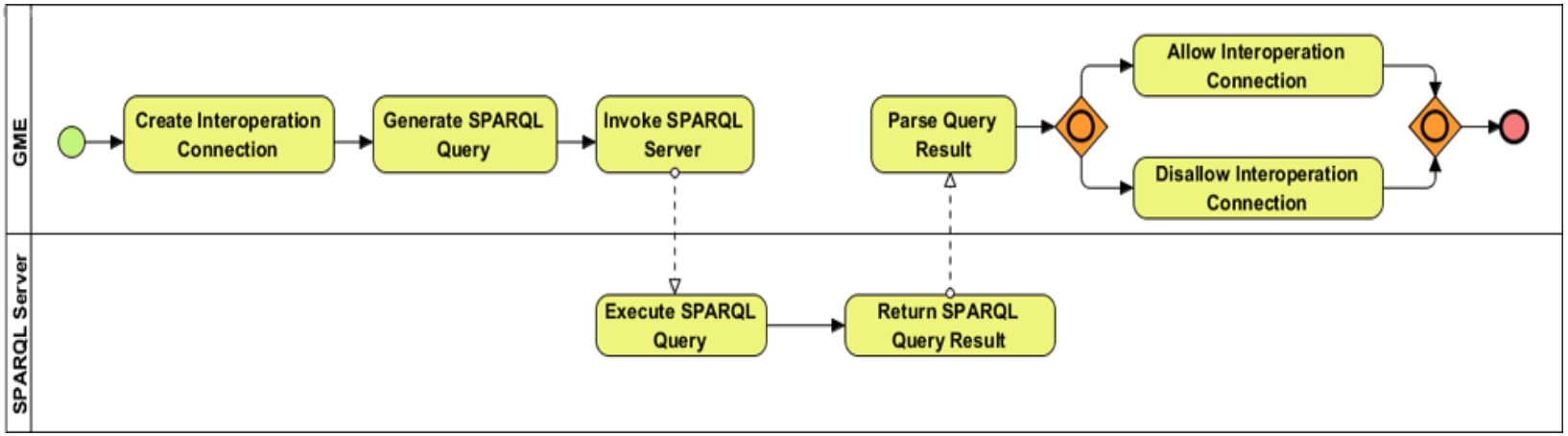

Figure 38: Semantic Guidance Process

Now that GME is configured and ready to be used for creating multi-modeling workflows including the setup of semantic guidance extension, any interoperation connection results in triggering the Add-on. The Add-on responds to any interoperation connection by doing the following:

- The entities involved in the connection are examined and a SPARQL query is generated. Figure 39 shows an example of a SPARQL query. The query checks if two elements, Element_A and Element_B, are of the same type.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX this: <http://www.owl-ontologies.com/DomainOntology.owl#>
SELECT ?x
WHERE {
this:Element_A rdf:type ?x.
this:Element_B rdf:type ?x }
```

Figure 39: SPARQL Query Example

- The SPARQL server query service is then invoked and the query is passed.
- The SPARQL server executes the query on the supporting domain ontology.
- The SPARQL server returns the query result to GME.
- The result is then parsed by the GME Add-on.
- Based on the query result, the connection under consideration is allowed or not.

CHAPTER SIX: IMPLEMENTING A WORKFLOW

Our approach focuses on the development of a domain specific multi-modeling workflow language capable of capturing multi-model interoperation activities. The aim of our approach is to help users of multi-modeling platforms in visually creating semantically correct workflows that capture multi-modeling activities that address complex analysis problems.

We talked earlier about the four layers of multi-modeling shown in Figure 6. We have also pointed out that the focus of our approach is the top two layers, the workflow and the semantic layers. While multi-modeling platforms generally provide the infrastructure for the lower two layers where the actual integration of modeling systems takes place, they usually lack the capability of capturing workflows of semantically correct model interoperation as discussed in section 2.2.1. Our approach complements such platforms by providing a methodology that leads to introducing a higher workflow level.

Our approach also addresses the implementation of the created workflows. Development of a multi-modeling platform that implements multi-modeling workflows is outside the scope of this research. We consider our approach as a higher level technique that can be integrated with existing platforms. The rationale behind this is that each

multi-modeling platform has its own architecture and depends on a lower level execution mechanism. In order to implement a workflow created using our approach, a transformation to the target implementation platform has to happen first; we call this Workflow Interpretation. The fifth and final step in our methodology addresses the tasks required for workflow implementation. In this chapter we discuss these tasks in detail.

6.1 Workflow Implementation Overview

In order to clarify how our approach can fit as a higher level on top of an existing multi-modeling platform, we discuss an example of a Service Oriented Architecture (SOA) based multi-modeling environment as shown in Figure 40. The architecture of such environment consists of two major parts. The top part is called the Workflow Generator; this is where our approach plays a major role. For any multi-modeling based analysis problem, we begin by identifying the domain to which this problem belongs. This is followed by developing the Domain Specific Multi-Modeling Workflow Language (DSMWL) and a supporting Domain Ontology, both used to create semantically correct workflows of multi-modeling activities.

Implementing a workflow requires the transformation (interpretation) into a format that can be executed in an implantation environment. This is what the lower part of the architecture shown in Figure 40 is expected to handle. In this example, the lower part is SOA based where participating modeling systems expose their features and functionalities as services. The use of these services to implement a multi-modeling workflow is managed and orchestrated by an Enterprise Service Bus (ESB).

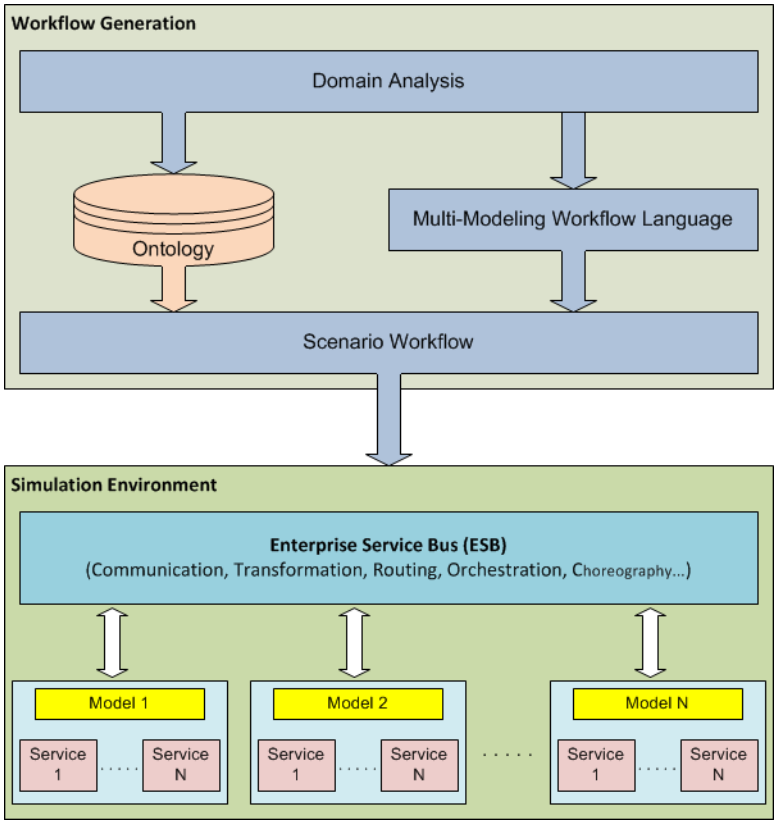

Figure 40: SOA Based Implementation Platform

One example SOA based environment that can be used to implement multi-modeling workflows is SORASCS [2]. Its high-level architecture is close to the lower part of Figure 40. Different modeling and simulation techniques can be integrated into SORASCS by adding their features as thin or thick services. SORASCS utilizes the Business Process Execution Language (BPEL) [16] to orchestrate the invocation of services. Since we proposed basing the domain specific multi-modeling workflow language on BPMN, which can be mapped to BPEL [57], generated workflows can be transformed to BPEL and then executed on SORASCS.

6.2 Workflow Interpretation

Workflows in our approach are created using GME after configuring it to use the developed DSMWL. Implementing the workflow on any platform requires that we transform these workflows into the desired platform execution language. GME provides the capability of integrating interpreters that can access its model's data, a workflow model in our case, and interpret the data into a code or language. This feature of GME allows us to develop an interpreter to transform workflows into the language required by an implementation platform to execute the workflow.

6.2.1 Implementation Platform Selection

Before we begin developing a GME interpreter for our workflows, a decision has to be made on which implementation platform is going to be used. This is an important step since the GME interpreter will be transforming workflow models into a format (language) that the implementation platform understands and uses to execute. This step includes the analysis and understanding of the structure and components of the implementation platform language and then mapping them to the constructs of the DSMWL.

As part of this research SORASCS was selected to be the implementation platform. In Chapter Four an initial and basic set of elements that constitute a DSMWL based on BPMN was proposed. In Table 7 a mapping between that initial set of elements and the constructs of BPEL, the language used by SORASCS is presented. Concepts of BPMN and BPEL mapping from [57] and [58] are utilized.

Table 7: Mapping the DSMWL to BPEL

Workflow Language Construct	BPEL Construct
Workflow Diagram	Process
Operation	Invoke
Model	Variable
Start Event	Receive
End Event	Reply

6.2.2 GME Interpreter

Once an implementation platform is selected and a mapping between the DSMWL and the platform execution language is defined, a GME interpreter can be developed. GME interpreters are not standalone programs; they are components that are loaded and executed in GME upon a user's request [59].

In order to develop a GME interpreter, a GME component has to be created and configured as GME Interpreter. GME provides tools that allow for creating and configuring such component. Creating the interpreter component includes the use of GME's Builder Object Network (BON). BON provides the capability to create objects for each of the elements in a GME workflow model. Access to these objects and relationships between them is available through methods that act on these objects. Once the interpreter component framework is ready, the interpreter code is written. The code basically uses BON to traverse the workflow model objects tree and create an equivalent representation in the desired execution language. [59]

Appendix B includes an interpreter code used to interpret workflow models created using the domain specific multi-modeling workflow language developed for the case study discussed in Chapter 7 into SORASCS implementation.

6.2.3 Workflow Execution

After building a GME interpreter and having GME configured to use it for the developed DSMWL, the interpreter can be used whenever a workflow model is created and ready to be executed. The interpreted workflow can then be loaded into the implementation platform to be executed. It is then the responsibility of the platform to orchestrate the sequence of activities and operations performed by the participating modeling techniques.

6.3 Closure

In Chapter Three, an overview of the multi-modeling approach was presented. In Chapters Four and Five the approach was discussed in details showing how a DSMWL can be developed. It was also shown how the use of such language can be supported by a domain ontology to guarantee capturing semantically correct multi-model interoperations while creating multi-modeling workflows. In Chapter Six the implementation of a DSMWL was discussed.

In the next two chapters, a case study is presented in which the methodology steps are followed in developing and then using a DSMWL to perform multi-modeling based analysis in a selected application domain.

CHAPTER SEVEN: CASE STUDY APPLICATION DOMAIN

The case study presented in this chapter involves a decision making problem in the Drug Interdiction application domain. In this case study we discuss a drug trafficking scenario that is being monitored and analyzed by a drug interdiction agency such as the Joint Interagency Task Force South (JITF-South). During the events of this scenario the agency receives substantial amounts of data. The analysis of such data requires the use of various modeling techniques. Traditionally, the agency analysts would use available modeling capabilities separately. In order to analyze data quickly and effectively, models will be used collaboratively; models complement or supplement each other for the sake of better and effective analysis results.

The use of multi-modeling platforms becomes a necessity so that agency analysts can perform multi-model based analysis. A major barrier usually associated with using such platforms is the level of technical experience required to integrate different modeling techniques into the platform. This maps to the physical and syntactic layers of Figure 6 and has been addressed from different perspectives in the research community. Platforms like the C2WT and SORASCS provide means of facilitating the integration of modeling techniques into their environment as shown in section 2.2

We hypothesize that for JIATF-South analysts, the use of any multi-modeling platform poses additional challenges. Capturing the sequence of multi-model interoperation activities in a form that analysts can easily visualize and implement is of great importance. This can be thought of as a high level representation of the multi-modeling analysis activities that is yet to be done using a formalism that can be implemented on the selected platform; a multi-modeling workflow layer is required. Another challenge is that modeling techniques are applicable to various domains. The application of a modeling technique in a specific domain might impose some constraints on the interoperations with models of other techniques. These constraints are usually both syntactic and semantic. It is highly important that analysts have a means to guide them on the valid multi-model interoperations while utilizing multi-modeling.

While the actual physical integration of modeling techniques is outside the scope of this dissertation, the proposed Domain Specific Multi-Modeling Workflow Language approach plays a major role in the selected case study. The approach provides a systematic methodology for allowing JIATF-S analysts to create semantically correct multi-modeling workflows for different analysis scenarios.

We begin this chapter by first introducing the Joint Interagency Task Force -South (JIATF-South). We then show how the methodology proposed by this dissertation can be applied to improve the overall multi-modeling experience within the agency. This includes a detailed domain identification process followed by the development of DSMWL and domain ontology. We then show how the new language and the ontology

can be used to create workflows of multi-model based decision making process. In the following chapter we present a scenario of a drug trafficking incident in which JIATF-South analysts employ multi-modeling to analyze data and to perform effective decision making.

7.1 JIATF-South

The Joint Interagency Task Force - South (JIATF-South) is a drug interdiction agency well known for interagency cooperation and intelligence fusion [8]. The agency usually receives disparate data regarding drug trafficking and drug cartels from different sources. Quick and effective analysis of data is essential in addressing drug trafficking threats effectively.

A typical case begins with JIATF-South receiving information form the Drug Enforcement Administration (DEA). This prompts the deployment of Unmanned Airborne Vehicles (UAVs) that subsequently detect and monitor a suspect vessel until JIATF-South can sortie a Coast Guard cutter to intercept. If drugs are found, jurisdiction and disposition over the vessel, drugs, and crew are coordinated with other agencies. [8]

The history of JIATF-S is full of successes and failures, but over time, efficient collection and integration of intelligence sources was a major factor in its success. The significant improvement of handling and analyzing intelligence information provided decision makers with the ability to produce timely and effective Courses of Actions (COAs). In another words, the activities of JIATF-South represent a decision making problem in the drug interdiction domain. Courses of Actions - identified by the agency -

are dependent on efficient analysis of received data. Effective analysis of disparate data requires the use of multiple modeling techniques, i.e. Geospatial Modeling, Social Networks Modeling, etc. This is a challenging task; no single modeling technique is capable of presenting an overall assessment of the situation. Interoperations between models bridge some gaps and help analysts in understanding the overall situation. This represents a typical multi-modeling problem for a specific domain. A high level workflow representation of multi-modeling activity facilitates the analysts' task. Such workflow could be reused for future missions.

7.2 Decision Making Process

In such evolving situation, correct decisions have to be made quickly. JIATF-S gathers information about the ongoing drug trafficking activities and should be capable of making a decision on the best Course of Action (COA) based on the available information. This decision making process cannot be arbitrary and has to be based on extensive data analysis. In addition to temporal and geospatial data, the agency receives information about the connections and relationships between key persons in the involved drug cartels. The variety of data the agency receives allows for the use of different analysis techniques. For example the geospatial information can be used to create geospatial models and perform analysis on these models. Relationships between key persons can be used to create and analyze social networks and organizational structure models. While each of these techniques provides enormous capabilities, a comprehensive analysis of the situation requires the interoperation and exchange of data between models. For an effective decision making process JIATF-S analysts could employ multi-modeling

based analysis. This includes the use of a number of modeling techniques and a multi-modeling platform that allows models to interoperate. The rationale is that while each modeling technique might be capable of capturing certain aspects of the available data, interoperation between models should improve the results of the overall process.

7.3 Modeling Techniques

Before we discuss how multi-modeling can help in the decision making process, we first postulate which modeling techniques the agency analysts use. These analysts are experienced in performing data analysis using a variety of techniques. In many cases they pull some existing models and load them with updated data based on an evolving situation. A major factor that determines what kind of techniques to use is basically the type of information under consideration. In the case of drug interdiction related information, temporal, geospatial, and social connections information are of the most important types of information.

7.3.1 Timed Influence Nets

Influence Net modeling is based on two well established techniques, Bayesian Nets and Influence Diagrams. Influence Nets utilize directed acyclic graphs as the Bayesian Nets. They are used for probabilistic modeling of the rationale of some group or organization. In an Influence Net model, nodes represent random variables such as beliefs, actions, events, etc., whereas edges represent causal relationships (influence) between nodes. The parent and child nodes are often called cause and effect, respectively. The causal relationship between a cause and an effect can either be promoting or

inhibiting as identified by the edge terminator (arrow head or filled circle) as shown in Figure 41. [14]

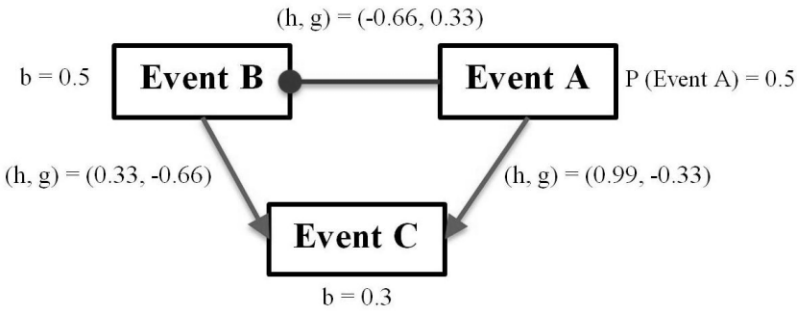

Figure 41: Example Influence Net [15]

Influence nets make use of the CAST Logic algorithm which defines influence parameters *h* and *g* assigned to each link (edge) between the two nodes. These parameters define the causal strength of the influence between the cause and effect nodes. Parameter *h* models the case when occurrence of a cause would increase the likelihood of occurrence of the effect whereas parameter *g* models the case when non-occurrence of cause would increase the likelihood of occurrence of the effect. [61]

Timed Influence Nets is a modeling extension of original Influence Nets that allows the modeler to allocate time delays associated with nodes and edges, representing an impact of events (actions or effects) that takes some time to reach and be processed by the affected events or conditions. For example, the third value assigned to edges in Figure 41 represents time delays in time units. Consequently, Time Stamps are associated with each node (including the action nodes). Hence, a user can specify a Course of Action

(COA) as a time sequence on the action nodes, the effects of which are propagated through the network and trigger changes to the probability values of the affected nodes. The change in probability value of a desired effect (leaf node) can be observed over time. A tool called Pythia developed at the System Architecture Laboratory at George Mason University supports modeling of Timed Influence Nets (TINs). It also supports analysis techniques such as Sensitivity Analysis, Course of Action Analysis, and the Sequence of Actions Finder (SAF) algorithm. [60] [61]

7.3.2 Social Networks

Social Networks modeling is a hot research area. In this dissertation we adopt the definitions and techniques of Carley [11] where a Social Network is defined to be a structure composed of real world entities and associations among them. An entity, or a node, can be an agent, organization, action, knowledge, and/or resource. The term structure more specifically refers to social structure which is a concept in sociology that defines an enduring relationship between real world entities. The resultant structure yields a graph-like formalism as shown in Figure 42 which has the properties of a typical graph that also considers other measures such as density, betweenness, closeness, and degree centralities etc. [11][60]

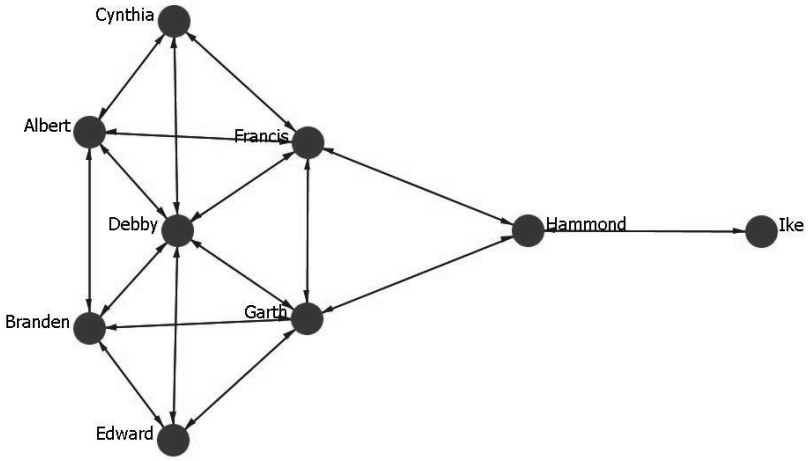

Figure 42: Example Social Network [11]

The graphical form of the social network can also have a matrix representation. ORA, a tool developed at Carnegie Mellon University, makes use of the matrix form to represent social networks. Single-mode matrices represent networks containing only one type of entities (e.g., persons only) while multi-modal matrices consider networks with multiple types of entities (e.g., persons, tasks, knowledge, resources etc.). ORA also implements a large set of social network measures. Network level measures provide information about the network such as network density or diameter. Node level measures provide information about a specific node in the network such as degree (in/out), betweenness, and closeness centralities. [11][60]

7.3.4 GIS Modeling

Geospatial intelligence has become an essential requirement in successful decision making processes. To support such intelligence, geospatial modeling techniques are required. These techniques should allow professionals to quickly and easily navigate

through massive volumes of spatial data. The most used application of geospatial modeling is generating interactive maps that allow for visually tracking and analyzing evolving situations. There exist many tools that support this kind of analysis. In our research we use a tool developed by ISS (Intelligent Software Solutions) called WebTAS (Web enabled Temporal Analysis System). Figure 43 shows an example of WebTAS models. [62]

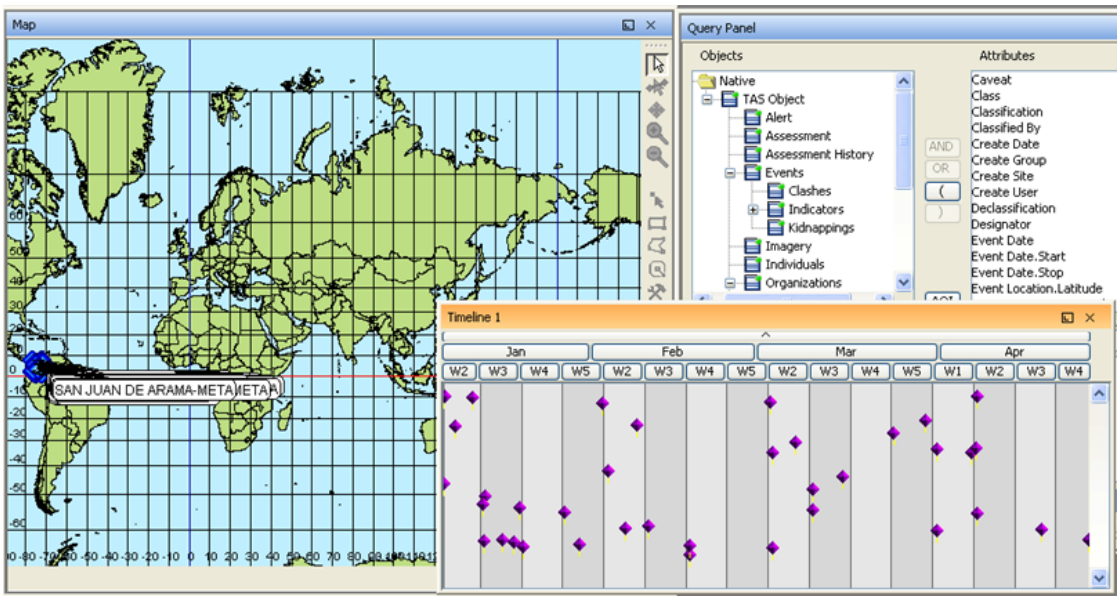

Figure 43: Example of WebTAS Models

WebTAS is a tool suite for fusion of large amounts of disparate data sets, visualization, project organization and management, pattern analysis and activity prediction, and contains various data presentation aids and tools. WebTAS provides sophisticated data query and visualization capabilities that allow users to easily discover

and comprehend complex relationships between large data sets and to perform ad-hoc queries against those disparate data sources. It also provides a situation assessment tool called KPASA which allows for generating alerts based on preset situation threshold limits. [62]

7.4 Applying the Multi-Modeling Approach

In order for the JIATF-S to perform an efficient multi-model based analysis we apply our multi-modeling methodology. We show in the following subsections how each step of our approach is applied to the domain and the modeling techniques used by the agency analysts. A Domain Specific Multi-Modeling Workflow Language based on the framework presented in Chapter Four is developed for this case study in addition to a Domain Ontology.

7.4.1 Domain Identification

We first begin with an informal description of the domain. Looking back at the way JIATF-South operates as described in section 7.1 and the scenario discussed in 7.2, the following list of statements describes the main concepts of the domain.

- Drug Interdiction involves information sharing, fusion of intelligence data and monitoring of drug trafficking activities.
- Given (incomplete and uncertain) information, decisions to be made on best COAs.
- Drug Interdiction involves dealing with Drug Cartels and Smugglers (RED groups) and Law Enforcement and Intelligence (Blue groups).

- Drug Smuggling takes different routes and originates from different sources.
- JIATF-S Analysts use Social Networks, Geospatial models, Timed Influence Nets, Organization Structures and Asset Allocation Modeling techniques.

These informal statements are then revised to scope the domain and exclude any concepts that are outside its boundary. In this example, the asset allocation and the scheduling problems are not addressed. A repository of related concepts is then identified. Table 8 shows examples of related concepts. The concepts are classified into two major categories, Domain Concepts, and Modeling Techniques Concepts.

Table 8: Domain Concepts

General Domain Concepts	
General Concepts	**Specific Concepts**
Blue Organizations	JIATF-South DEA (Drug Enforcement Administration) CBP (Customs and Border Protection) CG (Coast Guard) US Navy
Red Organizations	US Drug Trafficking Groups Latin America Drug Trafficking Groups
Data/Information	Geospatial Time Individuals Relationships Laws (Laws of the sea)
Routs	Land Sea
Decisions	Deployment of forces Stopping Vessels Carrying Drugs Arresting COAs

Table 9: Modeling Techniques Concepts

Modeling Techniques Concepts	
General Concepts	**Specific Concepts**
Social Networks	Link Node Graph Matrix Entity/Actor Agent Degree Centrality Betweeness Centrality Closeness Centrality
Influence Nets	Mathematical Technique Influence Proposition Bayesian Networks Inhibiting/Promoting Node (Input/Non-Input)Link Probability Sensitivity Analysis COA Analysis Sequence of Action Finder (SAF)

After identifying related concepts, we construct concept maps to capture the relations between these concepts. Concept maps are generally constructed to answer specific questions in the domain of interest. Figure 44 shows a concept map that addresses the question: How does JIATF-South perform drug interdiction? The same applies to other aspects of the domain and the modeling techniques. As part of the process, we refer to similar experiences and make use of existing assets. In [14], concept maps for Influence Nets and Social Networks were constructed. Appendix C includes the concept maps created for this case study.

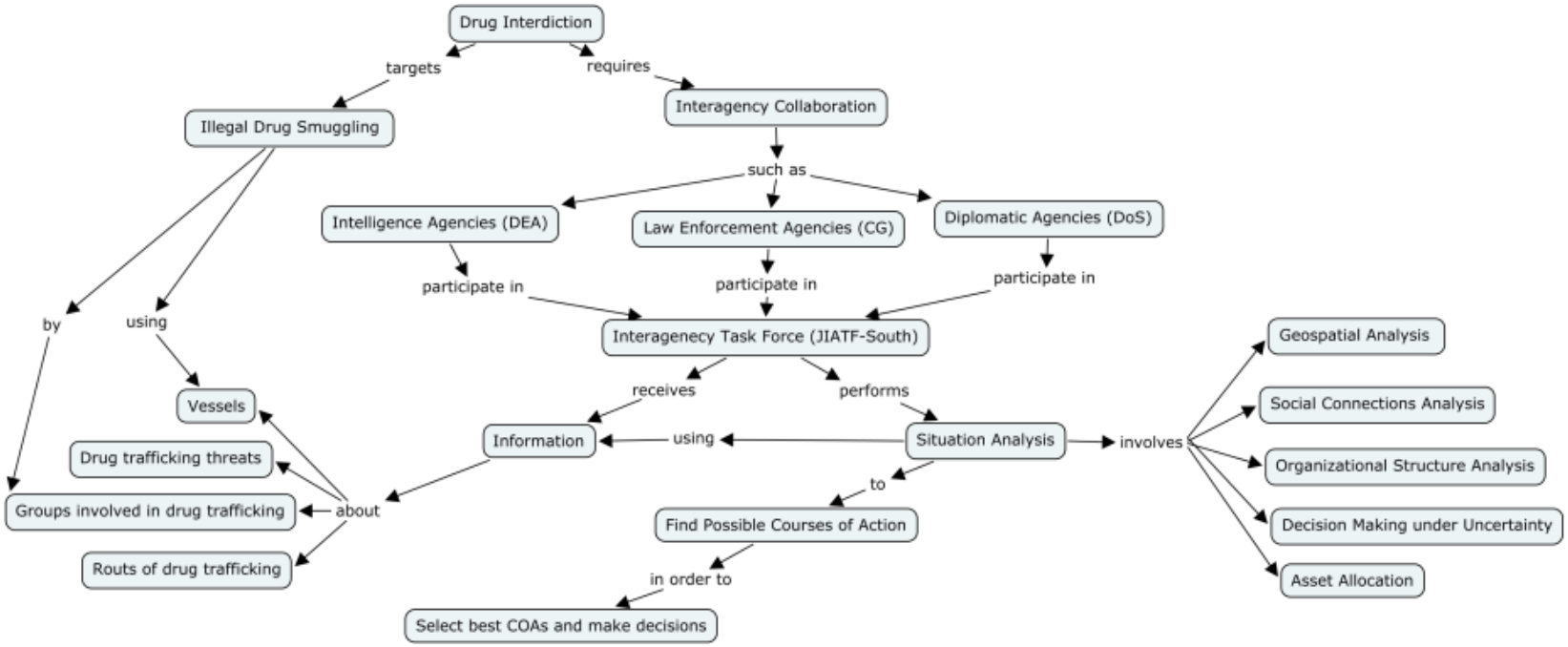

Figure 44: Concept Map (How does JIATF-South perform Drug Interdiction?)

7.4.2 Domain Analysis

In this step, we use the concept maps created as part of the domain identification step to perform domain analysis. We construct UML class diagrams to represent the constructs of the domain and the modeling techniques. Parallel to that, we identify semantic concepts and relations and capture them in OWL ontologies. In Figure 45, we show a high level UML class diagram that represents the constructs of the drug interdiction domain. Appendix C includes the UML class diagrams used to create the high level UML class diagram for this case study.

The process of capturing semantic concepts and then creating OWL ontologies discussed in Chapter Five is applied to create a high level ”upper” Domain Ontology for this case study. Figure 46 represents a visualization of this high level ontology. Appendix C includes the OWL ontologies used to create the high level domain ontology for this case study.

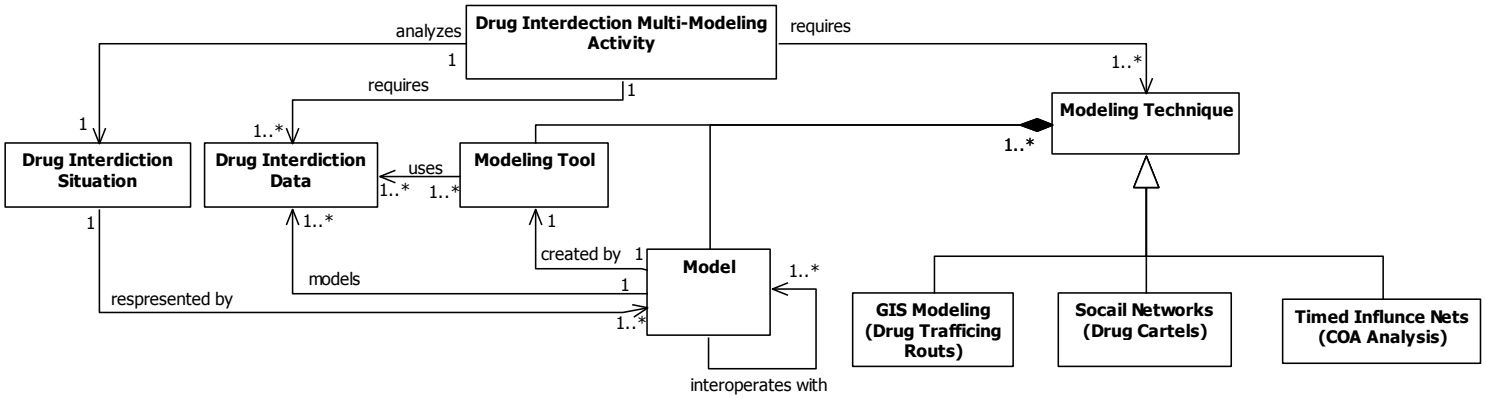

Figure 45: UML Class Diagram (Domain Concepts)

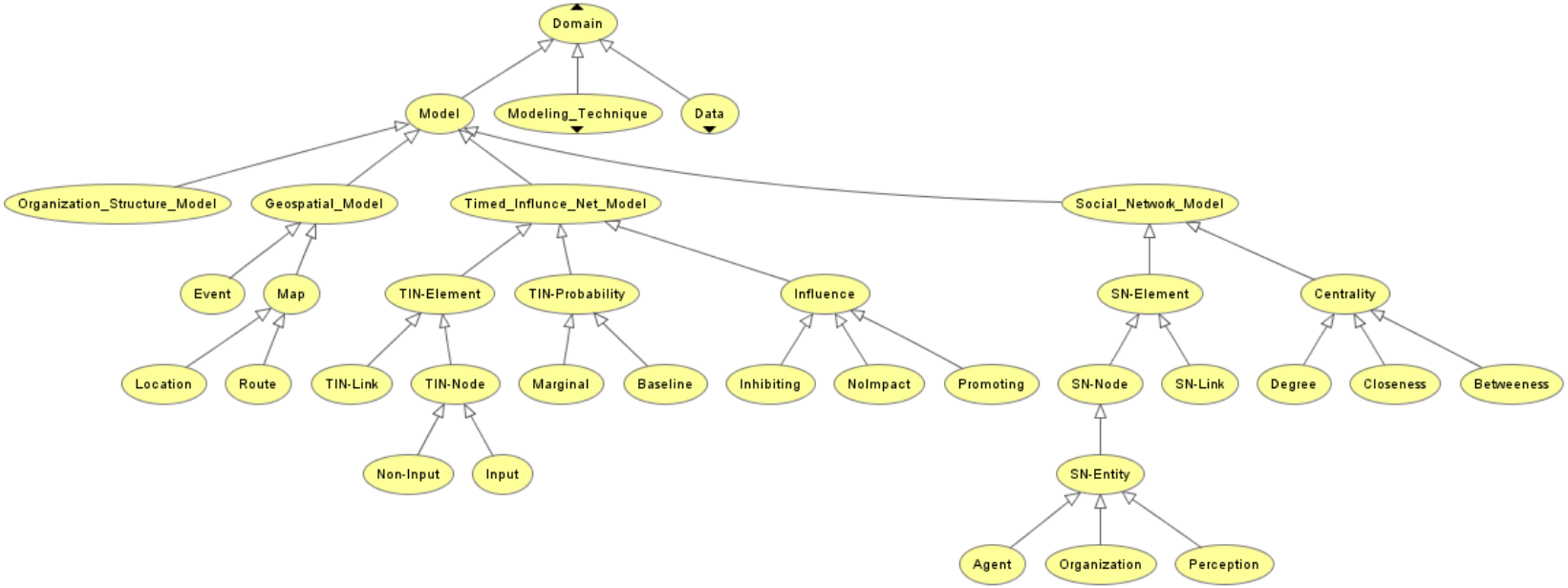

Figure 46: Domain Ontology

7.4.3 Domain Specific Multi-Modeling Workflow Language

Based on the products of the domain analysis step, the constructs of the new language are identified. The new language can be based on an existing language by extending/constraining it. As discussed in Chapter Five, we selected BPMN as the base language and we have already identified basic constructs of a domain specific multi-modeling workflow language. For this case study, we continue to use those identified constructs.

Using GME, a Meta-Model for the Domain Specific Multi-Modeling Workflow Language is developed as shown in Figure 47. Appendix C includes snapshots of GME meta-model configuration.

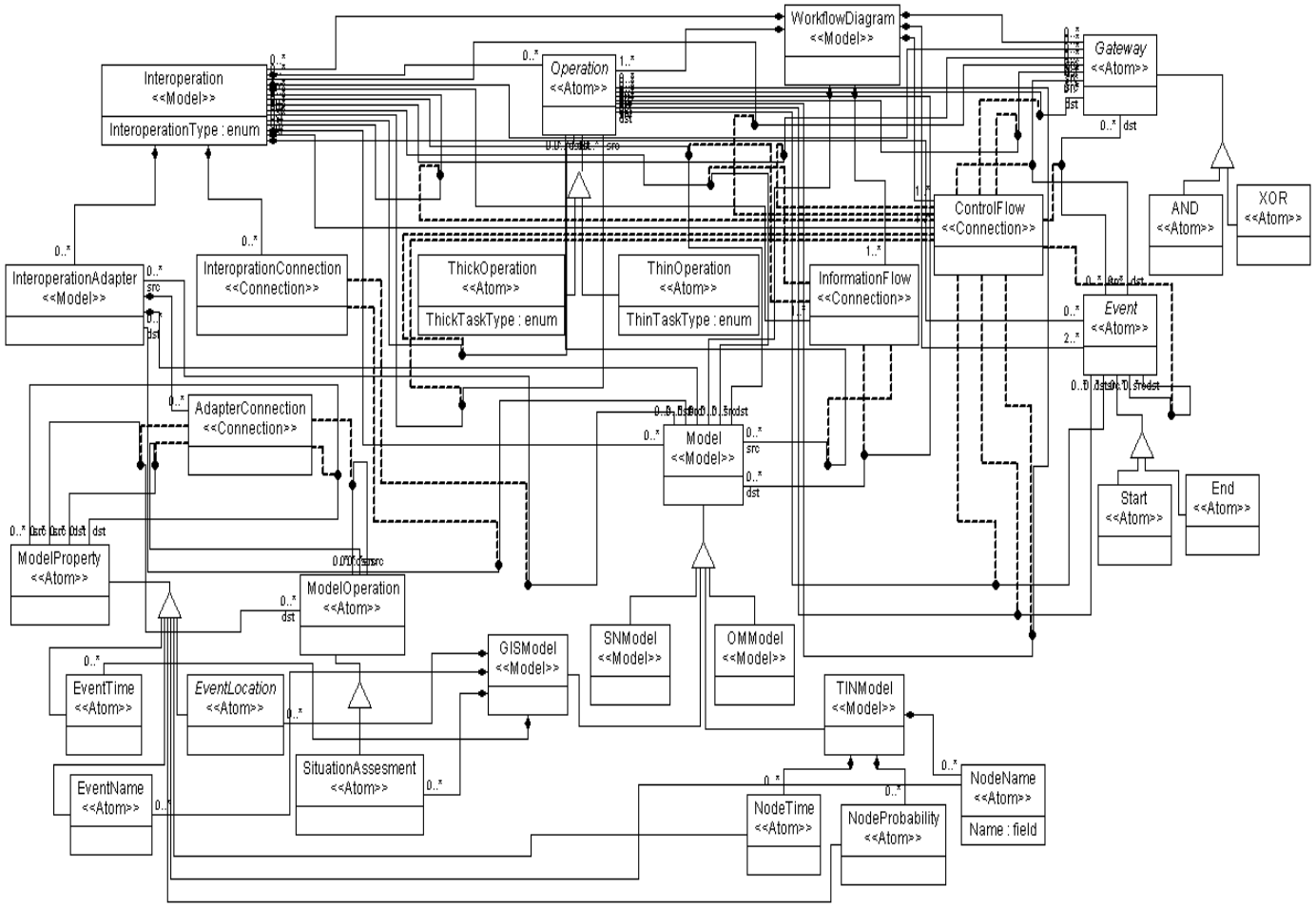

Figure 47: Meta-Model of Multi-Modeling Workflow Language (Visualization View)

7.4.4 Workflow Creation

Once the GME meta-model of the DSMWL is interpreted and registered as a new modeling paradigm in GME, we begin using the GME environment to create our scenario workflow. Figure 48 shows the top level view of the workflow. It begins with a start event that leads to a fork where parallel activities can take place. On the lower part a thick operation of constructing a TIN model *TIN 1* takes place. This operation is performed using Pythia. The construction of the TIN model can make use of existing models from previous scenarios. In parallel, a Geospatial model is constructed using WebTAS. An interoperation between the Geospatial Model *GIS 1* and the Timed Influence Net model *TIN 1* results in an updated Timed Influence Net model *TIN 2*. This first interoperation uses temporal information form *GIS 1* to update time information in *TIN 1*. A second interoperation between *GIS* and *TIN 2* results in a second updated version of the original TIN model, *TIN 3*, which has an updated probability values on some of its nodes based on situation assessment provided by the KPASA model of WebTAS. A Third parallel path on the top constructs a Social Network through a thick operation using ORA and produces *SN 1*. An interoperation follows to construct *OM 1*, an Organization Structure model out of *SN 1*. All of *SN 1*, *OM 1* and *TIN 3* become part of the latest interoperation in the workflow that produces a final revised version of the TIN model, *TIN 4*. This final version of the TIN model is used to select COA's using Pythia. This is represented as a thin operation since Pythia has the capability of exposing its SFA algorithm for selecting COA's as a service. The workflow ends at the far right side with the end event.

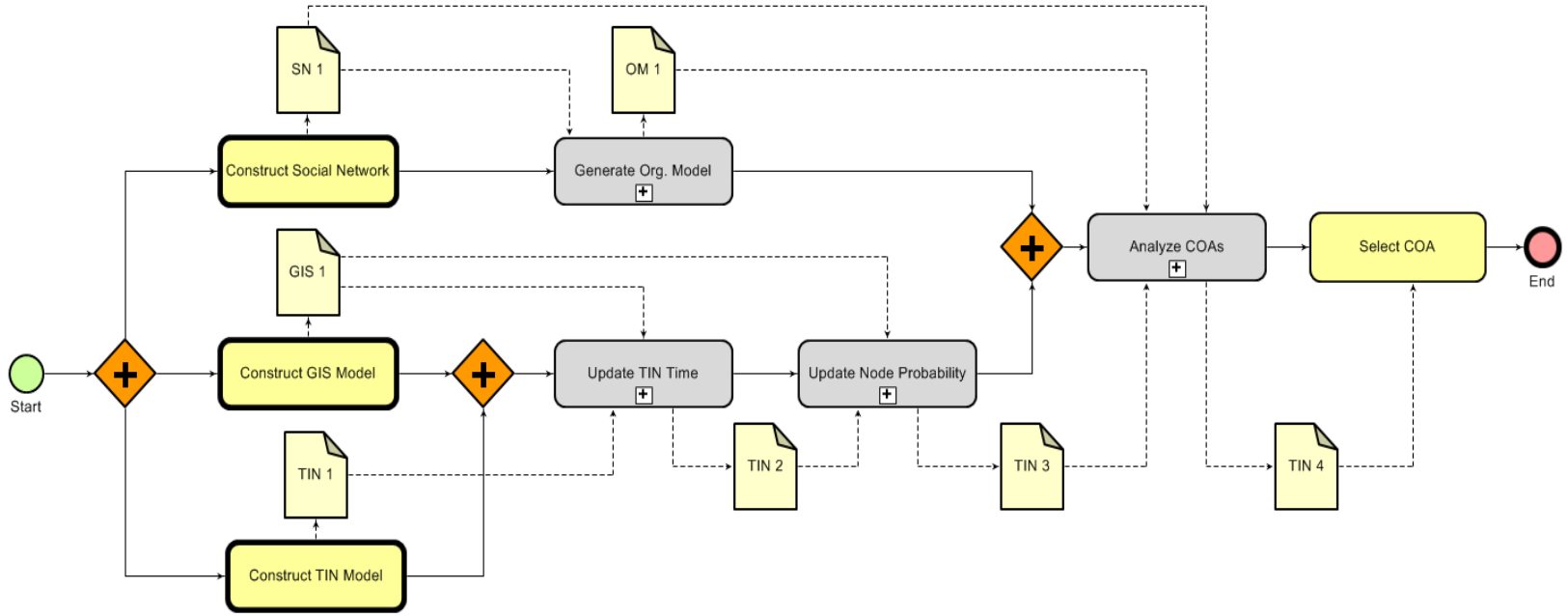

Figure 48: Scenario Workflow

The flow of the first interoperation that updates *TIN 1* time using temporal information from *GIS 1* is shown in Figure 49. It shows that a thick operation is used to get timeline information form *GIS 1* and another thick operation is used to update nodes' time in *TIN 1* to produce *TIN 2*.

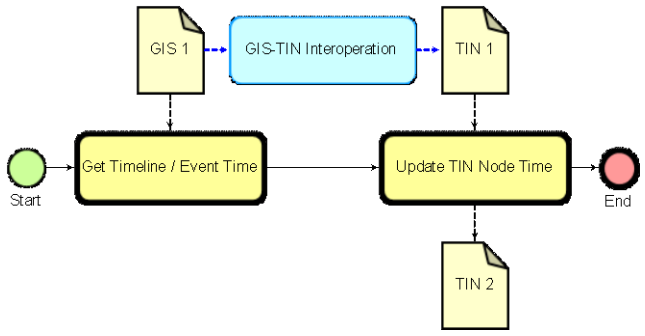

Figure 49: Interoperation Captured by the Workflow

The GIS-TIN interoperation adapter shown in Figure 50 captures the specific requirements for this kind of interoperation. It specifically maps a time element from the GIS model to node time in the TIN model.

Figure 50: Interoperation Adapter

It is at this level of the workflow, the interoperation adapted level, where semantic guidance takes place. In the case of any connection between two models, the GME Add-on developed as part of this research formulates a SPARQL query and invokes the SPQRQL query server to execute the query on the supporting domain ontology. Figure 51 shows an example of an error message resulting from a query execution. It tells the modeler that the connection created between the two models violates a semantic rule captured in the ontology.

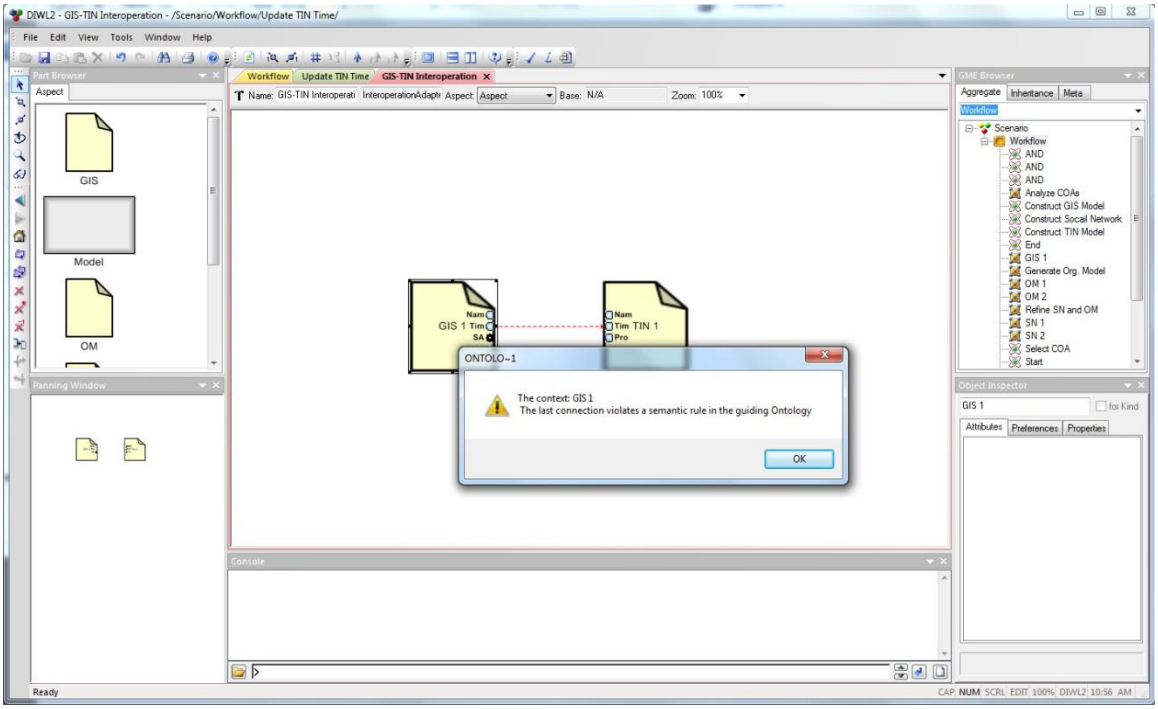

Figure 51: Semantic Guidance using GME Extension

CHAPTER EIGHT: CASE STUDY SCENARIO WORKFLOW

In this chapter we present a plausible scenario of a drug trafficking incident in which JIATF-South analysts employ multi-modeling to analyze data and to perform effective decision making. We conclude this chapter with an example of implementing the decision making workflow using SORASCS.

8.1 Scenario

In this section we present a fictitious scenario of a possible drug trafficking activity reported to, and monitored by, JIATF-South. The agency receives information about suspicious activities from different sources including local, regional and international intelligence agencies. The scenario events begin when a cargo ship with R flag (R is a country in the Caribbean) is being loaded with drugs. A drug cartel operating in country R is responsible for this drug smuggling activity. The local intelligence agency of country R intercepts a phone call between a person known to be the head of the cartel in country R and a customs officer in R's port authority. R's intelligence agency shares information with JIATF-S. JIATF-S officers react directly and begin analyzing the information. The suspected drug cartel in country R is already known to the officers. It is also known that this R based cartel is connected to another US based cartel. JIATF-S has an insider informant in the US based cartel; the informant is requested to provide more information about the cartel activities. The cargo ship leaves the port of country R on Day

n and enters international waters on day n+1. JIATF-S has UAVs that continuously monitor suspicious activities. Orders go out from JIATF-S directing some UAV's to monitor the suspected cargo ship and to keep it under surveillance. The cargo ship is expected to arrive to a US port on the Gulf of Mexico on day n+5. Figure 52 shows the main events of the scenario.

- A US based drug cartel is involved in drug trafficking activity from Country R in the Caribbean to the USA.
- A cargo ship with R flag is being loaded with drugs. A drug cartel operating in country R is responsible.
- JIATF-South receives information about the drug smuggling activity from its intelligence sources in country R.
- The cargo ship disembarks the port of country R on Day x and goes under JIATF-South surveillance in international waters starting Day x+1. The ship is scheduled to arrive to the USA on day x+5.

Figure 52: Scenario Brief

8.2 Scenario Models

In this section we present and discuss the models being used in the scenario analysis. These are the models that will participate in the multi-modeling workflow. JIATF-S analysts create these models based on the information they receive. In many cases they use existing models from similar situations and update them with most recent data. In this case study three types of models will be used, a Timed Influence Net model, a Social Network Model, and a Geospatial Model. To enhance the analysis capability, these models interoperate and exchange data. In some cases a model can be used to generate another. In the analysis performed by JIATF-South analysts a geospatial model

created using WebTAS provides a timed influence net model with temporal information. A WebTAS KPASA situation assessment feature provides probability measures to some of the timed influence net model nodes.

As mentioned earlier, these models are part of the scenario workflow. We discuss them in detail before we discuss the workflow to clarify the role of each model in the scenario workflow. Final results resembling COAs produced from the revised TIN model are shown as part of the workflow implementation step.

8.2.1 Timed Influence Net Model Using Pythia

A TIN model is used to analyze the effects of actions taken to address the drug trafficking threat in this scenario. Pythia [61] is used to create and analyze the TIN model. Figure 53 shows the model created and analyzed as part of the scenario workflow. Based on the information available, JIATF-S analysts constructed this model. The target goal as shown on the node on the right side of the model is to interdict drug trafficking activity. The actions taken by JIATF-S and other scenario actors are shown on the left side of the model. Pythia has the capability to analyze the effect of setting each of these actions on or off. TIN analysis results include a set of COAs with a probability profile for each of these COAs. The COA that maximizes the probability of the goal node, interdicting drug trafficking in this case, is considered to be the best COA. This was obtained using the SAF optimization algorithm embedded in Pythia [61]. The selected COA identifies which actionable events influence the success of drug trafficking efforts.

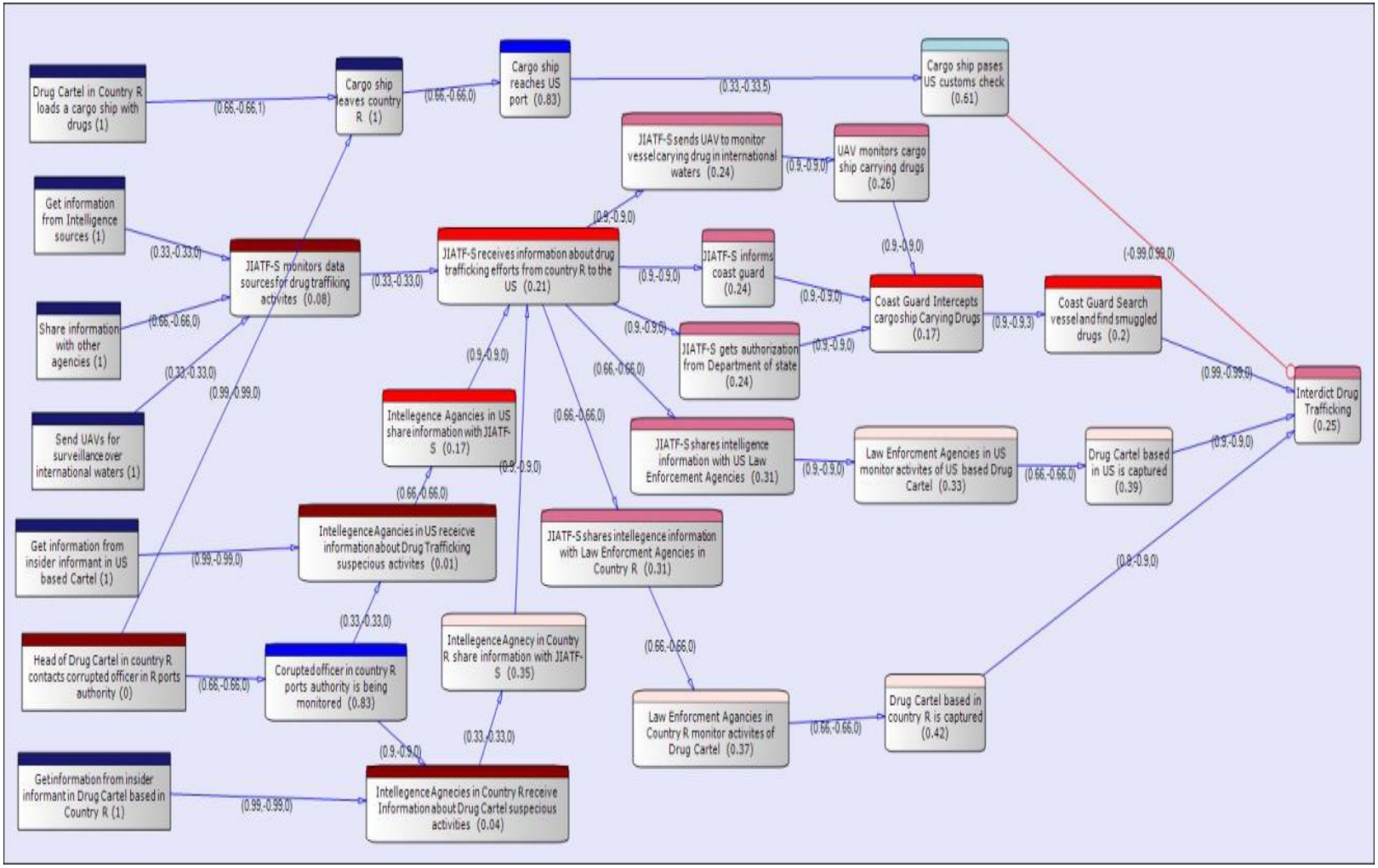

Figure 53: Scenario TIN Model

8.2.2 Social Network Model Using ORA

A SN model captures relationships between agents. JIATF-S receives continuous information about drug cartels and their members. With the help of ORA, relationships between cartel's members and among different cartels are captured. These networks can help to identify the most effective individuals, "leaders," and the paths of communication between different cartels. Figure 54 shows a simple SN for the cartels involved in the scenario. The nodes on the left side of the model represent the cartel based on country R while the nodes on the right represent the US based cartel. The arrows represent the relationships between the individuals and help to identify the most connected and influence individuals.

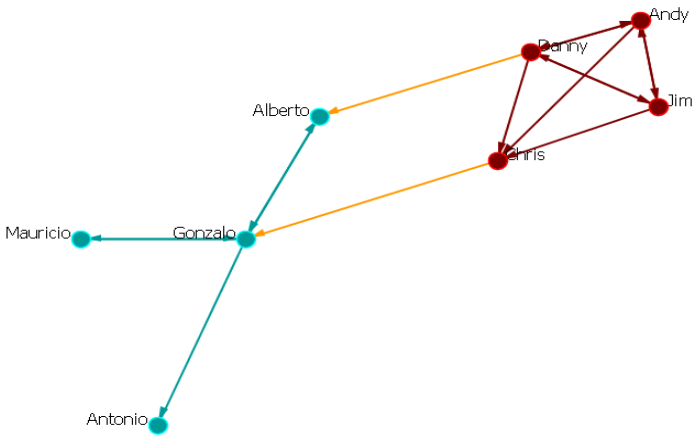

Figure 54: Scenario Social Network Model

These kinds of relationships and information captured in Social Networks help JIATF-S analysts to identify best actions to track and interdict drug trafficking activities taken by these two cartels. In the scenario workflow, a SN is used to generate an

Organization Structure for further analysis and then to revise the TIN model before using it to generate COAs.

8.2.3 Geospatial Model Using WebTAS

JIATF-S uses WebTAS to track temporal and geospatial data. WebTAS provides capabilities to create dynamic maps and timelines that help analysts to visualize data. Timing of events is very important to track in order to make effective decisions. Figure 55 represents a WebTAS interactive map that tracks locations and times of events. Events occurrence time captured in WebTAS is used to update the TIN model with events time. This is a very important step that allows Pythia to provide COAs based on accurate temporal information.

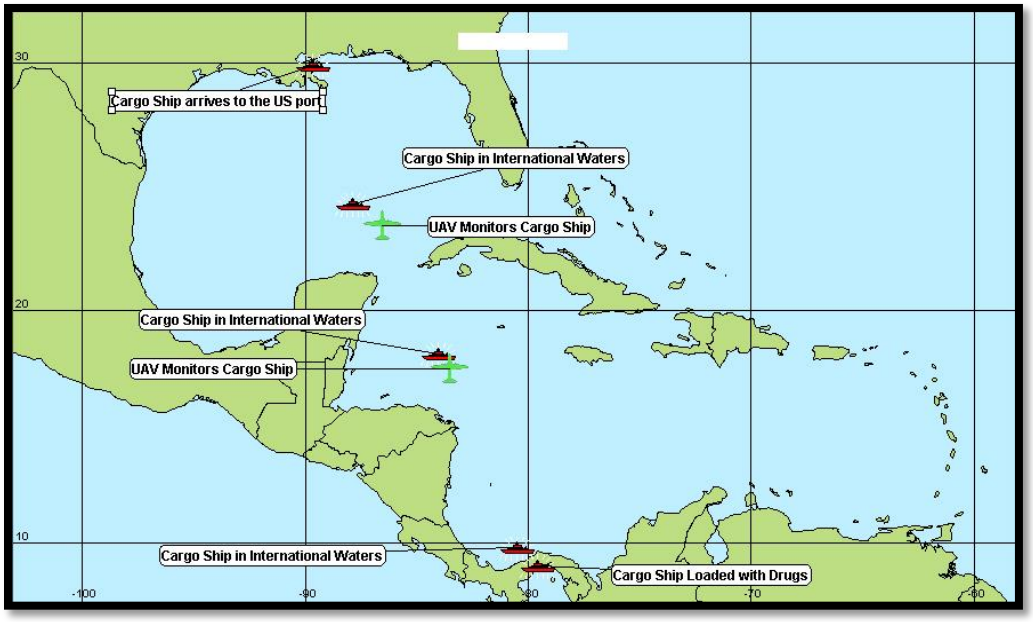

Figure 55: Scenario WebTAS Map

The KPASA situation assessment model assesses the probability of the suspected cargo ship to reach the US on day n+5. This probability is used to update an event node in the TIN model.

8.3 Workflow Implementation

The selected platform to implement our workflow determines how the workflow will be interpreted. In this case study we use SORASCS to implement our workflow. In Chapter Six we discussed in detail the tasks required to implement a workflow on a target platform. As part of the discussion we presented how a workflow can be interpreted to be executed on SORASCS.

After the workflow is interpreted, it gets executed using SORASCS. The participating modeling techniques are utilized through their supporting tools that are already integrated into the SORASCS platform. The focal point of the models is the TIN model. The workflow sequence operates on revising and optimizing this TIN model for better COA analysis and selection. Interoperations between models participating into the multi-modeling activity enhance the analysis process. The relationships captured in the Social Network model identify the most influential actors in the drug trafficking effort and are used to update the TIN model actions and probabilities. Temporal information captured in the WebTAS model is used to update the timing of actions in the TIN model. After the TIN model is refined based on the data received from other models, Pythia capabilities and algorithms are utilized for COA selection.

The last operation of the workflow, as was shown in Figure 48, represents the COA selection task. Different combinations of action events are examined to determine the COA that gives highest probability for a specific node goal, which is successful drug interdiction in this scenario. We show and discuss here three generated COAs.

The first COA visualized in Figure 56 shows that sharing information between the JIATF-South and other (local and regional) intelligence agencies in addition to the utilization of surveillance resources, the probability of the target node of effective drug interdiction reaches its highest level around 68%. This is the selected COA by JIATF-S since it maximizes the probability of Drug Interdiction for the scenario under consideration. In the second COA, the probability of interdicting smuggled drugs decreases dramatically to about 32% when information sharing between the JIATF-South and other local and regional intelligence agencies is not in effect. *This shows the value of information sharing to the success of drug interdiction efforts*. The third COA shows how the probability of effective drug interdiction can decrease even more to a level close to 25% if in addition to lack of information sharing, insider information from drug cartel is not available. Appendix C has the complete list of figures and tables for COA analysis.

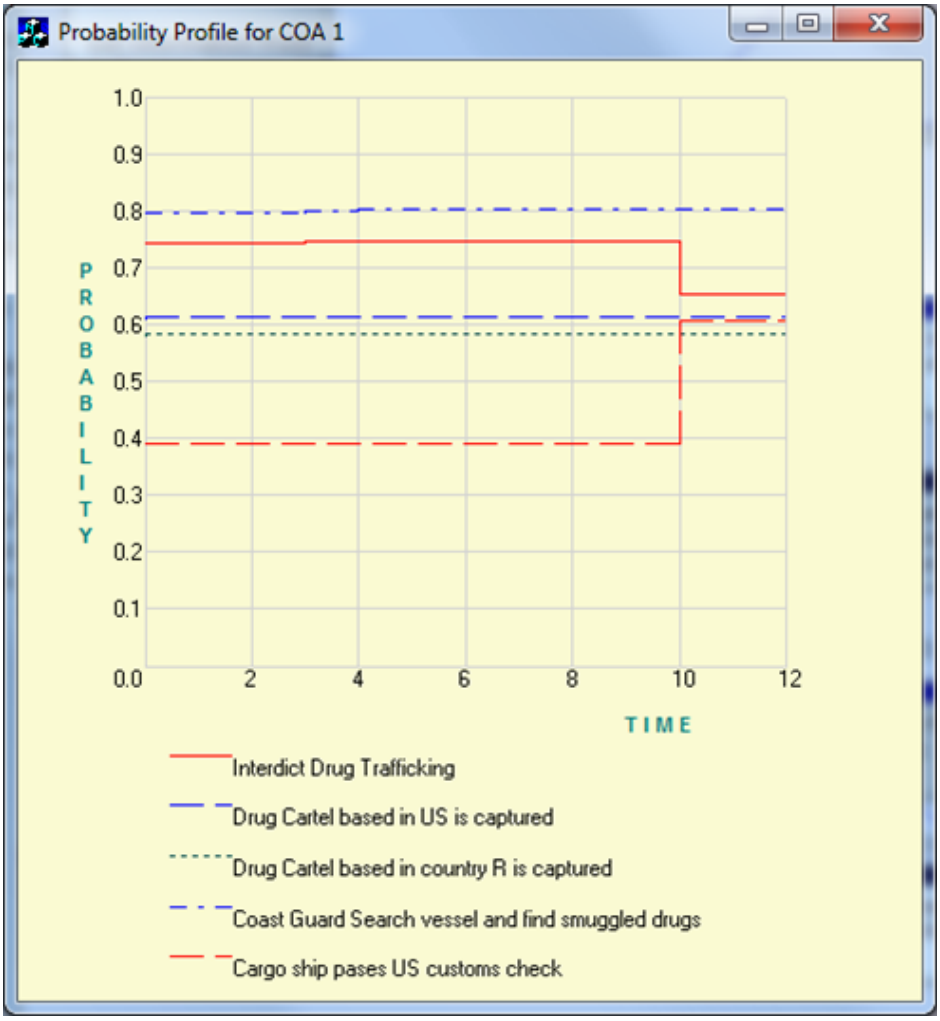

Figure 56: First COA

Table 10 shows the status of the actionable events that resulted in this probability. These actionable events are used to make decisions on which resources to employ and utilize for effective Drug Interdiction with respect to the scenario of interest. The comparison between the three COAs tells clearly that in order for an agency like the JIATF-South to be effective and successful in its drug interdiction efforts, all available resources should be utilized. It is also evident that information sharing and exchanging

real time data about any specific drug smuggling scenario increases the probability of success significantly.

Table 10: Selected COA

Action	**Status**
Drug Cartel in Country R loads a cargo ship with drugs	1
JIATF-S gets information from Intelligence sources	1
JIATF-S shares information with other agencies	1
JIATF-S sends UAVs for surveillance over international waters	1
Head of Drug Cartel in country R contacts corrupted officer in R ports authority	1
Get information from insider informant in US based Cartel	1
Get information from insider informant in Drug Cartel based in Country R	1

CHAPTER NINE: CONCLUSION, LIMITATIONS AND FUTURE WORK

9.1 Conclusion

The use of multi-modeling platforms to perform simulation and analysis is relatively new. The complexity of problems and the huge amounts of data available for analysis encourage the use of new approaches. Multi-modeling platforms provide many new capabilities but at the same time they come with other challenges. The human interaction with the multi-modeling process can be overwhelming if non-technical users are to be involved with the low level integration of models. In addition to that, since multi-modeling allows for multiple models to interoperate and exchange data, there exists a challenge in guaranteeing that only valid exchanges are allowed. This spans both syntactic and semantic aspects of interoperations.

In this dissertation we tried to address the aforementioned two issues. The first issue is to provide users of multi-modeling platforms with a high level visual capability to capture multi-modeling activities in the form of workflows. The second is to guarantee that interoperations between different models are semantically correct. The approach followed in this research stems from the fact that the use of modeling techniques in solving analysis problems is domain dependent. This led to introducing the concept of a Domain Specific Multi-Modeling Workflow Language (DSMWL). A systematic methodology for developing and using such language was the focal point of the research

including the use of a supporting domain ontology for semantic guidance. The methodology requires the characterization of the domain of interest as a preliminary step that precedes the development of the DSMWL or its supporting ontology. Domain characterization is a complex problem by itself. It has been addressed in our methodology by building on top of previous research in this area.

This five-step methodology for developing and using a DSMWL is the main contribution of the dissertation. This is accompanied by the incorporation of a Domain Ontology in the process of creating multi-modeling workflows for the sake of semantic guidance on the validity of multi-model Interoperation. In addition, GME extensions for integrating domain ontology and interpreting workflows are considered to be technical-level contributions.

The case study presented in this dissertation provided a detailed example on the application of the methodology. In this application, the methodology steps were followed to develop a DSMWL for the decision making process in drug interdiction domain. The DSMWL was then used successfully to create workflows that capture multi-modeling activities required to perform multi-model based analysis for a specific scenario. A workflow created using the DSMWL was interpreted and implemented using the target platform, SORASCS. Analysis results based on the multi-modeling activities captured in the workflow were obtained after using the platform.

The main conclusion out of this research effort is that it was possible to provide multi-modeling platforms users with a high level workflow layer. This layer provided the

capability of capturing multi-model interoperations. Another important conclusion was that the use of a supporting ontology in the multi-modeling process to guarantee the creation of semantically valid interoperations with respect to a domain of interest is achievable.

9.2 Limitations and Future Work

In this dissertation the focus was on the overall process of developing and using a domain specific multi-modeling workflow language. Each step of the proposed methodology represents a research area by itself. While we tried to address some of these areas in some level of detail, there still exist limitations and the need for future work to be done.

The first step of the methodology in which domain identification takes place requires more attention. The use of informal description of a domain and then creating concept maps out of that description was the approach used in the research. It allowed for some kind of semi-formal representation of the domain but is limited as it depends completely on the understanding and knowledge of the domain expert performing this task. A formal method with complete set of rules is still required.

In the domain analysis step, the second step of the methodology, two formal representations of the domain are to be obtained. One is the representation of the domain and modeling techniques structural and syntactic concepts in the form of UML class diagrams. The processes of using concepts from concept maps to create class diagrams and then consolidating class diagrams in one domain class diagram needs more

elaboration. Also, the process of creating upper domain ontology by employing different techniques is another area of the research that needs more work.

When it comes to the corner-stone step of the methodology in which the development of a domain specific multi-modeling workflow language takes place, a substantial amount of work is required each time a new domain of interest is being addressed. This can be seen as a limitation to the approach; however, a DSMWL can be always reused for the same domain. It has also been discussed of the trade-off between using an existing generic purpose language and developing a new language. While the case has been made for going in the direction of a domain specific multi-modeling workflow language, there is still a challenge in reaching a level of expressiveness sufficient to address a domain of interest using a new language. The tasks required to develop a new language presented in this research deserve additional future work.

The use of a DSMWL to create multi-modeling workflows is the ultimate goal and is addressed by the forth step of the methodology. In addition, the use of a domain ontology to guide the creation of semantically correct workflows has been presented. To allow the integration of the domain ontology guidance into the workflow creation process a GME extension was developed. The extension reacts to certain events while creating workflows and generates SPRAQL queries that get executed by a SPRAQL engine on the domain ontology. The formulated SPARQL query has not been addressed in detail in this dissertation. For the sake of the case study application, simple quires are formulated based on the entities involved in the workflow creation event that triggers the GME

extension. More future work is required to advance this technique and to provide more integration of the ontology into the overall workflow creation process.

Finally, the implementation of a workflow using a multi-modeling platform was achieved by interpreting the workflow into a format that can be executed by the platform. For this to happen a mapping between the developed domain specific multi-modeling workflow language and the execution language of the multi-modeling platform takes place. Although this mapping provides means to allow the implementation of workflows, it is limited to the level the two languages can be mapped. In future work, a research on the possibility of including the multi-modeling platform in the early stages of the methodology is required.

An example of a SOA based multi-modeling platform was discussed. Such a platform can be enhanced by using advanced automation techniques like Intelligent and Semantic Agents. This is another direction of research that can be taken. Intelligent semantic agents can play the role of experts in exchanging knowledge. A multi-modeling workflow can be implemented on a platform that allows intelligent agents to interact, reason about knowledge, and exchange data required for model interoperation.

APPENDIX A: GME ADD-ON FOR SEMANTIC GUIDANCE

```
void Component::objectEventPerformed( Object& object, unsigned long event, VARIANT v )
{
	// ======================
	// Insert application specific code here //object->getName().c_str()

if(object->getName()=="AdapterConnection")
{
 using namespace System;
 using namespace System::Net;
 using namespace System::IO;

	CString URL= "http://localhost:3030/drug/query?query=";

	CString query="PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX
this: <http://www.owl-ontologies.com/Ontology1355252085.owl#> SELECT ?x WHERE
{this:@elementA rdf:type ?x. this: @elementB rdf:type ?x}";

	URLEncode(query);

	String ^sURL= gcnew String (URL+query);
	WebRequest ^wrGETURL;
	wrGETURL = WebRequest::Create(sURL);

	WebProxy ^myProxy = gcnew WebProxy("myproxy", 3030);
	myProxy->BypassProxyOnLocal = true;

	wrGETURL->Proxy = WebProxy::GetDefaultProxy();

	Stream ^objStream = wrGETURL->GetResponse()->GetResponseStream();

	StreamReader ^objReader = gcnew StreamReader(objStream);
	String ^sLine = "";
	Int32 i = 0;
	CString str=objReader->ReadToEnd();
	CString str2="<result>";
	if (str.Find(str2) >0) {
		AfxMessageBox("Result Found");
	}
	else
	{
		AfxMessageBox("No Result");
	}
}
	AfxMessageBox( "The context: " + CString( object->getName().c_str() ) +
		"\r\n This conncetion violates a semantic rule in the guiding Ontology"
)
```

Figure A1: Workflow Event Handler

```
void URLEncode(CString & in_url)
 {
 int x;
 CString out_url;
 char hexstr[2];

for (x = 0; x < in_url.GetLength(); x++)
 {
if (in_url[x] == ' ')
 {
 out_url += '+';
 }

 else if ( (in_url[x] < '0') || ( (in_url[x] > '9') && (in_url[x] < 'A') ) || (
(in_url[x] > 'Z') && (in_url[x] < 'a') ) || (in_url[x] > 'z') )
 {

 if ( (in_url[x] !='-') || (in_url[x]!='_') || (in_url[x]!='.') )
 {
 out_url += '%';

_itoa( in_url[x], hexstr, 16);
 out_url += hexstr;
 }
 }

 else
 {
 out_url += in_url[x];
 }
 }
 in_url = out_url;

 }
```

Figure A2: SPARQL Query URL Encoder

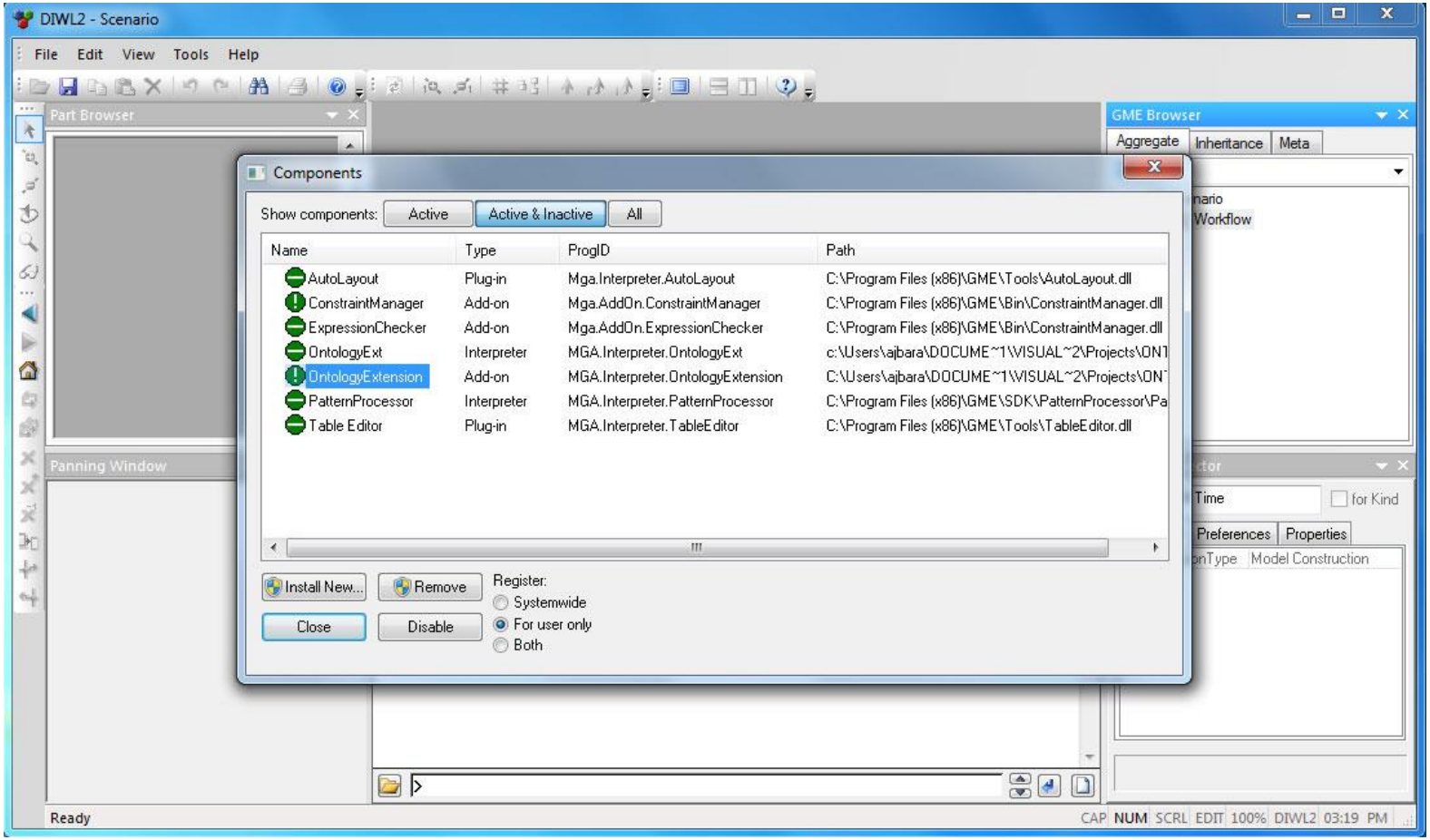

Figure A3: GME Ontology Extension Registration

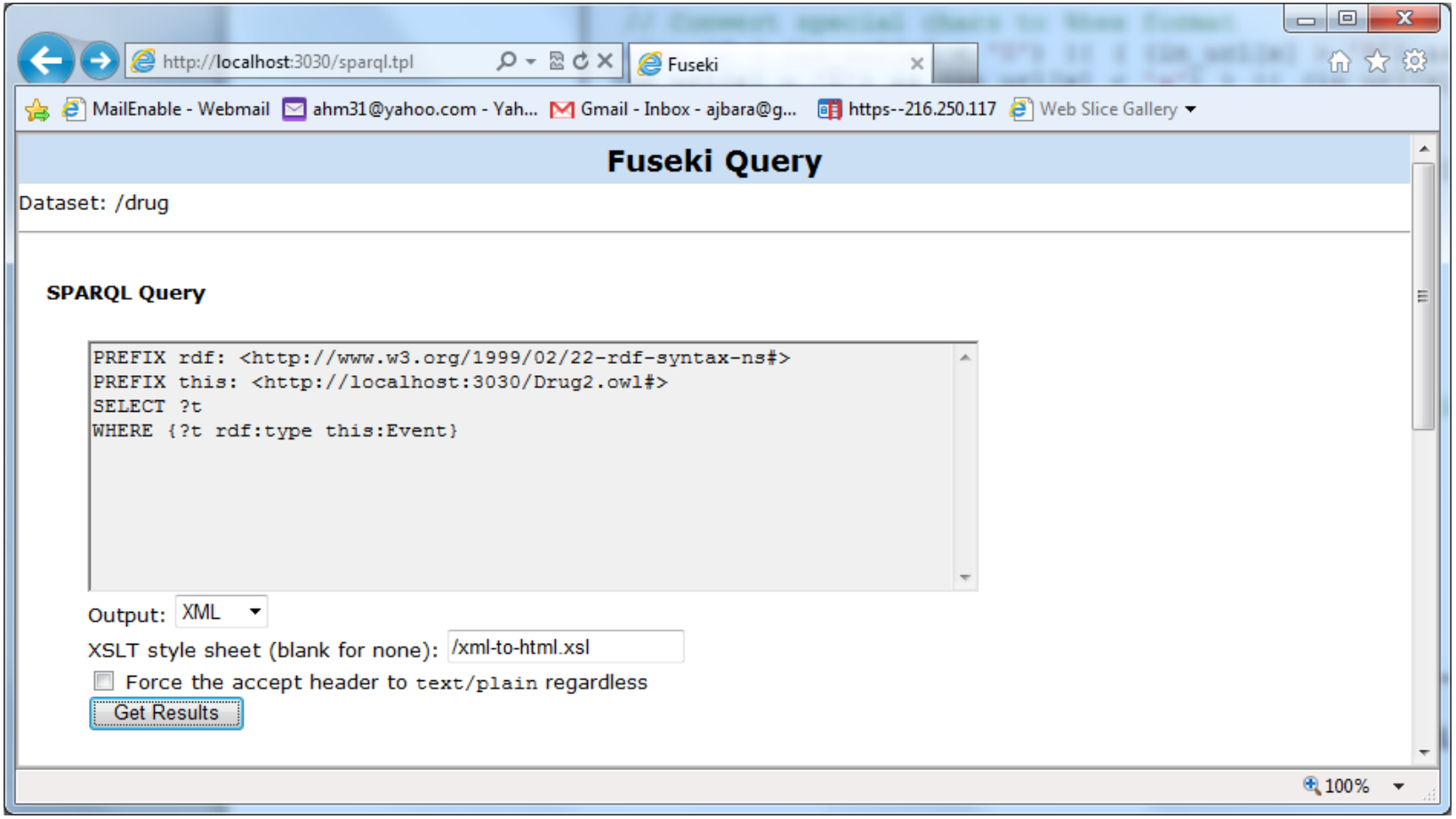

Figure A4: Fuseki Interface

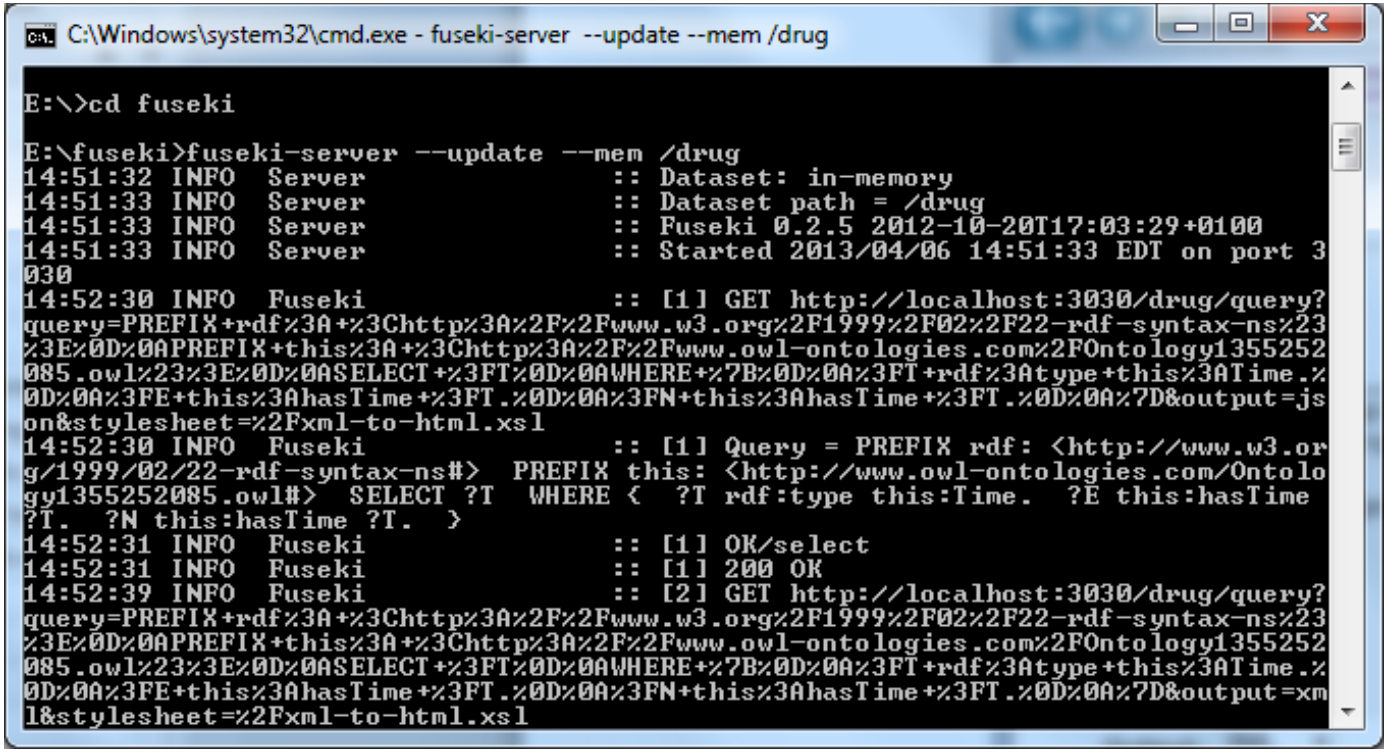

Figure A5: Fuseki SPRAQL Query Server

APPENDIX B: GME WORKFLOW INTERPRETER

```
public void Main(MgaProject project, MgaFCO currentobj, MgaFCOs selectedobjs,
ComponentStartMode startMode)
        {

            GMEConsole.Out.WriteLine("Interpreting Workflow....");


                        IMgaFolder rootFolder = project.RootFolder;


            foreach (MgaObject rootObject in rootFolder.ChildObjects)
            {
                if (rootObject.Name == "Workflow")
                {
                    interpretWorkflow(rootObject);
                }
            }

        }

        public void interpretWorkflow(MgaObject workflow)
        {
            XmlDocument bpelElements= new XmlDocument();

            foreach (MgaObject workflowObject in workflow.ChildObjects)
            {
                XmlElement bpelElement = maptoBPEL(workflowObject);
                bpelElements.AppendChild(bpelElement);
            }

            exportBPEL(bpmnElements);
        }
```

Figure B1: Function that interprets workflow into target platform language

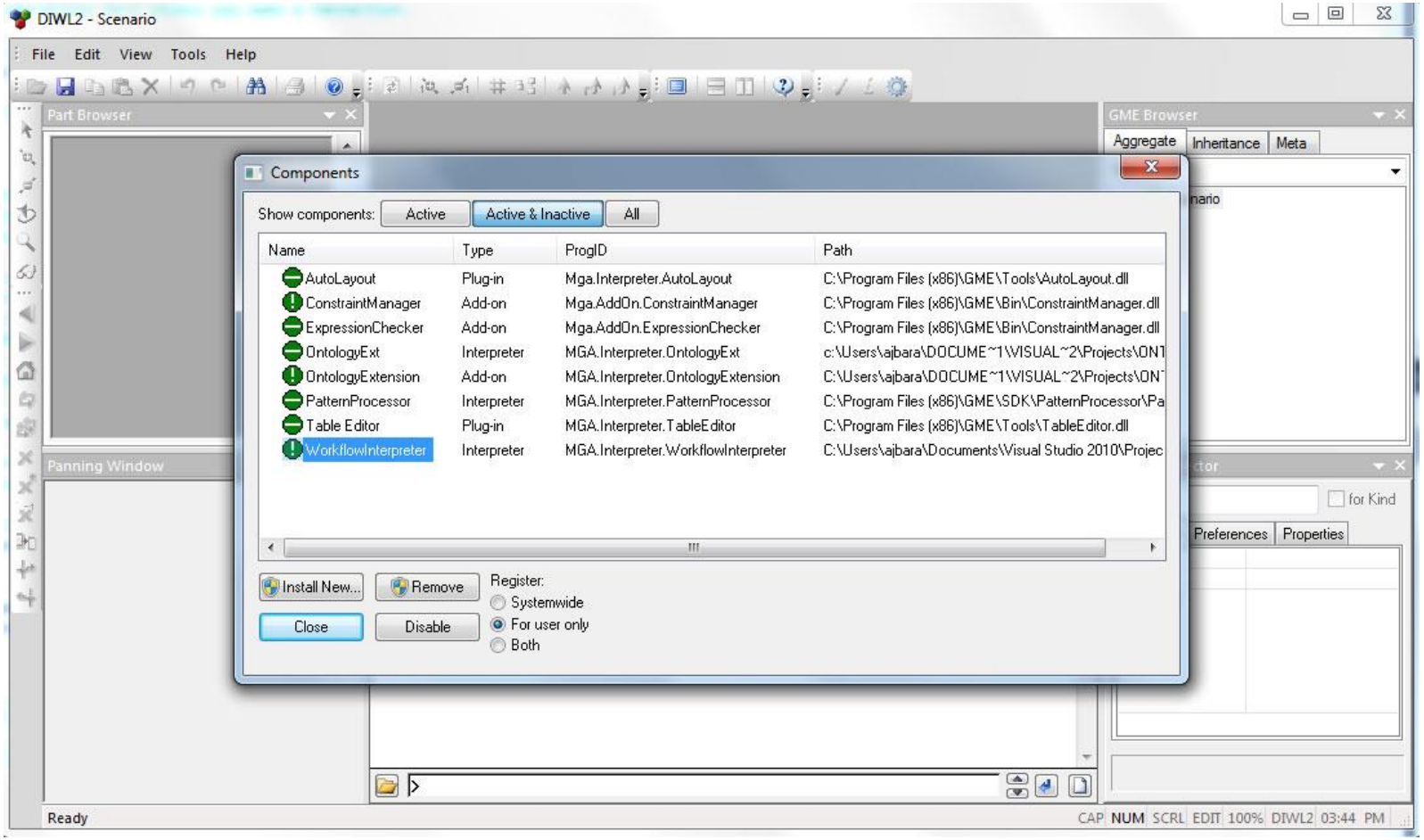

Figure B2: GME Interpreter Registration

APPENDIX C: CASE STUDY APPLICATION DOMAIN

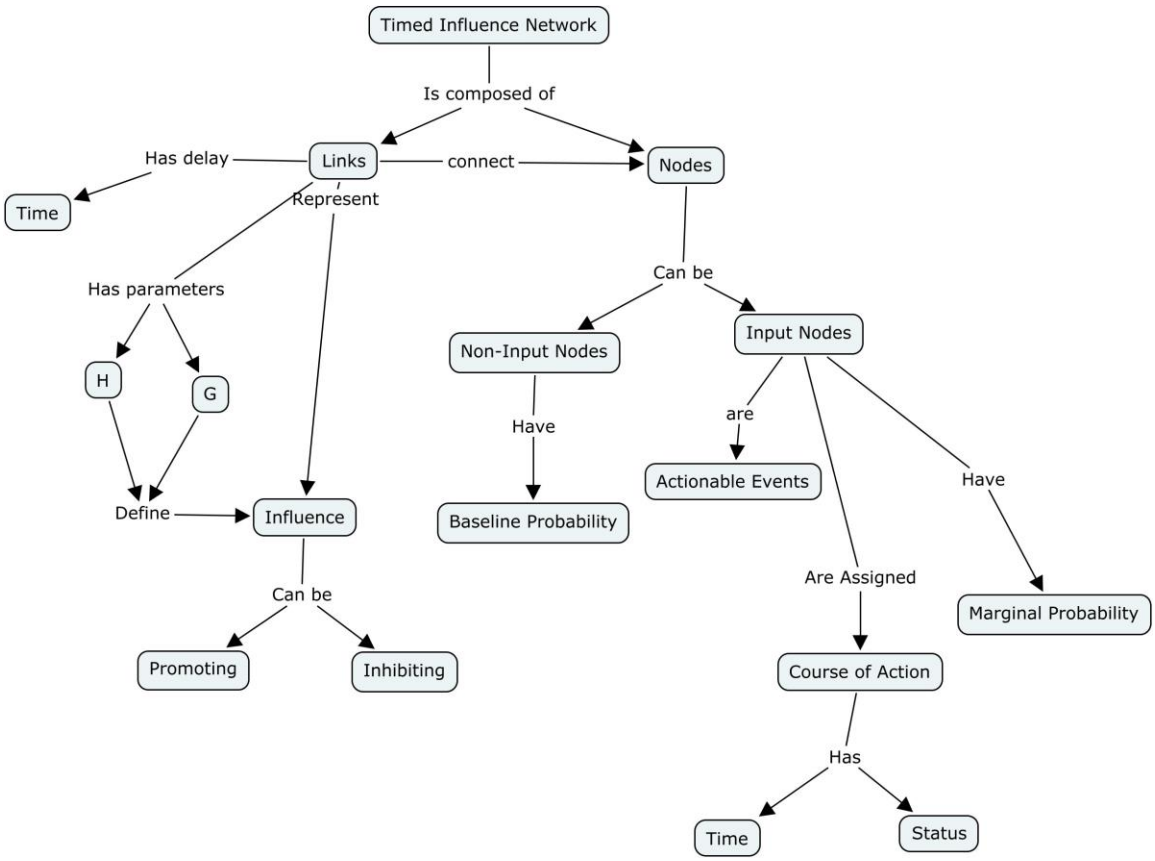

Figure C1: TIN Basic Constructs Concept Map

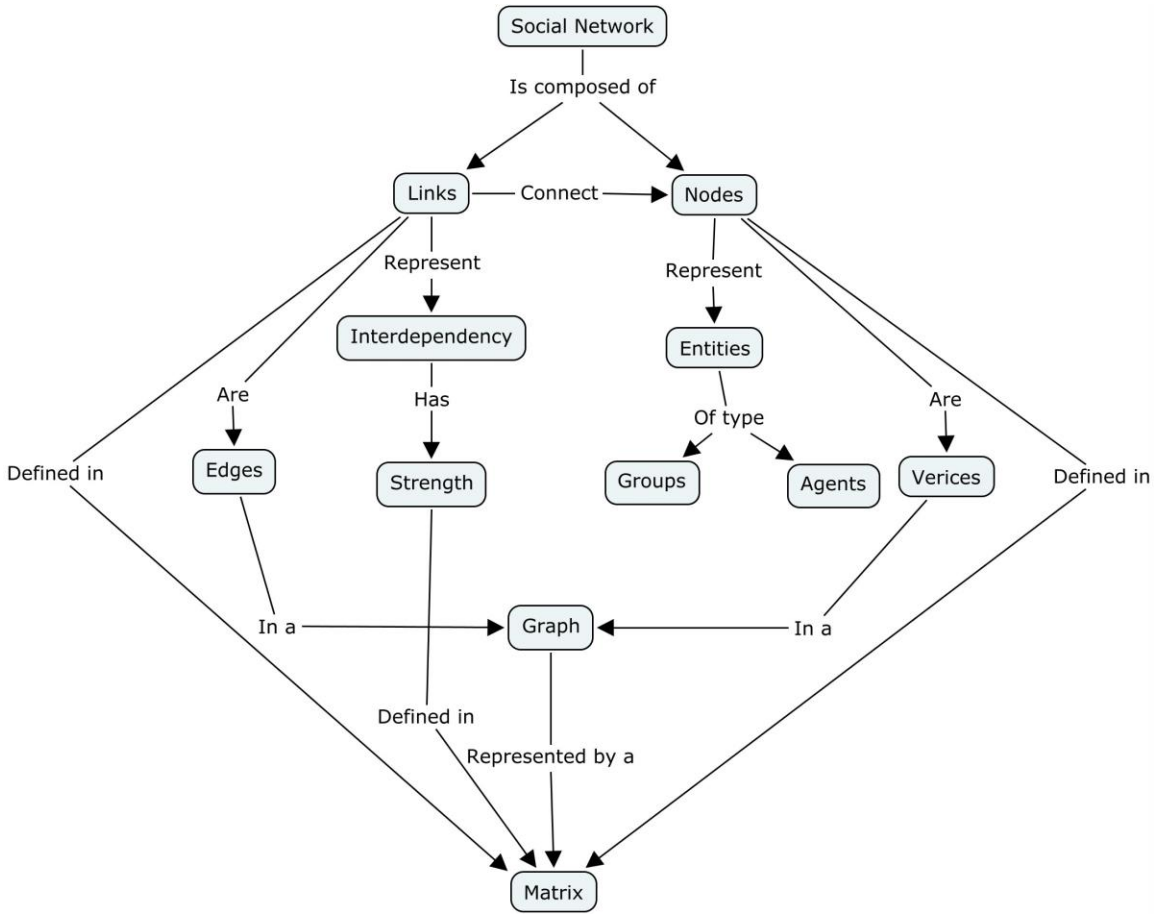

Figure C2: Social Network Basic Constructs Concept Map

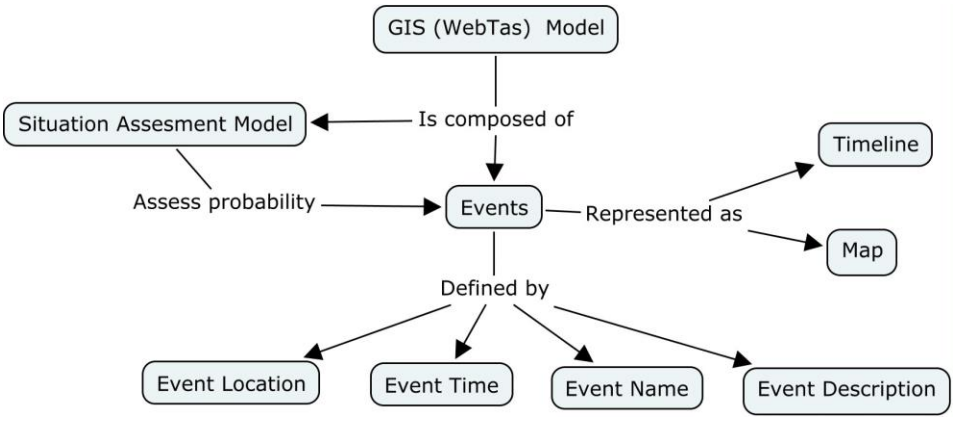

Figure C3: GIS (WebTas) Model Basic Constructs Concept Map

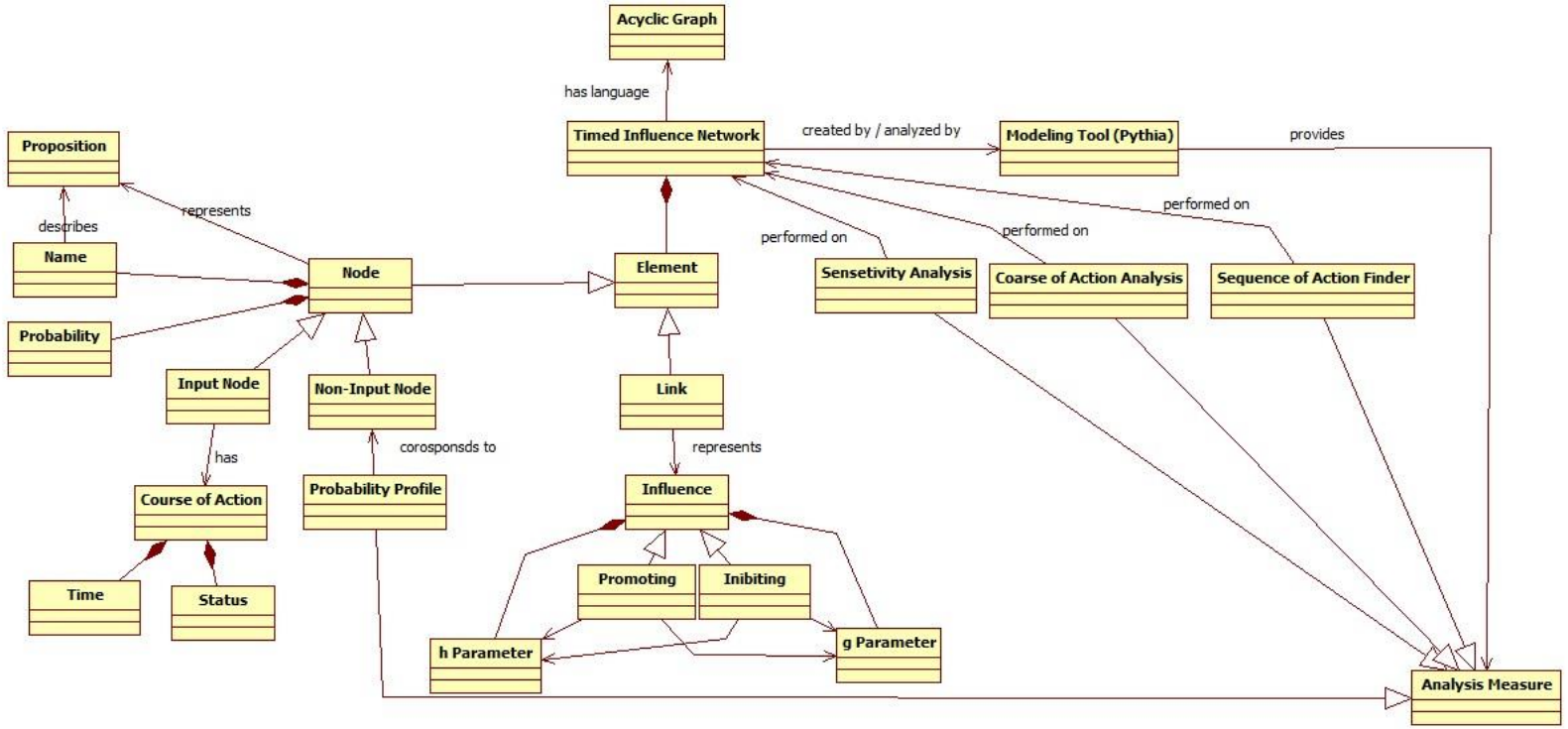

Figure C4: TIN Class Diagram

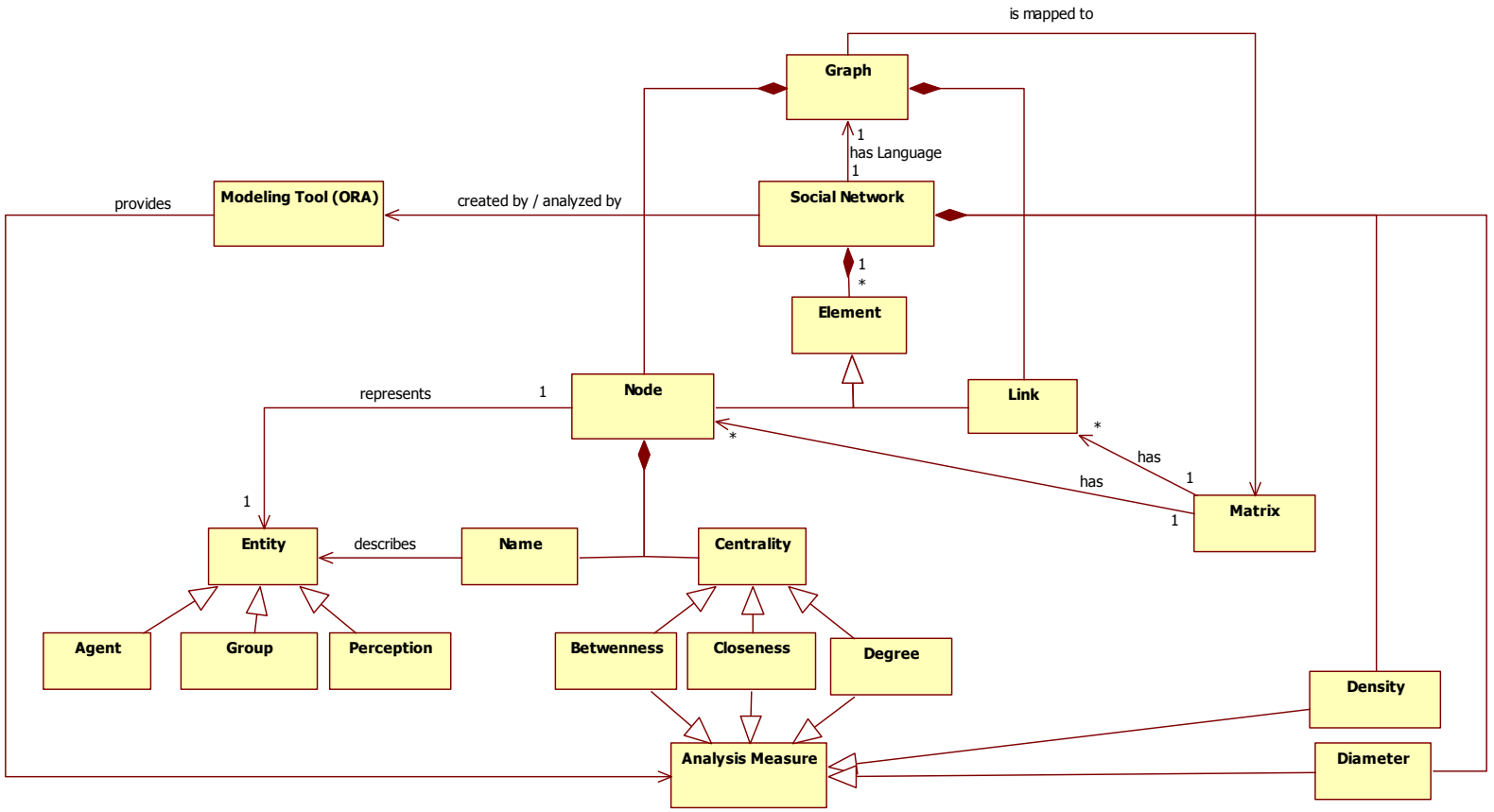

Figure C5: Social Network Constructs Class Diagram

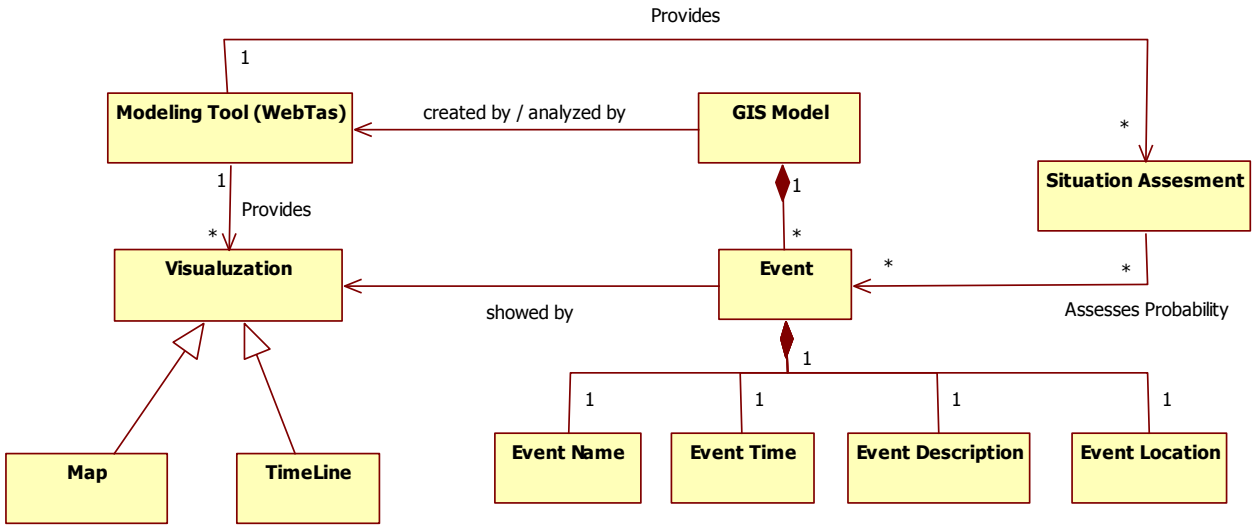

Figure C6: GIS Constructs Class Diagram

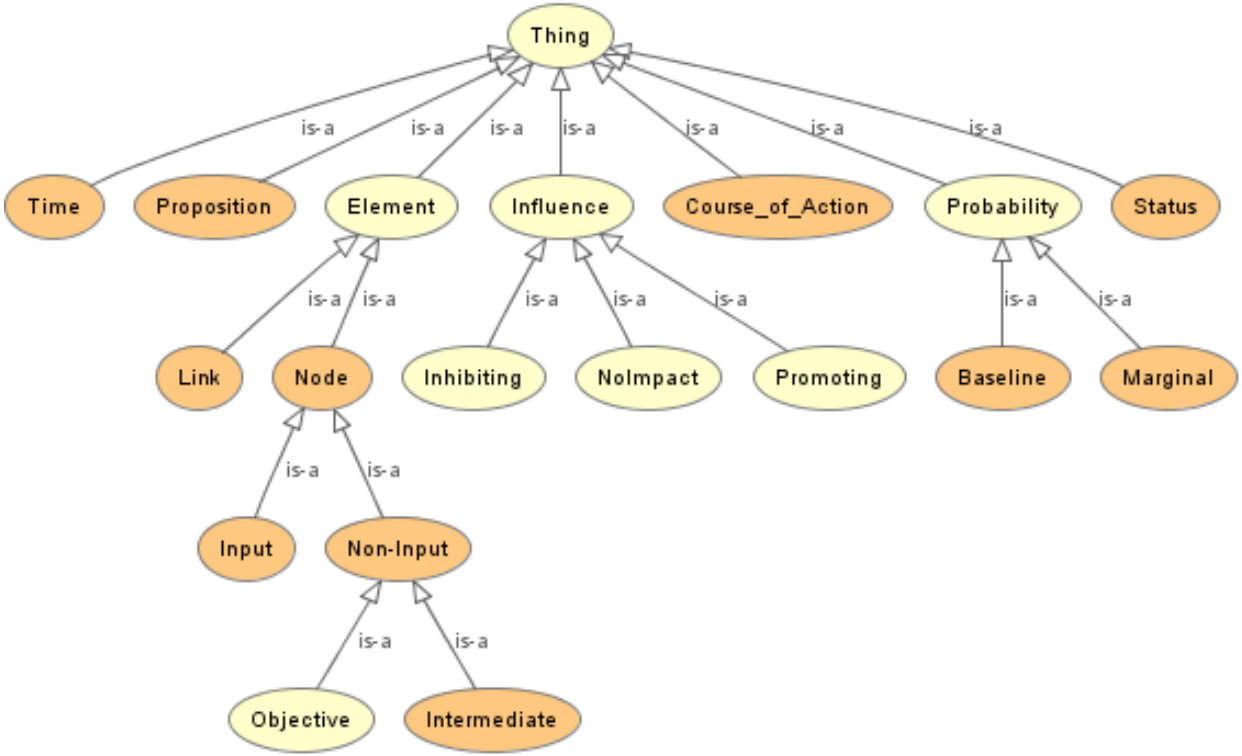

Figure C7: TIN Pseudo Ontology (OWLViz View)

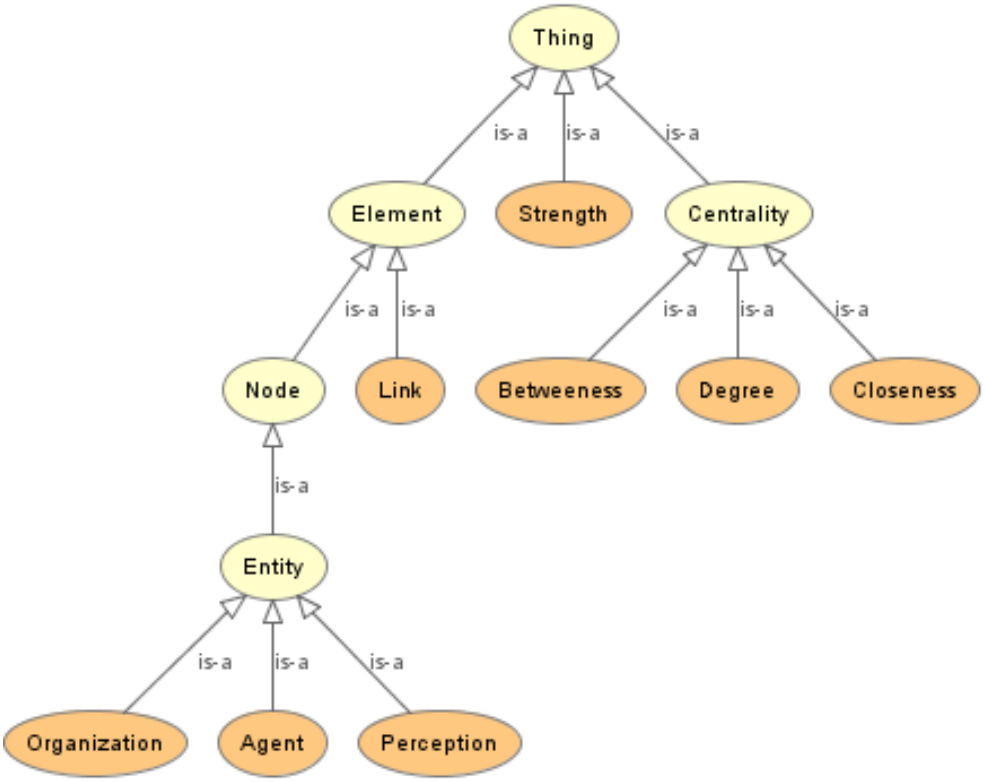

Figure C8: Social Network Pseudo Ontology (OWLViz View)

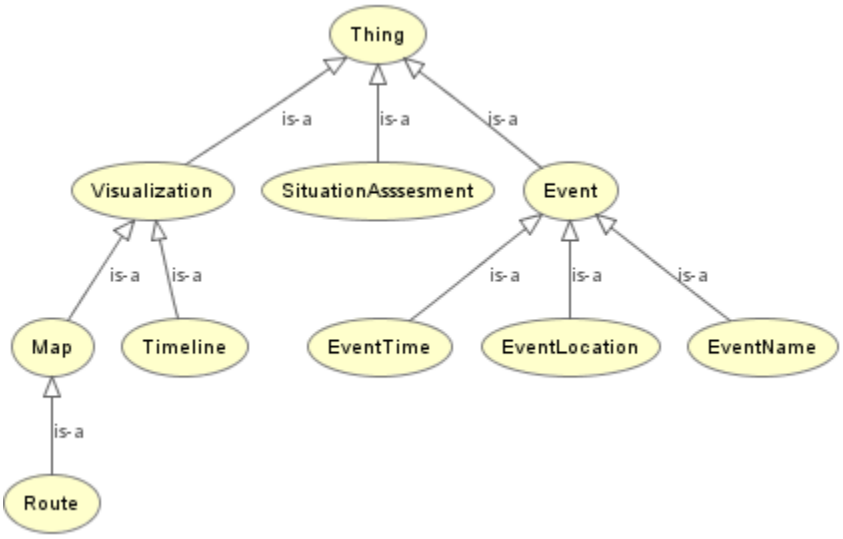

Figure C9: GIS Pseudo Ontology (OWLViz View)

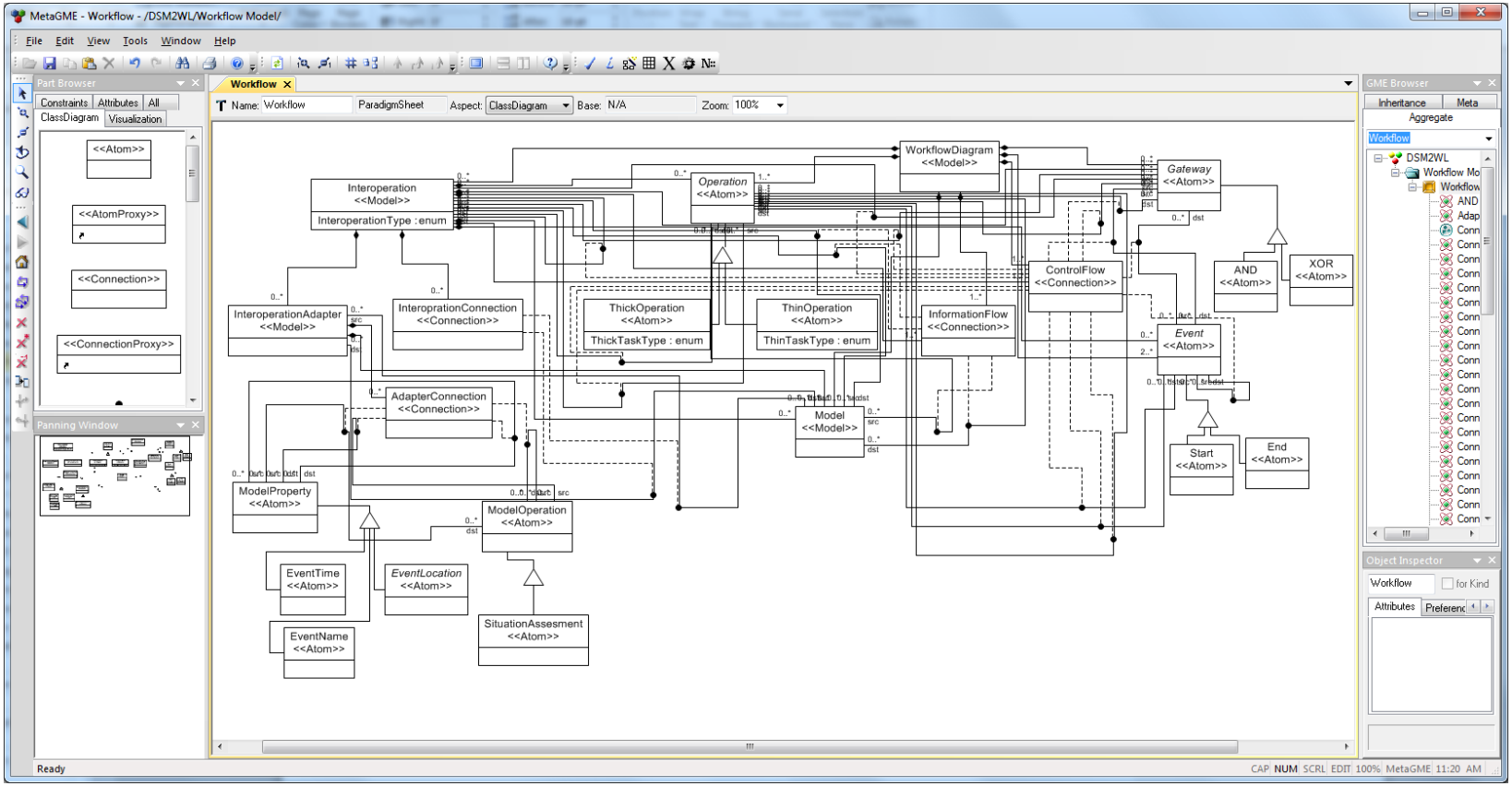

Figure C10: GME Classes Aspect

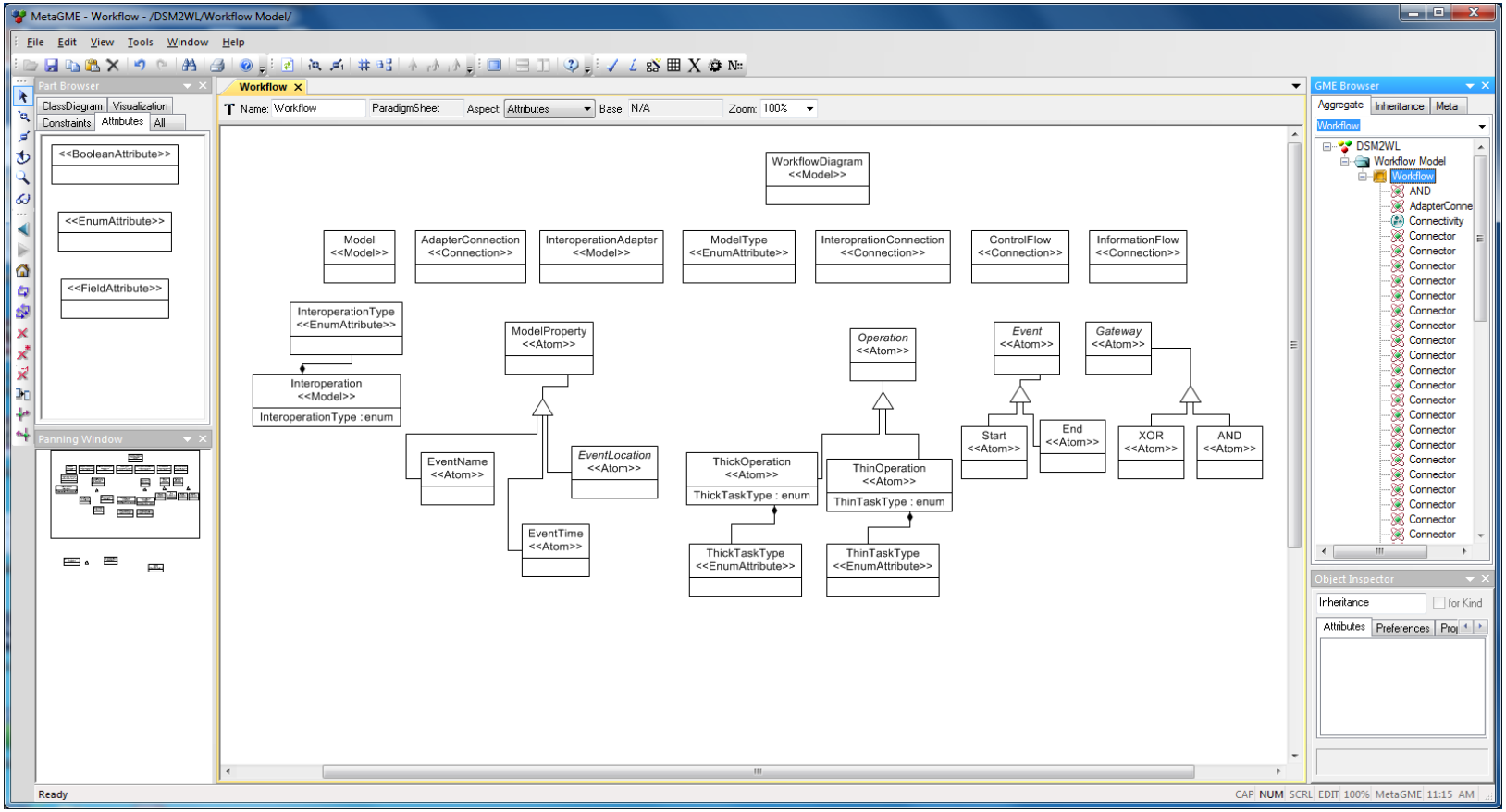

Figure C11: GME Attributes Aspect

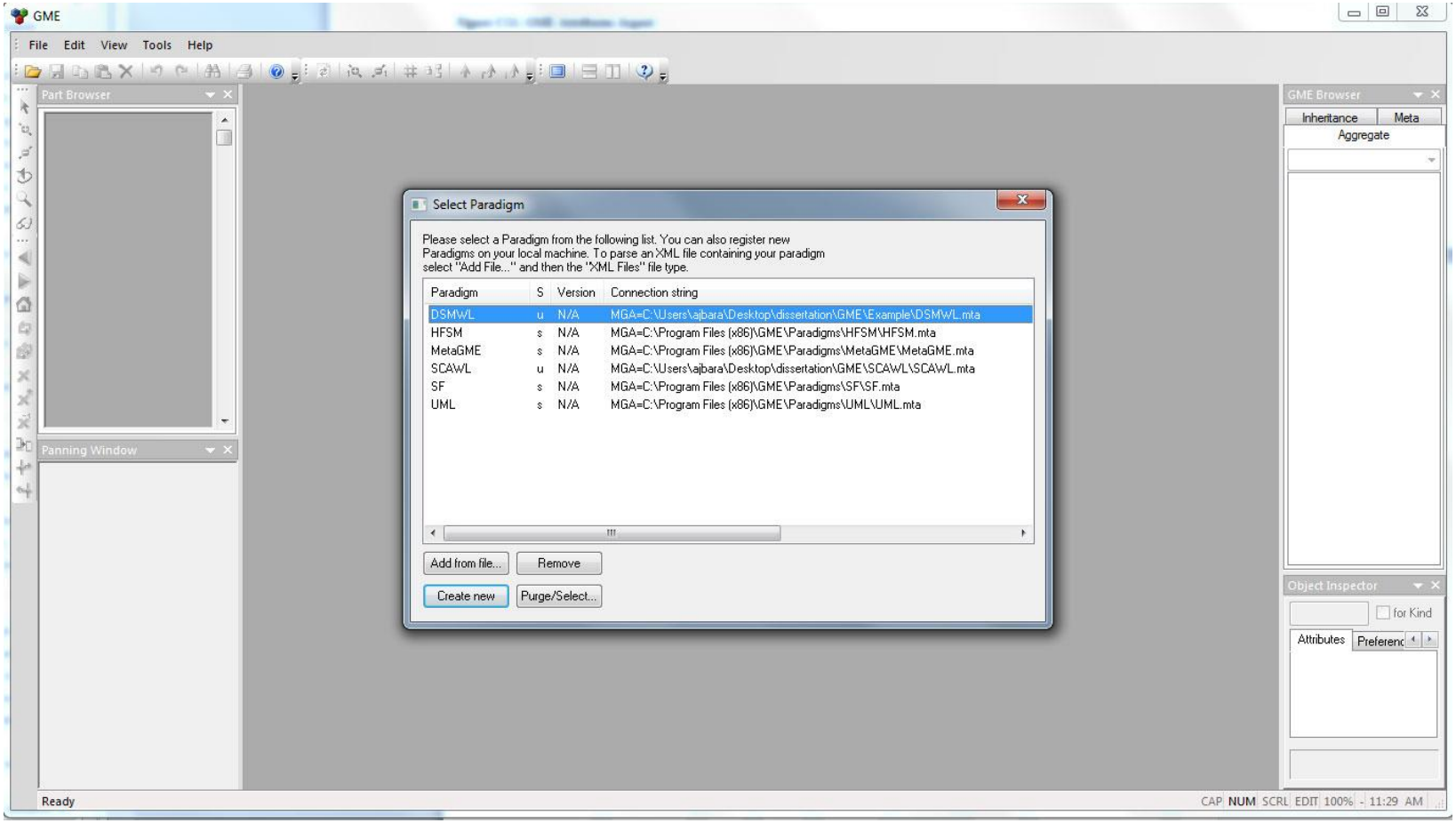

Figure C12: GME Paradigm (Meta-Model) Registration

REFERENCES

[1] G. Hemingway, H. Neema, H. Nine, J. Sztipanovits, and G. Karsai, "Rapid Synthesis of High-level Architecture-Based Heterogeneous Simulation: a Model-Based Integration Approach," *Simulation*, vol. 88, no. 2, pp 217-232, 2012.

[2] D. Garlan, K. M. Carley, B. Schmerl, M. Bigrigg, and O. Celiku, "Using Service-Oriented Architectures for Socio-Cultural Analysis," in *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE2009)*,Boston, USA, 2009.

[3] HLA Evolved Working Group. http://standards.ieee.org/findstds/standard/1516-2010.html

[4] James Davis, "GME: The Generic Modeling Environment," *in Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, New York, USA, 2003.

[5] Paul A. Fishwick, "Toward an Integrative Multimodeling Interface: A Human-Computer Interface Approach to Interrelating Model Structures," *Simulation*, vol. 80, no. 9, pp 421-432**,** 2004.

[6] Christopher Walton, *Agency and The Semantic Web*, Oxford University Press, New York, USA, 2007.

[7] Java Agent Development Framework (JADE). http://jade.tilab.com/

[8] Evan Munsing and Christopher Lamb, "Joint Interagency Task Force - South: The Best Known, Least Understood Interagency Success," Institute for National Strategic Studies, National Defense University, Washington, DC, USA, 2011.

[9] A. Maria, "Introduction to Modeling and Simulation," *Proceedings of the 29th Winter Simulation Conference*, Washington, DC, USA, 1997.

[10] Sajad Haider and Alexander H. Levis, "Effective Course-of-Action Determination to Achieve Desired Effects," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* , vol. 37, no. 6, pp. 1140-1150, 2007.

[11] K. Carley, "On the Evolution of Social and Organizational Networks," *Steven B. Andrews and David Knoke (Eds.)*, vol. 16, special issue of Research in the Sociology of Organizations., pp. 3-30, Greenwich, CT, USA, 1999.

[12] A. H. Levis, Computational Modeling of Cultural Dimensions in Adversary Organizations, Final Technical Report Submitted to Air Force Office of Scientific Research, System Architectures Lab, George Mason University, Fairfax, VA, 2010.

[13] Generic Modeling Environment (GME). http://www.isis.vanderbilt.edu/Projects/gme/

[14] Alexander H. Levis, Abbas K. Zaidi, and Mohammad F. Rafi, "Multi-modeling and Meta-modeling of Human Organizations," *Proceedings of the 4th International Conference on Applied Human Factors and Ergonomics*, San Francisco, CA, 2012.

[15] Business Process Model and Notation. http://www.omg.org/spec/BPMN/

[16] Business Process Execution Language. https://www.oasis-open.org/committees/wsbpel/

[17] M. Mernik, J. Heering, and A. M. Sloane, When and how to develop domain-specific languages, ACM Computing Surveys (CSUR), vol. 37, pp. 316–344, 2005.

[18] Mathias Weske and Gottfried Vossen, "Workflow Languages," *Handbook on Architectures of Information Systems*, pp. 359-379, Springer Berlin Heidelberg, Germany, 1997.

[19] Mathias Weske, *Business Process Management: Concepts, Languages, Architectures*, Springer, Berlin, Germany, 2007.

[20] M. Uschold, M. Gruninger, M. Uschold, and and M. Gruninger, Ontologies: Principles, Methods and Applications, *Knowledge Engineering Revie*w, vol. 11, pp. 93-136, 1996.

[21] I. Niles and A. Pease, Towards a standard upper ontology, *Proceedings of the international conference on Formal Ontology in Information Systems*, New York, NY, 2001.

[22] K. Kotis and M. Lanzenberger, "Ontology Matching: Current Status, Dilemmas and Future Challenges," *Procedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008)*, Barcelona, Spain, 2008.

[23] V. Mascardi, A. Locoro, and P. Rosso, "Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 609-623, 2010.

[24] Marcos Martínez Romero, José Manuel Vázquez Naya, Javier Pereira Loureiro and Norberto Ezquerra, *Ontology Alignment Techniques*, IGI Global, 2008

[25] Ruben Prieto-Diaz, "Domain Analysis: An Introduction," *Software Engineering Notes*, vol. 15, no. 2, p. 47, 1990.

[26] X. Ferré and S. Vegas, "An Evaluation of Domain Analysis Methods," *Proceedings of the 4th International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design*, Heidelberg, Germany, 1999.

[27] Trip Denton, Edward Jones, Srini Srinivasan, Ken Owens and Richard W. Buskens, "NAOMI – An Experimental Platform for Multi–modeling," *Proceedings of the 11th International MoDELS Conference*, Toulouse, France, 2008.

[28] Juan de Lara and Hans Vangheluwe, "AToM3: A Tool for Multi-Formalism and Meta-Modeling, Fundamental Approaches to Software Engineering," *LNCS*, vol. 2306, pp 174-188, 2002.

[29] V. Vittorini, M. Iacono, N. Mazzocca and G. Franceschinis, "The OsMoSys approach to multi-formalism modeling of systems," *Software and Systems Modeling*, vol. 3, no. 1, pp 68-81, 2004.

[30] SIMTHESys. http://www.dem.unina2.it/simthesys/

[31] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster, "The Möbius Modeling Tool," *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, Aachen, Gemany, pp. 241-250, 2001.

[32] E. Kapsammer et al., "Lifting metamodels to ontologies: A step to the semantic integration of modeling languages," *Information Systems Journal*, vol. 4199, pp. 528–542, 2006.

[33] M. Saeki and H. Kaiya, "On Relationships Among Models, Meta Models, and Ontologies," *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling*, Portland, Oregon, 2006.

[34] P. Höfferer, "Achieving business process model interoperability using meta-models and ontologies," *Proceedings of the 15th European Conference on Information Systems (ECIS 2007)*, St. Gallen, Switzerland, 2007.

[35] Akos Ledeczi, Miklos Maroti, Arpad Bakay, Gabor Karsai, Jason Garrett, Charles Thomason, Greg Nordstrom, Jonathan Sprinkle and Peter Volgyesi, "The Generic Modeling Environment," *Proceedings of Intelligent Signal Processing Conference (WISP)*, Budapest, Hungary, 2001.

[36] Joseph D. Novak and Alberto J. Cañas, The Theory Underlying Concept Maps and How to Construct and Use Them, Technical Report IHMC CmapTools, Florida Institute for Human and Machine Cognition, Pensacola, FL, 2006.

[37] Guillermo Arango, "A Brief Introduction to Domain Analysis," *Proceedings of ACM symposium on applied computing*, New York, USA, 1994.

[38] G. Arango and R. Prieto-Diaz. *Domain analysis: Concepts and research directions*, IEEE Computer Society Press, 1989.

[39] S. Wartik R. P. Diaz, "Criteria for Comparing Domain Analysis Approaches," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 3, pp. 403-432, 1991.

[40] R. Prieto-Diaz, Domain Analysis for Reusability, Software reuse: emerging technology, pp. 347-353, IEEE Computer Society Press, Los Alamitos USA, 1989

[41] Guillermo Arango, "Domain Analysis: From Art Form To Engineering Discipline," *Proceedings of the 5th international workshop on Software specification and design*, New York, NY, 1989.

[42] S. White, "Domain Engineering: The Challenge, Status, and Trends," *Proceedings of the IEEE Symposium and Workshop on Engineering of Computer-Based Systems*, Friedrichshafen, Germany, 1996.

[43] Ruben Prieto-Diaz , DARE A Domain Analysis and Reuse Environment, Technical Report submited to DARPA, Fairfax, VA, USA 1995.

[44] Maarit Harsu, A survey on domain engineering, Technical Report, Institute of Software Systems, Tampere University of Technology, Hervanta, Finland, 2002.

[45] Web Ontology Language (OWL). http://www.w3.org/TR/owl-ref/

[46] The Protégé ontology editor and knowledge acquisition system. http://protege.stanford.edu/

[47] Query Language for RDF (SPARQL). http://www.w3.org/TR/rdf-sparql-query/

[48] Morgan Kaufmann, *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Academic Press, San Diego, CA, USA, 2001.

[49] J. Becker et al., "Evaluating the Expressiveness of Domain Specific Modeling Languages using the Bunge-Wand-Weber Ontology," *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS)*, Honolulu, HI, USA, 2010.

[50] OntoViz. http://protegewiki.stanford.edu/wiki/OntoViz

[51] SWRL A Semantic Web Rule Language. http://www.w3.org/Submission/SWRL/

[52] OWLVIZ. http://protegewiki.stanford.edu/wiki/OWLViz

[53] SQWRL a query language for OWL. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL

[54] Fuseki. http://jena.apache.org/documentation/serving_data/index.html

[55] Jena. http://jena.apache.org/

[56] Richard N. Taylor, Nenad Medvidovic, Eric Dashofy , *Software Architecture: Foundations, Theory, and Practice*, John Wiley & Sons, Hoboken, USA, 2010.

[57] C. Ouvans et al., "From BPMN Process Models to BPEL Web Services," *Proceedings of the International Conference on Web Services*, Washington, DC, USA, 2006

[58] Stephen White, "Using BPMN to Model a BPEL Process," *BPTrends*, vol. 3, pp 1-18, 2005

[59] GME Interpreter. http://w3.isis.vanderbilt.edu/projects/gme/Tutorials/Lesson3.html

[60] M. F. Rafi, Meta-Modeling for Multi-Model Integration, MS Thesis, Department Computer Science, George Mason University, 2010.

[61] Peter Pachowicz, Lee Wagenhals, John Pham and Alexander Levis, "Building and Analyzing Timed Influence Net Models with Internet-Enabled Pythia", *Modeling and Simulation for Military Operations II*, SPIE Press, 2007

[62] WebTas. http://www.issinc.com/programs/webtas.html

CURRICULUM VITAE

Ahmed Abu Jbara received his Bachelor of Science in Computer Engineering from Islamic University – Gaza and a Master of Science in Electronic-Commerce/Computer Science from George Mason University. Ahmed has more than 10 years of experience in academia, research and industry.