Probabilistic Ontology Reference Architecture and Development Methodology

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

by

Richard J. Haberlin Jr.
Master of Science
Naval Postgraduate School, 1997
Bachelor of Science
United States Naval Academy, 1990

Co-Director: Kathryn B. Laskey / Dr. Paulo C. G. da Costa, Professor
Co-Director: Paulo Cesar G. da Costa, Associate Professor
Department of Systems Engineering and Operations Research

Fall Semester 2013
George Mason University
Fairfax, VA

DEDICATION

This dissertation is dedicated to my family who provide the inspiration to continue.
Mundus tuus est.

.

ACKNOWLEDGEMENTS

Over the past four years I have received support and encouragement from a great number of individuals. Dr. Kathryn Laskey, who first introduced me to probabilistic ontologies, has been a mentor, colleague, and friend and has made this process a rewarding journey. Dr. Paulo Costa, in the role of Co-Director always found time in his incredibly complex schedule to provide guidance and support. I also appreciate the long hours and sound advice of the rest of my committee: Dr. Alex Brodsky, Dr. Andy Loerch, and Dr. Dave Schum. Together they offered perspective and insight that allowed this work to be a valuable contribution to the community. I would also like to thank Dr. Rommel Carvalho for blazing a path in this emerging field and for always finding time to review material and answer my many questions. Similarly, Shou Matsumoto, while busy with his own studies, was always available to assist with the somewhat quirky code of UnBBayes. I am also grateful to the case study participants who provided their time and insight to the evaluation of my research. Most importantly, Linda, Joe and Kate have put up with four years of a somewhat distracted husband and father. Their bountiful support allowed consistent progress. The juice will be worth the squeeze.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Artificial Intelligence	AI
Area of Operations	AOR
Bayesian Networks	BN
Computer Science	CS
Conditional Probability Table	CPT
DARPA Agency Markup Language	DAML
Decision Maker	DM
Decision Support System	DSS
Department of Defense	DOD
DOD Architecture Framework	DODAF
Entity Relationship Diagram	ERD
Expressive Probabilistic Language	EPL
Extensible Markup Language	XML
First-Order Logic	FOL
Geographical Information System	GIS
Icam Definition for Function Modeling	IDEF0
Knowledge Base	KB
Knowledge-Based Model Construction	KBMC
Knowledge Engineering with Bayesian Networks	KEBN
Knowledge Interchange Format	KIF
Local Probability Distribution	LPD
Markov Chain Monte Carlo	MCMC
Markov Logic Network	MLN
Measure of Effectiveness	MOE
Measure of Performance	MOP
MEBN Fragment	MFrag
MEBN Theory	MTheory
Military Ship	MilShip
Model-Based Systems Engineering	MBSE
Moving Target Indicator	MTI
Multi-Entity Bayesian Networks	MEBN
Probabilistic Ontology	PO
Probabilistic Ontology Development Methodology	PODM
Probabilistic Relational Model	PRM
Probabilistic Web Ontology Language	PR-OWL
Reference Architecture for Probabilistic Ontology Development	RAPOD

Resource Descriptor Framework (Schema)	RDF(S)
Semantic Technology	ST
Situation-Specific Bayesian Network	SSBN
Statement of Work	SOW
Subject-Matter Expert	SME
System Development Life Cycle	SDLC
Systems Engineering and Operations Research	SEOR
Systems Modeling Language	SysML
Uncertainty Reasoning Process for Semantic Technologies	URP-ST
Unified Modeling Language	UML
Unified Modeling Process for Semantic Technologies	UMP-ST
Unified Process	UP
Web Ontology Language	OWL
Worldwide Web Consortium	W3C

ABSTRACT

PROBABILISTIC ONTOLOGY REFERENCE ARCHITECTURE AND DEVELOPMENT METHODOLOGY

Richard J. Haberlin Jr., Ph.D.

George Mason University, 2013

Dissertation Co-Directors: Dr. Kathryn B. Laskey / Dr. Paulo C. G. da Costa

The use of ontologies is on the rise, as they facilitate interoperability and provide support for automation. Today, ontologies are popular for research in areas such as the Semantic Web, Knowledge Engineering, Artificial Intelligence and knowledge management. However, many real world problems in these disciplines are burdened by incomplete information and other sources of uncertainty which traditional ontologies cannot represent. Therefore, a means to incorporate uncertainty is a necessity. Probabilistic ontologies extend current ontology formalisms to provide support for representing and reasoning with uncertainty. Traditional ontologies provide a hierarchical structure of entity classes and a formal way of expressing their relationships with first-order expressivity, which supports logical reasoning. However, they lack built-in, principled support to adequately account for uncertainty. Applying simple probability annotations to ontologies fails to convey the structure of the probabilistic representation. Similarly, other less expressive probability schemes do not convey the ontology structure, and are also inadequate. Representation of uncertainty in real-world problems requires *probabilistic* ontologies, which integrate the inferential reasoning power of probabilistic representations with the first-order expressivity of ontologies. Developing a probabilistic ontology is more complex than simply assigning probability to a class instantiation or representing a probability scheme using ontology constructs. Standard ontological engineering methods provide insufficient support for the complexity of probabilistic ontology development. Therefore, a specific methodology is needed to develop probabilistic ontologies from conceptualization to implementation. This dissertation introduces a systematic approach to probabilistic ontology development facilitated through a reference architecture which focuses on evolving a traditional ontology from conceptualization to probabilistic ontology implementation for real-world problems. The Reference Architecture for Probabilistic Ontology Development captures, catalogues and defines the components necessary for probabilistic ontology development. It includes an efficient, teachable, and repeatable Probabilistic Ontology Development Methodology for the development, implementation and evaluation of explicit, logical and defensible probabilistic ontologies developed for knowledge-sharing and reuse in a given domain.

CHAPTER ONE: INTRODUCTION

The use of ontologies is on the rise, as they facilitate interoperability and provide support for automation. Today, ontologies are popular for research in areas such as the Semantic Web [9], Knowledge Engineering, Artificial Intelligence (AI) and Knowledge Management [56]. However, many real world problems in these disciplines are burdened by a lack of complete information and other sources of uncertainty [161], which traditional ontologies cannot represent. Therefore, a means to incorporate uncertainty is a necessity.

Ontologies provide a hierarchical structure of entity classes and a formal way of expressing their relationships with first-order expressivity, which supports logical reasoning. However, they lack built-in, principled support to adequately account for uncertainty. Applying simple probability annotations to ontologies fails to convey the structure of the probabilistic representation. Similarly, other less expressive probability schemes do not convey the ontology structure, and are also inadequate. Representation of uncertainty in real-world problems requires *probabilistic* ontologies, which integrate the inferential reasoning power of probabilistic representations with the first-order expressivity of ontologies. Developing a probabilistic ontology is more complex than simply assigning probability to a class instantiation or representing a probability scheme using ontology constructs. The Semantic Technologies (ST) community needs a comprehensive methodology for the development, implementation, and evaluation of probabilistic ontologies. Traditional ontological engineering helps to ensure that ontologies developed for knowledge-sharing and reuse are explicit, logical, and defensible. However, these standard ontological engineering methods provide insufficient support for the complexity of probabilistic ontology development described above. Therefore, a specific methodology is needed to develop probabilistic ontologies from conceptualization to implementation.

To illustrate the problem, suppose there exists an ontology of organisms. Within this ontology is a Mammal class and Human sub-class. Entities of the Human class *usually* have attributes that include two arms, two legs, 10 fingers, 10 toes, etc. Yet humans have alternative numbers of digits for many reasons (e.g. injuries, genetics, birth defects), but are nonetheless human. Suppose Joe is born with eight toes. The difficulty in representation for the Joe instance stems from the fact that the premise of a valid argument (Humans have 10 toes) can be uncertain, in which case validity of the argument imposes no condition on the certainty of the conclusion (Joe is Human). Probabilistic ontologies address this issue by extending current ontology formalisms to provide support for representing and reasoning with uncertainty.

There is a large body of research on development of traditional ontologies, but these methodologies are not suitable for production of probabilistic ontologies, as described above. Development of probabilistic ontologies without the benefit of a methodology is a risky venture. Solutions tend to be ad-hoc and without consideration of interoperability. The literature on engineering probabilistic ontologies is extremely limited. Carvalho notes,

> *"It would be interesting to have a tool guiding the user on the steps necessary to create a probabilistic ontology and link this documentation to its implementation* ... [19]*."*

This dissertation introduces a systematic approach to probabilistic ontology development facilitated through a reference architecture which focuses on evolving a traditional ontology from conceptualization to probabilistic ontology implementation for real-world problems. The Reference Architecture for Probabilistic Ontology Development captures, catalogues and defines the components necessary for probabilistic ontology development. It includes an efficient, teachable, and repeatable Probabilistic Ontology Development Methodology for the development, implementation and evaluation of explicit, logical and defensible probabilistic ontologies developed for knowledge-sharing and reuse in a given domain.

1.1 Reference Architecture for Probabilistic Ontology Development

The Reference Architecture for Probabilistic Ontology Development (RAPOD) facilitates a unification of effort between multiple disciplines including probabilists, logicians, decision analysts and computer scientists to create a solution architecture for a domain-specific problem. A reference architecture guides and constrains architecture solutions, providing a blueprint that may be used for solutions scalable to any size domain. The RAPOD provides synergy of effort by identifying concepts, processes, languages, theories and tools for designing and maintaining probabilistic ontologies. It describes each of the components required for a functional probabilistic ontology and defines the criteria to be satisfied by any set of selected tools and methods.

Current ontological engineering practice ensures ontologies developed for knowledge-sharing and reuse are explicit, logical and defensible. The RAPOD is a cache of concepts and tools for development and implementation of *probabilistic* ontologies, which organizes ontological and probabilistic ontological engineering methodologies into categories and provides a collection of development knowledge for this rapidly evolving domain. It describes the purpose and relationships of each component required for a functional probabilistic ontology, and establishes the criteria to be satisfied by any set of selected tools and methods. This pragmatics is fully described in Chapter 3.

1.2 Probabilistic Ontology Development Methodology

Within the Reference Architecture for Probabilistic Ontology Development, a Probabilistic Ontology Development Methodology (PODM) is defined that specifically addresses the evolution of requirements into an ontology that is probabilistically-integrated. As introduced above, a probabilistically-integrated ontology combines the inferential reasoning power of probabilistic representations with the first-order expressivity of ontologies. A key component of that methodology is a detailed Construction Process, which explicitly describes the iterative tasks required to produce a probabilistic ontology with in-situ evaluation steps to ensure continuous operation for inferential reasoning. Synergy acquired through the use of the RAPOD and PODM allows efficient, repeatable, and defensible development of probabilistic ontologies.

Traditional ontological engineering facilitates the development of explicit, logical and defensible ontologies for knowledge-sharing and reuse. This research delivers a Probabilistic Ontology Development Methodology grounded in Model-Based Systems Engineering (MBSE) principles. Tasks associated with ontological engineering and the implementation of probability are applicable to both traditional and agile Systems Development Life Cycle (SDLC) processes. Within an SDLC framework, execution of the PODM specifically addresses the evolution of requirements into an ontology that is probabilistically integrated. The detailed PODM explicitly describes the iterative tasks required to produce a Probabilistic Ontology (PO) with in-situ evaluation steps to ensure continuous operation of a relational model produced for inferential reasoning.

1.3 Case Study Evaluation

The PODM is an early attempt within the ST community to define and evaluate an efficient, repeatable and teachable process for development of a PO. To demonstrate its utility, a case study was employed to demonstrate teachability, increased efficiency, and an improved final product. Participants in the case study were graduate students of George Mason University. The purpose of this study was:

1. Evaluate the effectiveness of the methodology;
2. Evaluate the teachability of the methodology.

The test population for the analysis was a group of Systems Engineering and Operations Research (SEOR) and Computer Science (CS) graduate students possessing varying degrees of experience with ontologies and probabilistic ontologies. A brief demographic survey offered at the commencement of the study captured student personal information.

Then, each participant completed two probabilistic ontology production exercises, one before introduction to the PODM and one after. At the conclusion of the exercises, each participant completed a post-project survey to capture individual recommendations and comments about the methodology.

The case study was completed by a total of three SEOR/CS participants. Each was provided with materials, training, and a statement of work (SOW). These items are summarized in Chapter 5 and fully described in Appendix E. The case study focused on evaluating the hypothesis of a causal linkage between the PODM and better probabilistic ontologies produced more efficiently. Further, qualitative descriptions of the participants' efforts were captured to illustrate improvements in clarity and efficiency introduced with the PODM. Finally, because probabilistic ontology development is in its infancy as an engineering discipline, there is no standard against which the PODM may be evaluated. The case study illustrates those attributes necessary to demonstrate utility in producing the desired model. The three questions answered by this case study are:

1. Does the methodology produce "better" probabilistic ontologies than those produced without using the PODM?
2. Does the methodology allow "more efficient" development of probabilistic ontologies than development without the PODM?
3. Is the methodology "teachable" to a population of graduate students?

Definitions of the applicable measures of effectiveness (MOE) are captured in Table 1. The MOEs capture the utility of a methodology by demonstrating the value in error reduction and reduced development time.

Table 1 - Case Study Measures of Effectiveness

"Better"
Fewer logical errors
Fewer relational omissions
Better documentation
Runs test cases correctly
"More Efficient"
Less time to complete
More focused effort
Fewer false starts
"Teachable"
Appropriately employ methodology

Further decomposition of these MOEs is discussed in Chapter 5. It is important to note that a methodology is not useful if is complex beyond the ability of its intended user group. The qualitative "Teachable" MOE was used to determine the utility of the PODM within the target demographic.

1.4 Examples

Several examples are used throughout this dissertation for tutorials, the case study, and an illustrative example. These are briefly introduced below.

1.4.1 Chest Clinic

The Chest Clinic Tutorial generally follows the Probabilistic Ontology Development Methodology from ontology engineering to PO operation. The problem is a familiar academic example, in which a patient experiencing symptoms of a chest ailment enters a clinic for diagnosis. The physician utilizes the answers to a few questions regarding the patient's recent travel to evaluate the likelihood that the patient has the flu or is suffering from some other virus. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus. The first part of the tutorial initiates the user to the Protégé 4.1 ontology development software tool developed by Stanford University [154]. Next, the tutorial demonstrates use of the UnBBayes software tool to create an operable probabilistic ontology to support diagnosis of the patient [159]. The probabilistic ontology allows the possibility that the patient has visited multiple locations with varying levels of local infection. Each visit is captured as evidence in the form of a First-Order Logic (FOL) sentence.

1.4.2 Vehicle Identification

Intelligence, surveillance and reconnaissance aircraft employ a suite of sensors to allow the operator to classify detected targets. In this example problem, a decision support system must be developed that provides the most likely vehicle type (wheeled or tracked) based on incoming evidence. The model may be used to infer the vehicle type from moving target indicator (MTI) and imaging sensor reports, weather reports, and geographical information system (GIS) reports. Vehicles may travel on-road, off-road, or on very-rough terrain. Weather affects imaging sensors and is characterized as clear or cloudy. Considering given domain knowledge, the participant was tasked to develop a probabilistic ontology for military vehicles that infers vehicle type (wheeled or tracked) from the MTI and imaging sensor reports, weather reports, and GIS reports.

1.4.3 Terrorist Crewmember

Crewmembers of merchant vessels are regularly multinational and transient. This is one possible way that terrorists or terrorist organizations can smuggle personnel or material into target countries. Using information about an individual crewmember's relations, influences, and group associations may provide insight into the likelihood of an individual sailor being involved in terrorism. While some affiliations may increase the likelihood that an individual may join a terrorist group and attempt access to a target country via merchant ship, there is always the uncertainty that comes from the human condition. Further, each crewmember may participate in multiple organizations (some of which may be associated with terrorism) or have multiple friends and relatives (some of whom may participate in terrorism). Uncertainty associated with the multitude of factors affecting the crewmember's context must be captured conditionally. Considering given domain knowledge, the participant was tasked to develop a probabilistic ontology to support inference about the likelihood an individual sailor is involved in terrorism.

1.4.4 Military Ship (MilShip)

Naval commanders often receive uncertain or incomplete information about contacts of interest. An ontology of warship classes may be created for an Area of Operations (AOR) from which the most likely ship class is inferred. As information arrives, either in the form of a report or from organic unit sensors, an updated inference is developed. Reports may be received about the contact of interest size and/or type; in the absence of a size report, organic sensors may introduce information about ship length and displacement from which a generalized size may be determined. Other organic sensors may provide additional evidence about the nationality, weapons and sensors aboard. It is not uncommon that military ships share similar capabilities among varied classes. There is a need for military commanders to distinguish the specific class of a potential adversary ship to best prepare for the threat. European warships are used as surrogates to represent hostile forces with ship characteristics taken from open-source information. Given a varied amount of data about a contact of interest, this probabilistic ontology is used to support inferential reasoning for assessing key aspects such as the most probable ship class from a set of ship classes, or how likely it is that the ship is a member of the set.

1.5 Terminology

Before delving into the specifics of the RAPOD and PODM it is necessary to clarify terminology. The simplified definitions below are used throughout this work.

Reference Architecture: A reference architecture defines the fundamental components of a domain and the relationships between them that benefit product development, software reuse, and maintenance [66].

Taxonomy: A taxonomy is a classification structure for ordering objects into categories [40].

Ontology: An ontology is an explicit specification of a conceptualization [60]. It should include well-define syntax and semantics, efficient reasoning support, and sufficient expressive power [3].

Probabilistic Ontology: A probabilistic ontology is an explicit, formal knowledge representation that expresses knowledge about a domain of application, including uncertainty about all forms of knowledge [29].

Methodology: A methodology is a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed [74].

Process: A process is a set of activities that collectively perform a function.

Activity: An activity is a constituent undertaking of a process [73].

Task: A task is the smallest unit of work subject to management accountability [56]. It is a well-defined work assignment for one or more project members. Related tasks are grouped to form activities [73].

Additional detail regarding reference architectures, ontologies and probabilistic ontologies are provided in Chapter 2.

1.6 Organization of this Work

This work is structured as follows. Chapter 2 summarizes the state of the probabilistic ontology development domain and related concepts by introducing reference architectures, ontological engineering methods, evidential reasoning principles, knowledge engineering constructs, and case study approaches in the literature. Chapter 3 provides a comprehensive description of the Reference Architecture for Probabilistic Ontology Development and an example of its implementation to produce an architecture for the Military Ship PO example problem. Chapter 4 details the Probabilistic Ontology Development Methodology using the Military Ship PO as a running example. Chapter 5 summarizes execution and results of the PO Case Study Evaluation to examine quality, efficiency, and teachability. Finally, Chapter 6 summarizes this work and provides an avenue for future research. The interested reader may find detailed supplementary information in the appendices.

CHAPTER TWO: LITERATURE REVIEW

The Semantic Technology community is lacking a comprehensive methodology for the development, implementation and evaluation of probabilistic ontologies. While it is recognized that ontology use is on the rise, and a means to incorporate uncertainty is a necessity, there have been few attempts to produce a methodology for production of probabilistic ontologies. Bergman states,

> *"…what is most striking…is the paucity of [methodologies] and the generality of those that do exist* [8]*."*

If that is the case for deterministic ontologies, it follows that an even greater deficit holds for probabilistic ontology engineering methodologies. In part, this may be a result of divergence within the ST community on the best probabilistic model to represent uncertainty. Elimination of this barrier calls for a consistent, structured methodology with broad applicability to alternative probabilistic relational representations. Current traditional ontological engineering methodologies provide insufficient support for the complexity of probabilistic ontology development. A Model-Based Systems Engineering approach to probabilistic ontology development will allow development of probabilistic ontologies that are explicit, logical and defensible.

Before delving into the MBSE methodology that allows development, implementation and evaluation of probabilistic ontologies in a systematic, comprehensive and repeatable process, it is first necessary to understand the components that comprise a probabilistic ontology, namely a traditional domain ontology and a means to incorporate uncertainty. This background is followed by a review of probabilistic ontology literature, including the only known methodology specifically designed to produce one. Finally, an overview of case study research methods provides detail on producing relevant case studies through rigorous design, collection, and analysis.

2.1 Reference Architectures

At the highest level of abstraction, a reference architecture framework defines structural organization and views associated with an architecture. An instantiation of a reference architecture defines a solution architecture, which describes the composition of a system by providing the selection of structural elements, their interfaces, and their behavior [95]. More formally, the ISO/IEC/IEEE 42010 Conceptual Model of Architecture Description defines the term Architecture Framework as [122]:

> *"An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community."*

As enterprises become more global, it is important to develop a mechanism to depict contributions of and relationships between entities. Further, products produced under differing architecture styles impede identification of opportunities for synergistic interoperability and integration. An architectural framework is a mechanism to streamline the complex architecture design process and assist in the evaluation of different architectures, enabling a better architecture to be developed for a particular domain. An architecture framework captures lessons learned and best practices, acknowledges wisdom and presents a set of services, design concepts, components and configurations applicable to a broad range of specific architectures.

2.1.1 Definition

The authoritative definition is provided by the Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII) which states, "[A] Reference Architecture is an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions [122]." Alternative literature generally aligns with this definition [137][127][141][93][46][101] [4][95] and extends the discussion into the purpose behind its production. Ultimately, the purpose of the reference architecture is to identify and define the fundamental components of the domain and the relationships between them [66][59]. It therefore acts as a predefined construction template for a particular context that may be used to produce similar solution architectures. The solutions exist on the continuum from very abstract to concrete [93] and may be produced from previous projects [46].

2.1.2 Systems Architecting

Sage defines systems architecting as an iterative systems engineering process to create complex, unprecedented systems that deliberately copes with uncertainty [137]. The systems architecting approach for reference architectures recommended by the OASD/NII includes:

1. Researching existing reference architectures for a representative sample;
2. Examining candidates to understand what the reference architecture is used for (goals, objectives, characteristics and key elements for common threads and best practices);
3. Developing the new reference architecture.

A solution architecture for a particular context is guided and constrained by the reference architecture by replacing its abstract elements with real-world components applicable to the domain.

2.1.3 Utility of Reference Architectures

Benefits associated with implementing a reference architecture are many, and can generally be categorized under construction and interoperability. During construction of a system architecture, the reference architecture acts as a template that improves the developer's understanding of the system and clarifies communications with the stakeholders [66]. The ability to view the entire system and its component interrelationships improves maintenance efficiency [66][59] and aids in analysis of design tradeoffs [59]. Interoperability is improved by creating a language-agnostic representation of the system upon which design concepts can be mapped for an analysis of alternatives. The reference architecture can also serve as a blueprint for meta-modeling that allows rapid synthesis of larger, complex systems [101].

2.2 Ontologies

Ontologies are widely used in Knowledge Engineering, Artificial Intelligence, Computer Science and Knowledge Management to map knowledge for a given domain, and are a key element in establishing reusable knowledge-based representations of the world. Their level of complexity can range from a simple taxonomy of agreed-upon terms to a comprehensive formal ontology with links representing relationships between entities. It is the latter that is of interest to this research as this recognizes the full potential of the Semantic Web vision in which, "…machine reasoning will be ubiquitous and devastatingly powerful [9]." Additionally, ontologies provide utility in software specification by establishing agreements about knowledge (e.g. assumptions and requirements) [60][110]. Using an ontology, developers share a common frame of reference with respect to entities, attributes, and their relationships which minimizes redundancy, reduces the likelihood of errors, and supports interoperability, extensibility and maintainability. Before delving into the newer domain of probabilistic ontologies, it is first necessary to establish a baseline of knowledge for the current state of ontological development.

2.2.1 Ontological Engineering

The field of ontological engineering (or ontology engineering) encompasses the activities that make-up the ontology development process, the ontology life cycle, and the methodologies, tools, and languages for building ontologies[56] [86] [86]. Characterizing ontology development in engineering terms aligns with MBSE principles which formalize application of modeling to support system requirements, design, analysis, verification and validation activities[76] throughout the development life cycle, from conceptualization to retirement. Bergman defines selection criteria for ontology reuse, and a unified methodology using an engineering approach [8]. After providing a definition of an ontology, the following sections provide a brief review of ontological engineering available in the literature.

2.2.1.1 Ontology Defined

The modern colloquial use of the term ontology stems from the realm of philosophy in which ontology is the branch of metaphysics concerned with the nature of existence. However, there is disagreement on the boundary of existence, specifically as to whether something actually has to exist, or can be conceptualized. The field of information systems has adopted the term to refer to a formal representation of knowledge about a domain. A commonly accepted definition of ontology presented by Gruber and adopted by Kishore et al. states,

"An ontology is an explicit specification of a conceptualization [60][86]*."*

This implies that anything that can be imagined may also be included in an ontology. Mizoguchi & Ikeda add that for the knowledge-base community, an ontology is "a theory of vocabulary/concepts used for building artificial systems [110]." They further break down the definition of ontologies by specifying the domain of application for the definition. This delineation is expounded by Kishore et al. who introduce the differences between philosophical ontology and computational ontologies [86]. Smith, on the other hand, believes that ontologies should represent entities as they exist in reality and delineates "good ontologies" from "bad ontologies" using this measure [151]. Costa describes a definition, grounded in Gruber, that anticipates the need to introduce probability to the ontology [29]. According to Costa, an ontology is an explicit, formal representation of knowledge about a domain of application. This includes a) types of entities that exist in the domain; b) properties of those entities; c) relationships among entities; and d) processes and events that happen with those entities; where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application. Finally, Keet defines an ontology as

> *"a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world* [83]*."*

Because of its seamless transition potential to the inclusion of uncertainty, the definition of ontology provided by Costa is most relevant to the creation of probabilistic ontologies.

2.2.1.2 Design Criteria

To enhance ontology reusability, it is important that a set of design criteria be met that ensures domain agnosticism and supports heterogeneous hardware implementation. Gruber provided a first attempt at such criteria which included clarity, coherence, extendibility, minimal encoding bias, and minimal ontological commitment [60]. His definitions for these criteria relative to ontology engineering are given below.

- Clarity: An ontology should effectively communicate the intended meaning of defined terms, with objective definitions that are independent of social situations or computational requirements.

- Coherence: An ontology should sanction inferences that are consistent with the definitions, and the defining axioms should be logically consistent. An ontology that allows inference of a contradictory sentence from its axioms is incoherent. However, the ontological engineering community has recognized that global coherence is impractical for large ontologies. Therefore, in these cases it is necessary to resort to contexts or "microtheories" that are internally consistent but may contradict each other. Consistency within a context or "microtheory" remains an important design criterion.
- Extendibility: An ontology should be designed to anticipate the uses of the shared vocabulary, and the representation should be crafted so that the ontology can be extended and specialized monotonically. Extension should be possible without revision of existing definitions.
- Encoding Bias: Encoding bias results when a representation is chosen purely for the convenience of notation or implementation. Therefore, conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding.
- Ontological Commitment: An ontology should make as few claims as possible about the world being modeled, allowing freedom to specialize and instantiate as needed. Ontological commitment minimized by specifying the weakest theory and defining only those terms that are essential to the communication of knowledge consistent with that theory.

Mizoguchi & Ikeda acknowledge the need to mimic industry, but provide no specification for criteria [110]. Keet adds that an ontology is supposed to be implementation independent [83]. These principles of consistency and implementation independence are also germane to the design and construction of probabilistic ontologies.

2.2.1.3 Construction Methodologies

Several construction methods for ontologies have been introduced by the ST community. However, many were designed for specific applications and none have been universally accepted. Kishore et al. introduced the Cue-n-Anchor construction strategy which is an evolutionary design strategy and admittedly not fully structured [86]. Instead it provides a set of guidelines to assist the developer in producing a continuously evolving, useful ontology. Keet suggests that methodologies may be grouped according to the context in which the ontology will be developed and implemented [83]. She further describes a "bottoms-up" development approach through ontology learning. Bergman discusses eight leading ontology development methodologies, including: Cyc, TOVE, IDEF5, ONIONS, COINS, METHOLTOLOGY, OTK, and UPON [8]. His sample flow diagrams for general engineering principles applied to ontology development provide a useful overview of common steps necessary in all methods. More recently, Gomez-Perez et al. also describe Cyc, METHONTOLOGY and TOVE, as well as introduce Uschold and King's method, KACTUS, SENSUS, and On-To-Knowledge [56]. A brief introduction to these methods follows.

- Cyc Method: The Cyc Method is based on the three processes that were used to create the Cyc Knowledge Base (KB) of assertions designed to capture a large portion of what people consider knowledge about the world: manual coding of articles and pieces of knowledge, knowledge coding aided by tools using the knowledge already stored in the Cyc KB, and knowledge codification primarily performed by tools using knowledge already stored in the Cyc KB.
- Uschold and King Method: Uschold and King provided the first method for building ontologies, proposed in 1995 [56]. The method consists of four processes: identify the purpose of the ontology, build the ontology, evaluate the ontology, and document the ontology. The method also requires techniques, methods, and principles for each of the above stages. A significant drawback to this method is the lack of a conceptualization process prior to implementation.
- Gruninger and Fox Method: The Gruninger and Fox methodology was used to build the TOVE ontology project. It was inspired by development of KB systems using FOL. The developer must propose the primary scenario applications in which the ontology will be used, then create natural language questions (competency questions) to determine the scope. Questions and their answers are used to extract the main concepts and their properties, relations, and formal axioms of the ontology. Knowledge is formally expressed in FOL. It is a formal methodology that takes advantage of the robustness of classical logic.
- KACTUS Method: This approach is conditioned by application development. Every time an application is built, the ontology that represents the knowledge

required for the application is refined. It can be developed by reusing other ontologies and can also be integrated into ontologies of later applications.

- METHONTOLOGY Method: METHONTOLOGY enables construction of ontologies at the knowledge level and has its roots in software development process activities. It includes identification of the ontology development process, a lifecycle based on evolving prototypes, and techniques to carry out each activity in the management, development-oriented, and support activities.
- SENSUS-based Method: The SENSUS method was proposed to link domain specific terms to the SENSUS ontology of objects, entities, qualities, and relations commonly encountered in machine translation. Terms that are irrelevant for the new ontology are pruned, leaving the skeleton of a new ontology.
- On-To-Knowledge Method: The basis of the On-To-Knowledge methodology is to apply electronically available information for improving the quality of knowledge management in large and distributed organizations. The method proposes to build an ontology by taking into account how the ontology will be used in further applications, and is therefore highly dependent on the application. On-To-Knowledge also proposes ontology learning for reducing the efforts in construction. This includes identification of goals to be achieved by knowledge management tools and is based on an analysis of usage scenarios. The methodology includes techniques, methods, and principles for each process and indicates the relationships between such processes.

Additional methodologies current in the literature include:

- Knowledge-Engineering Methodology [117]: The Knowledge-Engineering Methodology is an iterative approach of seven steps with a rough pass followed by a series of refinements. Each pass revises and refines the ontology to fill in details.
- DOGMA [153]: The DOGMA engineering approach consists of creation of an ontology base that holds a set of intuitive context specific conceptual relations and a layer of relatively generic ontological commitments that hold the domain rules. The ontology base consists of intuitively plausible domain fact types, represented and organized as sets of context-specific binary conceptual relations called lexons. The ontological commitments mediate between the ontology base and its applications. Each ontological commitment corresponds to an explicit instance of a first order interpretation of a task in terms of the ontology base. Each commitment consists of rules that specify which lexons from the ontology base are visible for usage in this commitment.
- Cue-N-Anchor Guided Strategy [86]: Cue-N-Anchor is an evolving strategy for construction in a heuristic sense, not a pre-specified and fully structured iterative process. Cue-N-Anchor includes a non-fully specified, semi-structured strategy of crisscrossing among the various developments that occur during the process. The overall strategy is evolutionary, heuristic, and guided with seven guidelines to aid the developer.

Gomez-Perez et al. also describe methods for ontology re-engineering and merging, including ONIONS, FCA-Merge and PROMPT. These ST contributions to ontology construction provide a baseline for development of a probabilistic ontology design methodology.

2.2.1.4 Types of Ontologies

Understanding the typology of ontologies aids in proper application to a domain. Mizogushi & Ikeda introduce a simplified categorization, which includes ontology, domain ontology, and general ontology, along with 11 subcategories [110]. The focus of Kishore et al. is also on the use of the ontology, but they distinguish terminological from axiomatic ontologies [86], using Sowa's definitions. The former is an ontology whose categories need not be fully specified by axioms and definitions [152]. The latter is a terminological ontology whose categories are distinguished by axioms and definitions stated in logic or in some computer-oriented language that could be automatically translated to logic [152]. Sowa's definitions of ontologies include: a formal ontology which is a conceptualization whose categories are distinguished by axioms and definitions and are stated in logic to support inference and computation, a prototype-based ontology in which categories are formed by collecting instances extensionally, and a terminological ontology which describes concepts by labels and synonyms without axiomatic grounding [152]. Finally, through her discussion of ontology reuse to support development, Keet examines three types of ontologies, including: foundational, reference and domain [83]. The MBSE-inspired probabilistic ontology design methodology will be applicable to axiomatic ontologies applied to a domain.

2.2.2 Probabilistic Ontology Development

Probabilistic ontologies are used to comprehensively describe knowledge about a domain and the uncertainty embedded in that knowledge in a principled, structured and sharable way [29]. They extend current ontology formalisms to provide support for representing and reasoning with uncertainty. The probabilistic Web ontology language introduced by Costa [29] and advanced by Carvalho [19] provides a roadmap for research into the field of probabilistic ontology development.

2.2.2.1 Probabilistic Ontology Defined

Extending the definition of ontology introduced in Section 2.2.1.1, a probabilistic ontology is "an explicit, formal knowledge representation that expresses knowledge about a domain of application. This includes: a) types of entities that exist in the domain; b) properties of those entities; c) relationships among entities; d) processes and events that happen with those entities; e) statistical regularities that characterize the domain; f) inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge related to entities of the domain; and g) uncertainty about all the above forms of knowledge; where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be reasoned about within the domain of application." [29]

2.2.2.2 Probabilistic Web Ontology Language

Costa introduced the Probabilistic Web Ontology Language (PR-OWL), a Bayesian framework for probabilistic ontologies that provides a basis for plausible reasoning services in the Semantic Web [29]. Multi-Entity Bayesian Networks (MEBN) was chosen as the underlying semantics of PR-OWL due to its expressiveness and flexibility. Carvalho continued where Costa left off by extending PR-OWL to improve compatibility with OWL by mapping OWL properties to PR-OWL random variables and using existing OWL datatypes in PR-OWL, developing an upper ontology capturing the syntax of this connection (PR-OWL 2.0), defining a semantics clearly specifying the PR-OWL to OWL mapping, and developing a proof-of-concept tool to model PR-OWL 2.0 in UnBBayes [19]. With this extension, UnBBayes is now the primary software application for implementation of probabilistic ontologies.

2.2.2.3 Probabilistic Ontology Development

Probabilistic ontology development techniques are still extremely limited in the literature. As noted by Carvalho,

> *"…although there is now substantial literature about what PR-OWL is, how to implement it, and where it can be used, little has been written about how to model a probabilistic ontology* [19]*."*

Traditional ontologies provide a deterministic representation of entities and their relationships, but the existence of uncertainty must be accounted for to provide solutions to real-world problems. Traditional ontologies lack built-in, principled support to adequately account for uncertainty. Probabilistic ontologies extend traditional ontologies to provide support for representing and reasoning with uncertainty by integrating the inferential reasoning power of probabilistic representations with the first-order expressivity of ontologies. Carvalho introduces the Uncertainty Reasoning Process for Semantic Technologies (URP-ST), which involves building a model, populating the knowledge base, and then reasoning with the model. For the modeling phase of this process, he borrows from the Unified Process (UP) of software development to produce an iterative development methodology, the Uncertainty Modeling Process for Semantic Technologies (UMP-ST). The four stages of the UMP-ST (Requirements, Analysis and Design, Implementation, Testing) occur as part of a Probabilistic Ontology Modeling Cycle (POMC). This work is the first methodology specific to development of probabilistic ontologies and was most recently employed by Ravi and Singh in the Risk Prediction domain [134]. Carvalho's work provides a baseline for future research.

2.3 Evidential Reasoning

If all evidence were perfect or complete, there would be no uncertainty. Schum best sums up the probabilistic nature of evidence by noting that,

> *"One attribute of human inference common across different situations is that conclusive evidence is either in very short supply or is quite impossible to obtain* [144]*."*

Reality provides masses of inconclusive evidence that are incomplete on matters relevant to our hypotheses, and arrive from sources of questionable credibility. In fact, all sources of evidence must be examined for credibility, including ourselves.

Inferred conclusions are necessarily probabilistic due to the existence of incomplete or uncertain evidence. Many problems of interest do not have a single, correct answer. In other cases, evidence may be incomplete or inconclusive, making determination of a certain conclusion impossible. Either of the above situations results in inferences from available evidence being necessarily uncertain. In real world applications, uncertainty occurs in many forms and is present for a variety of reasons. Continuing decomposition of a problem into sub-problems may allow refinement of evidence, but may also introduce further uncertainty. Restricted by time or budget, an analyst may choose an arbitrary termination of decomposition, which may incorrectly bias the conclusion. Finally, the participation of multiple individuals in forming a solution to a problem may introduce alternative perspectives on types of reasoning, judgments, roles, and biases that all define different solutions. This inconclusiveness further adds to uncertainty.

Schum identifies three primary probabilistic systems used to define and combine inferential force or weight of evidence: Bayesian, Belief Function, and Baconian [144]. Common to each is the idea of relevance. Schum defines relevant evidence is that which allows us to change our beliefs about the likeliness of hypotheses or possibilities of interest. It therefore has some force in probabilistic belief revision and should be graded in probabilistic terms. Further, he describes relevant evidence as "vector-like" in that it is applied in a certain direction toward one or more hypotheses.

Evidence arrives in two forms, tangible and testimonial. Tangible evidence is that which may be examined directly to determine what it reveals. It may be in the form of documents, photographs, models, etc. On the other hand, testimonial evidence is a report or assertion coming from another person. The credibility of each type of evidence is evaluated by examining different credentials. According to the taxonomy of evidence proposed by Schum, tangible evidence has credibility likelihood defined by the authenticity and accuracy of the item. Specifically, authentic evidence is that which it is defined to be. Authenticity in technological data is negatively affected by anything that produces a mis-characterization. For example, radar jamming introduces false returns representing nonexistent contacts. In the legal domain, authenticity of evidence is preserved through the chain of custody. Accuracy represents the uncertainty of the evidence related to truth values, and can be measured through a true positive rate and false alarm rate for the reporting source. Humans provide testimonial evidence, which requires a credibility estimation of both the witness and the person to whom the testimony is given. Credibility of testimonial evidence is measured by veracity, objectivity, and observational sensitivity of the witness. Veracity represents the perceived truthfulness of the witness represented by the likelihood that the source believes what he reports. This is based on ancillary evidence of prior testimony and background related to the source. Objectivity of a human source is determined by the source's ability to provide evidence that is observed, rather than driven by his personal bias, motivation, or expectation. Finally, sensitivity is given by the accuracy and physical ability of the source to collect the evidence.

2.4 Knowledge Engineering and Representation

Knowledge engineering is a computer science discipline that involves collecting and incorporating knowledge into computer systems in order to solve complex problems. Such computer systems are known as knowledge-based systems and perform inferential reasoning on their knowledge bases using the first-order logic of graphical probability models and Expressive Probabilistic Languages (EPL).

2.4.1 First-order Logic

Philosophically, logic is the study of correct reasoning [149]. First-order logic is a system for formalized knowledge representation and reasoning that has a very long history, possesses strong mathematical foundations, and has proven to be expressive enough to serve as the foundation for a wide variety of knowledge representation languages and extended logics. It affords the ability to represent entities of different types interacting with each other in various ways and provides the theoretical foundation for the type-systems used in object-oriented and relational languages [29]. Also, formal ontologies are usually expressed in languages based on FOL or its subsets. FOL allows reasoning about properties that are shared by many objects through the use of variables, and is distinguished from propositional logic by the use of quantified variables, as shown below.

A FOL theory is a set of first-order logic sentences over a specified domain of discourse. FOL uses variable and quantifiers to allow reasoning about properties shared among objects in the domain. For example, if Dog(x) means that x is a dog, and Mammal(x) means that x is a mammal, then the FOL sentence

$$\forall_x (Dog(x) \to Mammal(x))$$

means that for every x, if x is a dog, then x is also a mammal. A sentence is satisfiable if there exists at least one possible world in which it is true. An interpretation, also called a possible world, is a set of objects, functions defined on the objects, and relations that hold between objects [37][45]. Sentences are stored in a FOL Knowledge Base which may be accessed by a knowledge-based system to perform inference in response to a query.

Despite its expressiveness, pure FOL has limited applicability to practical ST problems [45]. In many if not most contexts it is difficult to produce theories that are consistent, consist of true sentences, and contain enough content for useful reasoning. While first-order theory implies truth-values for the valid sentences, it provides no means to evaluate the plausibility of other sentences [96]. Although first-order logic is sufficient for formalizing much of mathematics, and is commonly used in computer science, its expressiveness is insufficient for application to plausible reasoning.

2.4.2 Uncertainty

Costa argues that uncertainty is predominant throughout the real world and its presence necessitates incorporation of probability in ontology development [36]. In the ST domain, uncertainty characterizes incomplete or erroneous evidence which leads to beliefs that fall short of knowledge and the formation of fallible conclusions. Korb and Nicholson recognize three forms of uncertainty plaguing intelligent systems operating in a real-world context [89]:

- Ignorance: limits of knowledge lead to uncertainty;
- Physical randomness or indeterminism: probability of states occurring randomly;
- Vagueness: classification into a specific category is not deterministic.

The pervasiveness of uncertainty in real-world problems calls for the incorporation of probability into ontological engineering. For example, in naval applications some radar systems are used for both air search and for surface search. At any given time an observer may be ignorant of which purpose is being served. A given kind of sensor may be characterized by random measurement error, which gives rise to characteristic rates of false positive and false negative reports. Similarly, a small surface combatant may be of type destroyer, frigate, corvette, etc.; its classification is vague. Historically, both the size and primary mission of a surface combatant determined its type, but today those lines are less clear. All these kinds of uncertainty are important to characterize in ontologies to support evidential reasoning about the domain.

There are numerous usages of the term probability, but we are primarily concerned with the Pascalian interpretation, also known as mathematical probability. Interpretations of Pascalian probability include frequentist, measured, subjective, evidential, etc [144][105]. The defining characteristic of Pascalian probability is compliance with the three Kolmogorov Axioms [144]; it includes classical statistical theory based on analysis of historical data or of relative measurement within a finite context. The two most commonly discussed interpretations of Pascalian probability are the subjective interpretation, in which probabilities represent rational degrees of belief, and the frequency interpretation, in which probability represents a limiting long-run frequency of random sequences. The Pascalian system is of particular interest because it is often the case that historical data is not available for the context of the current problem and that expert elicitation of probabilities is required.

2.4.3 Knowledge-Based Model Construction

Knowledge-Based Model Construction (KBMC) algorithms construct a ground model from an expressive representation and problem-specific evidence. A ground model is generated by instantiating the first-order representation with evidence relevant to a specific query. KBMC has a long history beginning with Horsch and Poole in 1990 [72] and is now identified as the most prominent family of models in the field of first-order probabilistic inference [13]. Each KBMC approach generates a propositional graphical model from a first-order language specification to answer a query in the domain of interest [13][45]. Approaches introduced by Horsch and Poole [72], Breese [15], Goldman and Cherniak [54], Poole [131], Glesner and Koller [53], Koller and Pfeffer [88], Jaeger [81], and Haddaway [64] all produce Bayesian networks. More recently the KBMC family has been extended to include Probabilistic Relational Models (PRM) [52], Relational Dependency Networks (RDN) [114],Relational Markov Networks (RMN) [156], Bayesian Logic Programs (BLP) [84], Logical Bayesian Networks (LBN) [48], Constraint Logic Programming (CLP) [139], Markov Logic Networks (MLN) [45], and Multi-Entity Bayesian Networks (MEBN) [96]. A ground model constructed by a KBMC algorithm is solved using a standard inference algorithm. The following sections describe BNs, PRMs, MLNs and MEBNs in further detail.

2.4.4 Graphical Probability Models

Graphical probability models provide a powerful language for compactly specifying probability distributions (joint) over many interrelated random variables. A graphical model is defined by a graph representing the dependency structure and a set of local variables that specify numerical probabilities. Each node in the graph represents a random variable, and has a mutually exclusive and exhaustive set of states. Each arc represents a direct dependency between two random variables. A local distribution represents numerical probability information. Together, the graph and local distributions define a joint probability distribution over the random variables represented by the nodes in the graph. A joint probability distribution is a probability distribution defined on the Cartesian product of the state spaces of two or more random variables. Within the domain of graphical models exist directed acyclic graphical models and undirected graphical models, most commonly represented by Bayesian networks (BN) and Markov networks (MN), respectively.

2.4.4.1 Knowledge Engineering with Bayesian Networks

Drawing from software engineering, Korb and Nicholson propose an iterative and incremental methodology for the development and deployment of Bayesian networks. The methodology is called Knowledge Engineering with Bayesian Networks (KEBN). A Bayesian network is a formal language for representing knowledge about uncertainty defined by a directed acyclic graph and a set of local distributions. Each node represents a random variable, or uncertain quantity, which can take on two or more values [129]. The structure of the graph captures relationships between nodes, and these relationships are quantified by conditional probabilities associated with a node and its parents. Conditional probability is the probability of an event, given the existence of some evidence. Bayesian probabilistic inference is performed by computing the posterior probability distribution of a set of network query nodes given new values for evidence (conditioning) nodes. A BN for a given domain consists of a pre-specified set of variables and relationships. It is this limitation that makes BN unsuitable for complex domains that require representation of a varying number of entities. Still, BNs are widely used throughout academia and industry to provide inferential reasoning support for static problems in applicable domains.

Korb and Nicholson introduce six primary tasks for KEBN, all of which are of heavily dependent on expert elicitation [89].

i. Identify the variables and their states

ii. Identify the graph structure

iii. Identify the parameters (probabilities)

iv. Identify the available actions/decisions and their impacts

v. Identify utility nodes and their dependencies

vi. Identify the preferences (utilities)

Parameters and structures can also be specified through learning from data. Pearl defines learning as, "...the process of acquiring and effective internal representation for the persistent constraints in the world...as well as assembling the computational facilities by which predictions and explanations are produced [129]." Learning of Bayesian networks is typically divided into two subtasks: structure learning and parameter learning. A discussion of learning from data in the context of probabilistic ontology development is included in Section 3.2.2.5.

The preferred method for construction of complex BNs is to perform a spiral development cycle using prototypes for project planning and testing of code sections [89][99]. Prototyping allows functional implementation of parts of the final system with real input and output. Boehm's spiral model provides an iterative process of requirements analysis, design, implementation, and testing to produce an increasingly functional design. Section 4.1 describes the use of the Spiral Development Cycle model in probabilistic ontology development.

2.4.5 Expressive Probabilistic Languages

Expressive Probabilistic Languages merge probabilistic representations with traditional logic formalisms to support inference over large and complex domains. EPLs extend the expressive power of probabilistic modeling languages to support representation of real-world problems in terms of objects and relationships, and are therefore applicable to probabilistic ontology engineering. EPLs continue to evolve; this section describes three of the more commonly recognized languages which continue advancement in the ST community.

2.4.5.1 Probabilistic Relational Models

Probabilistic Relational Models [13][52][45][96] are used to represent uncertainty in relational data by combining a Frame System logical structure with probabilistic representation based on directed graphical models. A Frame System is a relational schema consisting of objects (classes with descriptive attributes), and relationships (reference slots) among objects. This structure maps directly to relational databases, with each class describing a table and its attributes describing the columns. Reference slots correspond to attributes in related tables. The relational skeleton provides a template for instantiation of the relational schema by specifying the objects of each class and their relationships, but not the attribute values. A PRM specifies a distribution over instantiations of this relational skeleton, and is comprised of a qualitative dependency structure and associated quantitative parameter data. The dependency structure is defined by the parents of each attribute, and is represented by a directed graphical model (BN). For a given graphical model, a conditional probability table (CPT) is specified for each attribute which specifies the probability of an attribute state, given its parents.

PRMs extend BNs with the concept of objects, their properties, and relations between them to provide a natural way to represent uncertainty about values of unary functions and relations [96]. Queries to PRMs are computed by performing exact or approximate inference on the ground Bayesian network, which is defined by instantiation of a PRM for a particular skeleton. A PRM is similar to a BN in that each node has a set of directly influencing parents and a local probability distribution that specifies the dependence on these parents. It is different from BN in that dependency is defined at the class level, which allows the dependency relationship to be used for any object in the class. Still, representation uncertainty about n-ary functions and relations is complicated in PRMs, and simplified in logic-based languages, as discussed below.

2.4.5.2 Markov Logic Networks

Markov Logic Networks [13][45], are proposed as a unifying framework to facilitate transfer of knowledge across tasks and approaches, to compare approaches, and to help bring structure to the field of statistical relational learning [45]. The basis for a MLN is a Markov network, an undirected, possibly cyclic, graph and a set of potential functions (cliques). The potential function is a non-negative, real-valued function of the state of the clique. The structure of a MLN includes a set of FOL sentences and an associated weight with each sentence which together define a probability distribution over possible worlds. Markov Logic is syntactically indistinguishable from FOL except that each sentence has a weight attached. A set of MLN sentences represents a probability distribution over possible worlds, where a possible world is a set of objects, functions, and relations that hold between those objects. In FOL, worlds that violate axioms have zero probability. However, in MLN, the probability of existence for a world is reduced monotonically by the number of axioms it violates. Probabilities on possible worlds are represented by a set of weighted first-order sentences.

A significant strength of MLNs is their ability to subsume other FOL languages due to their expressivity [13]. A first-order KB can be transformed into a MLN by assigning a weight to each formula. Drawbacks to the flexibility of MLNs include slow inference due to large underlying networks, difficulty of learning for undirected graphs, and deterministic relationships are not naturally modeled, requiring infinite weights. MLNs are particularly suited to the tasks of collective classification, link prediction, link-based clustering, social network modeling, and object identification [45].

2.4.5.3 Multi-Entity Bayesian Networks

The Multi-Entity Bayesian Network [96][100][97] language provides a means to express complex graphical models with repeated structure by integrating FOL with Bayesian probability. Probabilistic knowledge is captured in parameterized BN fragments called MEBN fragments (MFrags) which are organized into a set that satisfies consistency requirements, called a MEBN Theory (MTheory). Each MFrag represents probability information about a collection of related random variables, and the MTheory ensures existence of a unique probability distribution over its random variables. MEBN extends BNs to provide first-order expressive power, and extends FOL to provide a means to specify probability distributions over interpretations of first-order theories [96].

In MEBN, a FOL sentence is represented as a random variable in one or more MFrags which when combined into an MTheory implicitly express a joint probability distribution over the truth values of the FOL sentences. MEBN represents uncertainty about the values of n-ary functions and relations by specifying distributions over conceptually meaningful clusters of related hypotheses within the MFrag construct. Occurrences of a given argument are bound to the same entity when the MFrag is instantiated because random variables are defined at the MFrag level. This allows flexibility unavailable in other representations (PRMs, OOBNs) by allowing hypotheses in an MFrag to refer to attributes and relations of different entities.

2.4.6 Inference

Korb and Nicholson define inference as, "Conditioning a variable upon the arrival of new information [89]." More simply, inference in a probabilistic system means computing the posterior probability distribution for the model after new evidence is received. There are two major categories of inference algorithms: exact and approximate. Selection of the appropriate algorithm is a function of the size and complexity of the probabilistic structure. Exact inference computes the actual posterior distribution at the cost of greater computational cost, while approximate inference saves computation by computing an approximation. Approximate inference reduces the calculation requirement of the original probabilistic representation by one of many algorithms including arc reversal, clustering, cutest conditioning, junction tree, likelihood weighting, logic sampling, loopy propagation, variable elimination, etc. [89]. Currently, most inference algorithms are specified at the propositional level, which means that the algorithms are applied to a model generated by instantiation of the first-order representation (a grounded model). On the other hand, lifted inference algorithms are applied directly to the first-order representation. Exact or approximate inference is possible in either specification.

Propositional inference (propositionalization) algorithms are performed on instantiations of first-order models relative to the current query, not the explicit first-order structure. The instantiation of the first-order model is called the grounded model. Exact propositionalized inference performed on Probabilistic Relational Models and Multi-Entity Bayesian Networks is accomplished by solving the exact Bayesian network represented by the grounded model. Exact inference may become intractable for large or complex networks. Larger BNs may be solved through inference algorithms that employ computational reuse and exploit a model's class hierarchy. Exact propositionalized inference on a Markov Logic Network is performed by using Bayes theorem to solve a conditional probability defined by instantiating the MLN over all possible worlds where the first-order sentences hold. Domingos and Richardson propose an algorithm for establishing this conditional probability in [45]. However its computation will be intractable in all but the smallest domains[45]. An approximate inference algorithm for MLNs is proposed by Shavlik and Natarajan, that reduces the size of the grounded network by preprocessing to speed up inference [150]. Many standard inference algorithms are available for both exact and approximate inference. Getoor et al. endorse Belief Passing (BP), which guarantees convergence to correct marginal probabilities for singly connected BNs and often converge for general networks [52]. Domingos and Richardson favor Markov Chain Monte Carlo (MCMC) Gibbs sampling for Markov Logic Networks, but state that any approximate inference algorithm provides a viable alternative [45]. For Multi-Entity Bayesian Networks, Laskey provides an algorithm that produces a sequence of approximate Situation Specific Bayesian Networks to perform inference. If the SSBN terminates without a stopping criterion specified, the results are an exact query response or the findings are inconsistent. If the algorithms does not terminate, the SSBNs converge to a correct query response if it exists [96].

Lifted inference means that the algorithm is executed directly on the first-order representation. Instead of performing an inference algorithm on a grounded model, lifted inference algorithms operate on the representation at the first-order level by eliminating variables that are irrelevant to the current query. Efficiency is achieved over propositional inference by eliminating groups of random variables. Poole proposed the first exact lifted inference algorithm in 2003, which combined variable elimination with resolution. Braz et al. extended Poole's work with additional elimination techniques by the first-order variable elimination (FOVE) algorithm [14] which removes multiple variables at each step. It was later extended by Milch et al to produce C-FOVE [109], and Sen et al. to produce the rv-elim graph[148]. Approximate lifted inference is performed by either deterministic message passing algorithms, sampling-based methods, or interval-based methods.

2.5 Case Study Development

Yin [165] provides a comprehensive overview of a case study methodology from design of the case study to analysis and presentation of results. A case study is used to explain, describe, illuminate or enlighten a contemporary topic within its environmental context. It is especially useful to explain presumed causal links in real-world relationships that are too complex for surveys or experimental strategies. The six steps of case study development identified by Yin, (plan, design, prepare, collect, analyze, share) provide a clear methodology to properly apply research techniques and avoid common pitfalls of case study research [165].

2.5.1 Plan

When planning research, it is first necessary to identify the methodology that will be employed to arrive at a solution. Yin identifies three conditions that determine type of research that should be utilized [165]

- Type of research question posed.
- Extent of control an investigator has over actual behavioral events.
- Degree of focus on contemporary as opposed to historical events.

If the investigator has little control over the sequence of events in research involving "how" or "why" questions about a contemporary event within a real-world context, then a case study is the preferred method [165]. It is also important to note that unlike statistical analyses, a case study does not represent a sample, and therefore may not enumerate frequency data for a population. Instead it may be generalized to theoretical propositions.

2.5.2 Design

The research design is a logical plan to ensure the study questions are properly addressed and that the appropriate data are collected. Yin identifies five components of the design: study questions, propositions, units of analysis, logic linking the data to the propositions, and the criteria for interpreting the findings [165]. Quality of the design is measured through four critical conditions, described as:

- Construct Validity – Construct validity identifies the correct operational measures for the concepts being studied.
- Internal Validity – Internal validity seeks to establish a causal relationship (used for exploratory or causal case studies only).
- External Validity – External validity defines the domain to which a study's findings can be generalized.
- Reliability – Reliability demonstrates that operations of the study are repeatable with the same results.

An appropriate design will include the overall theory behind the case study, which is also the level of generalization for the results. The final analysis is an evaluation against the propositions supporting the theory.

2.5.3 Prepare

The preparation task sets the stage for a successful case study by ensuring all participants are appropriately trained, that the data collection protocol is developed and approved, that candidates are screened, and that a pilot study is conducted. Yin notes that all preparation will be negated if a biased investigator uses a case study to substantiate his preconceived outcome [165]. Proper training standardizes data collection and minimized bias.

2.5.4 Collect

Yin identifies three overriding principles that enable a successful case study data collection, the use of [165]:

- Multiple sources of evidence;
- A case study database;
- A chain of evidence.

The six sources of evidence available for case studies include: documents, archival records, interviews, direct observation, participant-observation, and physical artifacts. By using multiple sources of evidence, the investigator may produce "converging lines of inquiry" that corroborate findings over the propositions [165]. A database of notes, documents, and narratives provides a formal presentation of evidence that strengthens the reliability of the case study. Similarly, the chain of evidence ensures integrity of the database and its contents.

2.5.5 Analyze

Yin suggests five techniques of data analysis to examine, categorize, tabulate and test evidence to draw empirical conclusions: pattern matching, explanation building, time-series analysis, logic models, and cross-case synthesis [165]. These techniques should be employed to evaluate the evidence against the theoretical propositions derived from the original case study objective. When applying these techniques, there are four underlying principles that an analyst should follow: show that all evidence was attended to, address all major rival explanations, address the most significant aspect of the case study, and use prior, expert knowledge [165].

2.5.6 Share

With the analysis complete, the case study is ready to share with interested parties and stakeholders. To produce a cogent report at the correct level of abstraction and fidelity, Yin provides three tasks [165]:

1. Identify the audience for the report;
2. Develop its compositional structure;
3. Have drafts reviewed by others.

Because of the natural language descriptions used for real-world events, a case study is an excellent communication tool for expressing information about complex events.

Yin's process of case study planning, design, preparation, collection, analysis, and sharing was used in this work to ensure a valid research design and a common set of ground rules for the investigation team. The sources of data may be vast, and collection techniques must be tailored to both the data and the overall analysis to produce a result consistent with the study objectives. Analyzing data and reporting results must explore rival hypotheses and produce an understandable output that accurately reflects empirical observations.

CHAPTER THREE: A REFERENCE ARCHITECTURE FOR PROBABILISTIC ONTOLOGY DEVELOPMENT

3.1 Introduction

3.1.1 Utility and Flexibility of a Reference Architecture

The Reference Architecture for Probabilistic Ontology Development (RAPOD) presents a compilation of components required for probabilistic ontology development and therefore facilitates design, implementation, and support processes without rigid adherence to a particular set of tools. The Department of Defense (DOD) defines a Reference Architecture as:

> *"... an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions* [122].*"*

Common throughout the literature on reference architectures is the idea of serving as a blueprint for architects to develop specific solution architectures within a defined domain [122][93]. As the blueprint, it serves as a template for software development, defining integral components and their relationships, thereby reducing development time and project risk. Further, it standardizes language among participants, provides consistency of development within the domain, provides a reference for evaluation, and establishes specifications and patterns [122].

3.1.2 Background

Development of the RAPOD provides synergy of effort within the ST community by identifying concepts, processes, languages, theories and tools for designing and maintaining probabilistic ontologies. Presently, ontological engineering facilitates the development of explicit, logical and defensible ontologies for knowledge-sharing and reuse. A similar pragmatics in the form of the Probabilistic Ontology Development Methodology has been produced for probabilistic ontologies and is described in Chapter 4 of this dissertation. The RAPOD facilitates synergy of effort between multiple disciplines including probabilists, logicians, decision analysts and computer scientists. It describes each of the components required for a functional probabilistic ontology and their interrelationships, and defines the criteria to be satisfied by any set of selected tools and methods using a Unified Process-inspired methodology.

3.1.3 Scope

The RAPOD spans the knowledge, processes, models, and tools necessary for engineering probabilistic ontologies at a high level of abstraction. Through decomposition or aggregation of existing methodologies, it provides universal techniques and a generalized framework for the fundamental components needed to construct probabilistic ontologies from conceptualization to operation through multiple tasks, including:

- Model conceptualization and framing
- Ontology development through elicitation and ontological learning
- Probability incorporation through iterative decomposition

There are many participants involved in realizing an operational probabilistic ontology. Specifically, three contributors collaborate frequently to complete this architecture:

- Stakeholder Decision Maker (DM). The eventual owner/operator of the probabilistic ontology or his representative, the Stakeholder DM provides high-level guidance as well as specific system requirements. Ultimately the operational PO must satisfy this individual.
- Subject-Matter Expert (SME)/Analyst. One or more SMEs provide domain expertise that serve as input regarding the classes and relationships of the ontology and offer insight into the proper representation of uncertainty in the PO through elicitation techniques and/or probabilistic learning.
- Probabilistic Ontology Developer. The developer works closely with the Stakeholder and SMEs to solve the Stakeholder's specific problem. If a PO is deemed to be the proper solution, he must take this model from conceptualization as an objective to an operational model supporting the Stakeholder DM.

Within the scope of the RAPOD, these individuals coordinate to instantiate a collection of concepts and tools for development and implementation from existing and proposed ontological and probabilistic ontological engineering methodologies, providing a single collection of knowledge to solve a domain-specific problem. Their solution is defined as a domain-specific architecture that may be reused for comparable problems in similar domain contexts. Section 3.3 introduces an architecture for the Military Ship

Probabilistic Ontology as an example. This architecture is applicable to other Decision Support System (DSS) problems involving inferential reasoning in the maritime domain.

3.1.4 Model Implementation and Viewpoint

The concept behind the RAPOD is to establish intellectual control of the PO model, stimulate reuse, and provide a basis for development through instantiation of a particular set of tools the developer will utilize to design and implement complex probabilistic ontologies for a particular domain [95]. Intellectual control establishes common semantics and allows consistent integration of new system components by anticipating their inclusion from design. Reuse is a prime tenet of ontological engineering and is enabled through identification of common components and relationships. Further, a well-defined and properly architected PO may be reused entirely through spiral modification to incorporate additional knowledge or relationships. Subsequent spirals advance the capability of the PO by incorporating additional Prime Queries or relationships, as described in Chapter 4. Most importantly, the architecture serves as a blueprint for the PO Developer and a clear mechanism between him and the Stakeholder Decision Maker. The architecture allows individuals, teams, and organizations to communicate objectives, requirements, constraints, components and relationships with a common vocabulary and understanding of the objective. Ontological engineering, and *probabilistic* ontological development, may be completed by several different methodologies depending on the context and domain of the problem. Therefore, the RAPOD provides ready access to tools, techniques, and procedures that have proven successful in the past. The RAPOD also exposes synergies in algorithms, heuristics and model use between ontological and probabilistic ontological engineering. Through careful selection of tools with common parameters, the final model is more intuitive. The viewpoint of this reference architecture is that of the Probabilistic Ontology Developer in support of a Stakeholder Decision Maker desiring decision support for a defined area of interest.

3.2 Reference Architecture for Probabilistic Ontology Development

The Reference Architecture for Probabilistic Ontology Development facilitates PO development and reuse by providing a template from which multiple PO solutions to similar problems may be constructed. The output of the RAPOD is a domain and problem-type specific architecture that may be used to develop POs for similar problems. Reusable architectures provide a shortcut to future development by identifying inputs, methodologies, and support artifacts that have previously produced successful solutions within the domain.

In each of its three layers, the RAPOD identifies components necessary for the construction of a probabilistic ontology without specification to particular tools. Working with the stakeholders, the PO Developer selects individual component solutions that suit the problem-type and domain. Specification of a set of tools for each component instantiates an architecture that is used to develop the PO. Figure 1 provides an overview of the RAPOD, discussed in detail below.

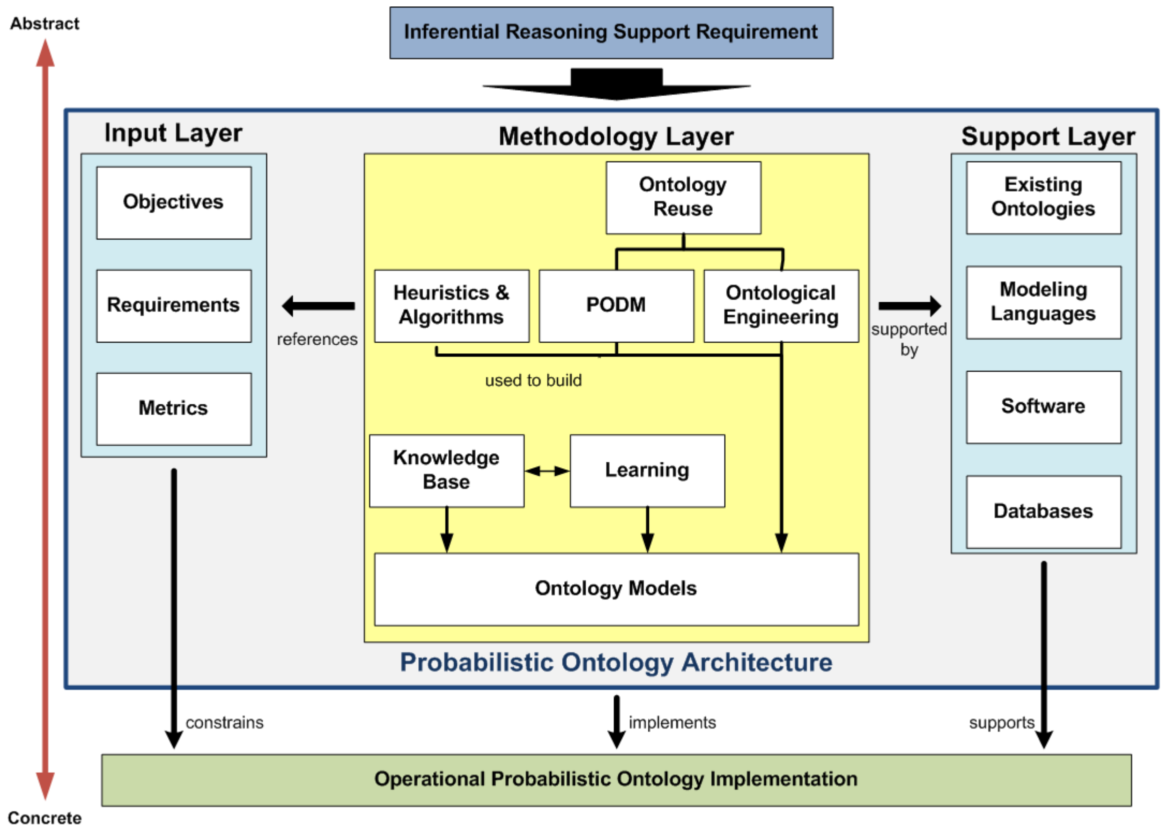


Figure 1 - Reference Architecture for Probabilistic Ontology Development

The Reference Architecture for Probabilistic Ontology Development shown in Figure 1 illustrates the scope of the reference architecture from abstract to concrete. At the top of the illustration is the most abstract conceptualization defined as a problem or objective by the Stakeholder Decision Maker that requires implementation of a probabilistic ontology. The base of the illustration represents the operational implementation of the probabilistic ontology to provide inferential reasoning support. Between lies the probabilistic ontology architecture, which translates the conceptualization into a blueprint for development. The probabilistic ontology architecture is comprised of three interacting layers, which group and characterize similar functionality: the Input Layer, Methodology Layer, and Support Layer. These and their relationships are described in the following subsections.

3.2.1 Input Layer

The Input Layer defines external influences on the probabilistic ontology and is referenced by components of the Methodology Layer. It contains those components expected to provide detail on the purpose of the PO and its bounding constraints in the form of system requirements. Population of the Input Layer occurs primarily during the early stages of the development process during which the Stakeholder Decision Maker and PO Developer work closely to identify the objective of the model, expectations of its performance, and resource restrictions. Parameters specified in the Input Layer will constrain the operational implementation.

3.2.1.1 Objectives

The objectives hierarchy contains a representation of performance, cost and schedule attributes that determine the value of the system, with an over-arching Objective Statement that captures its primary intent [17]. Objectives state the overall intent of the project in short, clear, descriptive phrases. They are defined by the Stakeholder DM to bound the scope of the final product and set expectations. These are often described in the following form [4]:

To Action + Object + Qualifying phrase

For a probabilistic ontology model, applicable categories of objectives may include: performance, reliability, compatibility, adaptability, and flexibility. Further descriptions of these and other categories may be found in Armstrong [4]. Choosing the correct objectives ensures that the desired problem is solved and that the PO Developer and Decision Maker have clearly communicated. The entire project is best focused through a Top-level Objective Statement.

3.2.1.2 Requirements

Requirements define the system to be implemented in terms of its behaviors, applications, constraints, properties, and attributes. The systems engineering literature on requirements elicitation and development is rich, but there is consensus that no single methodology exists for requirements engineering [90][57]. In general, requirements elicitation approaches may be categorized as structured or unstructured [57] using a combination of strategies depending on the scope of the system under development and the participation commitment of the Stakeholder Decision Maker.

Requirements are elicited from the Stakeholder Decision Maker and SMEs through an iterative process that generally includes objective setting, background knowledge acquisition, knowledge organization, and requirements collection as introduced by Kotonya and Sommerville [90]. Grady categorizes three strategies for requirements analysis: structured analysis, cloning, and freestyle [57]. Using one or more of these strategies and concentrating on the four tasks above will lead to identification of appropriate requirements to satisfy valid model development.

3.2.1.2.1 Structured

As its name implies, the structured analysis strategy introduced by Grady falls into this category and is further partitioned into Top-down, Bottom-up, and Middle-out methodologies [57].

- Top-down. Customer need is decomposed from the abstract to concrete, and from the system level to component and part level.
- Bottom-up. Individual components are identified and then integrated to produce a system, or system of systems. This method is useful when there is a constraint that requires use of existing components to produce a new system.
- Middle-out. Existing or identified components are recognized and then the system is abstracted from their integration, and decomposed into their parts. This method is best used for integrating multiple existing systems into a system of systems.

3.2.1.2.2 Unstructured

Cloning and freestyle strategy techniques fall into the Unstructured category approach. The PO Developer acts independently to produce requirements in an ad hoc manner, then consolidates them into a formal requirements document.

- Cloning. An existing library is reused to produce the requirements from scratch and specifications for a system or update.
- Freestyle. The experienced Developer creates the requirements based on the Stakeholder Decision Maker’s objectives.

There is inefficiency and risk involved in the unstructured methods as there is nothing to prevent duplicative work, incompleteness, conflicts and misdirection.

3.2.1.3 Metrics

Metrics are used to describe parameters, Measures of Performance (MOP) and Measures of Effectiveness (MOE) that characterize the criteria against which the fielded system is to be evaluated. Green defines a hierarchy of effectiveness measures that follows the system of systems concept, and is shown in Figure 2, below [58].

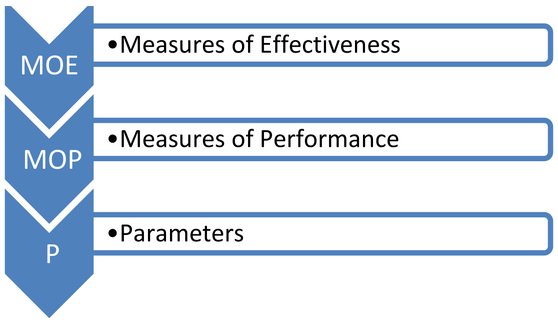


Figure 2 - Hierarchy of Effectiveness Measures

The following definitions are adapted from those offered by Green to accommodate the PO development process:

- Measures of Effectiveness. A measure of system performance within its intended environment (e.g. overall system effectiveness).
- Measures of Performance. A measure of one attribute of system behavior derived from its parameters (e.g. probability of correct identification).

- Parameters. Properties or characteristics whose values determine system behavior (e.g. error rate).

Armstrong [4] opines that useful metrics take quantifiable form with both a clear definition of the measure and its associated units. They must also be mission-oriented, discriminatory, sensitive, and inclusive [58]. In all cases, appropriate metrics depend on the system under development and its ultimate purpose (objectives).

3.2.2 Methodology Layer

The Methodology Layer contains the heart of the probabilistic ontology development process including the Probabilistic Ontology Development Methodology that allows creation of a specific probabilistic ontology implementation to support the requirements of a Stakeholder Decision Maker. The Methodology Layer references information gathered in the Input Layer and is assembled using components and tools from the Support Layer. Its individual components are introduced below.

3.2.2.1 Probabilistic Ontology Development Methodology

The Probabilistic Ontology Development Methodology provides specific activities and tasks that evolve Stakeholder Decision Maker requirements into an ontology that is probabilistically-integrated, a probabilistic ontology. The activities of the Probabilistic Ontology Development Methodology are introduced in the below activity diagram (Figure 3) and further detailed in Chapter 4. These activities fit well within both Waterfall and Spiral Development Life Cycle processes where in Spiral Development iteration is explicitly anticipated.

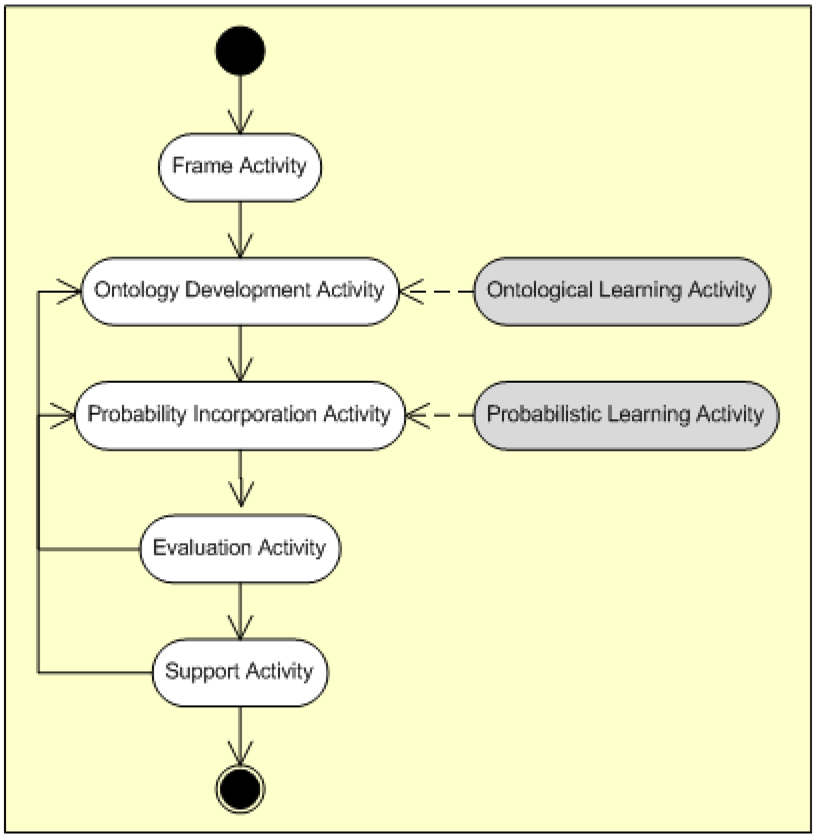

Frame Activity
Ontology Development Activity
Ontological Learning Activity
Probability Incorporation Activity
Probabilistic Learning Activity
Evaluation Activity
Support Activity

Figure 3 - PODM Activity Diagram

Completion of the PODM activities and tasks establishes a framed solution to a specific inferential reasoning problem grounded in an inclusive ontology representing its entities and incorporating probability to represent uncertainty.

3.2.2.1.1 Frame Activity

The Frame Activity bounds the problem by the requirements and prepares the developer for successful creation of the probabilistic ontology solution. The activity culminates with an initial hierarchy of the entities that represent the Core Model that will later be extended to the full probabilistic ontology. The hierarchy may be represented as a class diagram or other suitable artifact, and will be used to create a first-order logical model in the Probability Incorporation Activity. Tasks of the Frame Activity include:

i. Define the Spiral
ii. Define Requirements
iii. Define Metrics
iv. Identify Tier-one Attributes
v. Draft Initial Class Diagram

During this activity, one or more Prime Queries are established to satisfy the inferential reasoning support required of the system by the Stakeholder Decision Maker. These Prime Queries and their associated Tier-one Attributes define the Core Model extended in the PO Construction Cycle to create the full probabilistic ontology model. Details of the Prime Queries and Tier-one Attributes can be found in Chapter 4.

3.2.2.1.2 Ontology Development Activity

The Ontology Development Activity summarizes the ontological engineering process required to produce a working ontology. The tasks described in Chapter 4 are adapted from the METHONTOLOGY ontology development process as described by Gomez-Perez et al. [56]. However, each ontology design requires a unique series of development events that are tailored to the specific domain to be modeled. Integration of similar tasks and the addition of tasks emphasizing ontology reuse expand the basic process to make use of ever-extending online ontology resources. The ontology engineering steps of the Ontology Development Activity method are:

i. Conduct Ontological Engineering
ii. Research Reusable Ontologies
iii. Research Heuristics and Algorithms
iv. Implement Ontology Model
v. Conduct Ontological Learning

The Ontology Development Activity culminates with a working ontology that will be extended with first-order probability relationships to represent uncertainty in the evidence presented to the inferential reasoning model.

3.2.2.1.3 Probability Incorporation Activity

The Probability Incorporation Activity is the heart of the methodology and the definitive component of the PODM. It begins with creation of a central probabilistic model focused on the Prime Queries and their Tier-one Attributes. Together, these form the Core Model. The initial Spiral Core Model is the keystone of the complete probabilistic ontology model and is evaluated for correct operation and logic before it is expanded to include additional attributes using the iterative Probabilistic Ontology Construction Process. After Core Model development, the PO Construction Process systematically decomposes each of the primary, secondary and tertiary attributes, evaluating the model logic at each step. For the Spiral Development Cycle process, subsequent spirals of the PODM incorporate additional Prime Queries. The sequential tasks of the Probability Incorporation Activity are:

i. Establish the Core Model

ii. Iterate the Probabilistic Ontology Construction Process

The specific details of these procedures are described in Chapter 4. After the PO for the initial Prime Queries is completely represented, additional Prime Queries and their Tier-one attributes may be modeled by spiraling the PODM in a similar fashion. Details on this process are discussed in Chapter 4. The Probability Incorporation Activity results in a probabilistic ontology, ready for incorporation of evidence, evaluation, and eventual implementation as the inferential reasoning system solution.

3.2.2.1.4 Evaluation Activity

The Evaluation Activity completes the PODM by performing a series of evaluation tasks. Laskey and Mahoney [99] describe three types of evaluation for probabilistic models: elicitation review, importance analysis, and case-based evaluation. Elicitation review involves expert review of node definitions, state definitions, independence assumptions, and probability distributions to develop a high-level view of the overall model to evaluate consistency, correctness and adequacy as a representation for the domain. For the PODM, this task is performed by SMEs and the PO Developer. The importance analysis measures importance of a particular variable as it relates to an evidence variable. Case-based evaluation involves developing a series of case studies to test the system in realistic scenarios. Case studies are developed to test the spectrum of inference tasks expected to be encountered during the Operation Phase of the Software Development Life Cycle. Each case is run, and results are evaluated against existing models where available, or by SMEs and the PO Developer. Models that demonstrate erroneous behavior are corrected through one or more iterations of the Construction Process. In the PODM, importance analysis and case-based analysis are combined to evaluate variables of the PO model in realistic operational scenarios. The tasks in the Evaluation Activity are:

i. Conduct Elicitation Review
ii. Draft Case Studies
iii. Populate Evidence Variables
iv. Run Probabilistic Ontology Model and Evaluate Results
v. Correct Model, as required, through Construction Process

Case study models are run independently. Once a model performs as expected, it is documented and the next case study is applied. The evaluated cases should cover the breadth of operations expected to be encountered and include typical, non-typical, and unusual conditions [99]. Additional detail on the Evaluation Activity and its component tasks is given in Chapter 4.

3.2.2.2 Ontological Engineering

In Gomez-Perez et al, ontological engineering is defined as the activities that concern the ontology development process, life cycle, construction methodologies and tools [56]. While traditional ontological engineering methods ensure that ontologies are explicit, logical and defensible, these methods provide insufficient support for the complexity of probabilistic ontology development, as discussed above. A systematic approach to PO development is needed that addresses the evolution of requirements into an ontology that is probabilistically integrated. The underlying ontology may be engineered by many methods; here an adaptation of the METHONTOLOGY process is used, but ultimately each methodology provides a structured means to produce ontologies from conceptualization to implementation. Some principal design criteria must always be considered: clarity, coherence, extendibility, minimal encoding bias, and minimal ontological commitment as discussed in Section 2.2.1.2 [60].

3.2.2.3 Ontology Reuse

3.2.2.3.1 Overview

There are two types of ontology reuse: re-engineering and merging. Ontology re-engineering involves transforming the conceptual model of an implemented ontology into another conceptual model [56]. On the other hand, ontology merging uses information captured about one or more domains of interest in the creation of a new ontology. Therefore, model reuse is the process by which available knowledge and conceptual models are used as input to generate new models, in this case ontologies and probabilistic ontologies. Ontology development is a complex and labor-intensive task. The potential for reuse is an identified strength of ontologies and allows expansion of existing knowledge bases by capitalizing on previous research and development. The literature liberally addresses the concept of ontology reuse, but there is little guidance offered in methods for merging and/or integration. Integration of similar tasks and the addition of tasks emphasizing utility of existing ontologies expand the basic process of ontological engineering to make use of ever-expanding online ontology resources. Before beginning construction of a new ontology, it is useful to research existing ontologies in related domains to be reused and/or extended for the current problem. The ST community is actively expanding free access to the growing body of ontological knowledge, as discussed below.

3.2.2.3.2 Ontological Model Databases

As the ever-expanding application of ontologies continues within the ST community, databases and portals have been established to allow sharing among ontological engineers. It is through these open-source resources that developers may freely search for existing ontologies that may be imported or extended to aid in a new domain application. Some of the more common portals include:

- WebOnt.org – Web Ontology Portal: www.webont.org/
- DAML Ontology Library: http://www.daml.org/ontologies/
- SemanticWeb.org: http://semanticweb.org/wiki/Ontology
- Open Ontology Repository (OOR): http://ontolog.cim3.net/cgi-bin/wiki.pl?OpenOntologyRepository

Ontologies are also often available on individual investigator or laboratory websites, SourceForge, and other online sites.

More recently, several forums and a wiki have been established to create a community of practice for sharing of best-practices and lessons learned. Some of the more active forums include:

- ONTOLOG – collaborative work environment: http://ontolog.cim3.net/
- OntologWiki: http://ontolog.cim3.net/cgi-bin/wiki.pl/
- The Ontology Forum: http://www.ontologyforum.com/

Through the above portals and the forums available on the assorted wikis, the PO Developer may find existing resources that will reduce development time and identify best practices.

3.2.2.4 Heuristics and Algorithms

Generally, a heuristic is an experience-based technique for problem solving, learning, and discovery and an algorithm is a stepwise procedure for calculation of a problem solution Heuristics and algorithms are used to express relationships between classes within ontologies and probabilistic ontologies in order to constrain the models. For example, the heuristic "A weapon is cued by a single sensor" gives a plain-language description of a relationship in which each weapon is assigned a single sensor, but sensors may be assigned multiple weapons. This plain language description captures the machine-readable cardinality statement of $\infty\ldots1$ in a format understandable by the entire development group, including the Stakeholder Decision Maker and SMEs. Heuristics and algorithms are captured as part of the PODM as described in Chapter 4.

3.2.2.5 Learning

Currently, ontology development is a labor-intensive, manual process. However, the need for greater automation features has been recognized and is a focus of the ST community. The PODM has integration points primed for future expansion in the areas of Ontological Learning and Probabilistic Learning. These two functions assist the modeler in ontology creation and elicitation of probabilities for the probabilistic relationships used for inferential reasoning.

3.2.2.5.1 Ontological Learning

Ontological learning is the process of extracting relevant classes, properties and relationships from a given data set, in this case to reduce effort in development of an ontology which will be developed into a probabilistic ontology. Buitelar et al. identified innovative aspects of ontology learning that set it apart from traditional knowledge acquisition [18]:

- It is inherently multidisciplinary due to its strong connection with the Semantic Web, which has attracted researchers from a very broad variety of disciplines: knowledge representation, logic, philosophy, databases, machine learning, natural language processing, image processing, etc.
- It is primarily concerned with knowledge acquisition from and for Web content and is moving away from small and homogeneous data collections.
- It is rapidly adapting the rigorous evaluation methods that are central to most machine learning work.

Through application of ontological learning, both the process of developing a probabilistic ontology and the development risk may be reduced.

Sowa defines three types of ontologies: a formal ontology which is a conceptualization whose categories are distinguished by axioms and definitions and are stated in logic to support inference and computation, a prototype-based ontology in which categories are formed by collecting instances extensionally, and a terminological ontology which describes concepts by labels and synonyms without axiomatic grounding [152]. Ontological learning in support of inferential reasoning is concerned primarily with developing the latter two categories for the specified domain of interest. The various sources used for ontology elicitation may include databases, documents, taxonomies, and relational databases. As ontologies are typically hierarchically arranged, the primary means for ontological learning is through clustering. In this method, using a suitable clustering algorithm, a semantic distance is measured between terms and the nearest terms are clustered and formed into a prototype-based ontology. Ontological learning may also be accomplished through pattern matching using a co-occurrence matrix or bootstrapping from a seed lexicon that is extended by measuring similarity.

The above methods are all primarily focused on learning ontologies from plain text corpuses. Recent work includes extracting ontologies from non-text formats including relational databases, structured knowledge bases, databases, and the Semantic Web. Albarrak developed an extensible framework for generating ontologies from Relational Database (RDB) and Object-Relational Database (ORDB) data models [1]. Li et al. introduce a novel set of 12 learning rules that build a complete OWL ontology of classes, properties, characteristics, cardinality and instances [102]. A database analyzer extracts key information from the relational database, which is then passed to an ontology generator containing the rules. It is also possible to map ontologies through machine learning to transform existing ontologies within the Semantic Web to a format useable in the domain context for the current problem. Doan et al. have introduced the GLUE system to semi-automatically create these semantic mappings using a multi-strategy learning approach based on the joint probability distribution of the compared concepts [42][43]. The concept is to produce a map between the existing domain and the desired domain that translates between taxonomies. Future research promises to reduce the human interaction required for ontological engineering.

3.2.2.5.2 Probabilistic Learning

Elicitation of conditional probabilities to populate distribution tables remains a difficult endeavor, accomplished through SME interview and experimental data collection. Probabilistic learning seeks to reduce the effort involved in establishing prior and conditional probabilities for domain entities by specifying a model using empirical data. Pearl identified two tasks for probabilistic learning [129]:

i. Extracting generic hypothesis evidence-relationships from records of experience, and
ii. Organizing the relationships in a data structure to facilitate recall.

Accuracy and consistency in the PO model could be improved by learning numerical parameters for a given network topology from empirical data instead of relying on SME input. The literature contains numerous techniques for parameter learning; two commonly employed methods are:

- Maximum Likelihood [39][52] – Parameters are estimated from a set of empirical data using a likelihood weighting algorithm.
- Bayesian Learning [39][52] – Prior knowledge about parameters is encoded and data is treated as evidence to reduce the learning process to calculation of posterior distributions.

For the MilShip PO, a data set is not available, and parameter learning is not implemented.

Learning is segregated into the categories of structure learning and parameter estimation [37][52]. In parameter estimation, the dependency structure of the probabilistic representation is known. The learning task is to define the parameters of the LPDs. The goal of structure learning is to extract the structure of the probabilistic representation from the dataset. A discussion of learning for the EPL models introduced in Section 2.4.5 follows.

Learning a PRM requires input in the form of a relational schema that describes the set of classes, the attributes associated with the classes, and the relations between objects of classes for the domain. In the parameter estimation task, the structure is given, which defines the parents for each attribute. The parameters that define the CPDs for the structure are learned using the likelihood function to determine the probability of the dataset given the model. Structure learning of a PRM is more complex and requires a method to find possible structures and then score them. Getoor et al. describes the use of a greedy local search procedure to produce a candidate structure which is then scored using the prior probability of the structure and the probability of the dataset, given the structure [52].

Recall that the structure of a MLN includes a node for each variable and a potential function for each set of nodes that is pairwise linked. Parameter estimation for MLN is performed by computation of the Markov network weights that represent the clique potential using an optimization of the likelihood function. Structure learning is performed by a greedy algorithm on the network features [45].

MEBN learning also takes advantage of the structure associated with a relational database. A key component is generation of a MEBN-RM model that specifies a mapping of MEBN elements to the relational model of the database. MEBN parameter learning estimates the parameters of the local distribution for a resident node of an MTheory, given the structure and the database using maximum likelihood estimation. MEBN structure learning organizes random variables into MFrags and identifies parent-child relationships between nodes, given the database. Any Bayesian Network Structure search algorithm may be used [124]. More recently, Park et al. has extended the MEBN learning algorithm to include both discrete and continuous random variables [125]

3.2.2.6 Knowledge Base

The knowledge base is a historic collection of domain-specific knowledge contributed by domain SMEs and may include ontological information (classes, properties, characteristics, and relationships), logical constraints, heuristics, and probabilities. The breadth of knowledge stored within is unspecified. To distinguish the

KB from evidence, there is no temporal component associated with the knowledge base; information contained therein may not represent the current domain state. Marakas differentiates a database from a knowledge base in this fashion:

> *"… a collection of data representing facts is a database. The collection of an expert's set of facts and heuristics is a knowledge base* [104]*."*

3.2.2.7 Ontology Structures

Ontologies, including probabilistic ontologies, provide a means to represent knowledge and relationships between hierarchically organized classes of objects. Ontologies exist to enable knowledge sharing and reuse [29][60]. As a set of definitions of formal vocabulary, ontologies allow knowledge sharing among hierarchically organized entities. A probabilistic ontology addresses the inherent uncertainty involved in inferential reasoning applications with inconclusive evidence by representing it probabilistically.

3.2.2.7.1 Ontology

A working ontology captures the classes, properties, and the relationships of a domain of interest. Production of this relational framework facilitates comprehension of the hierarchical organization of domain entities; the relationships between and properties of domain entities; as well as causal relationships among entities. When uncertainty about aspects of the domain is important to the purpose for which the ontology is being developed, a probabilistic ontology is needed to represent the uncertainty.

3.2.2.7.2 Probabilistic Ontology

A probabilistic ontology provides a means to represent and reason with uncertainty by integrating the inferential reasoning power of probabilistic languages with the first-order expressivity of ontologies. Few things are certain, and inferring in the presence of uncertainty allows the decision maker to focus attention on the most relevant data through designed queries.

3.2.3 Support Layer

The Support Layer provides the background technology and design strategy necessary to instantiate the conceptualization of a specific probabilistic ontology to satisfy identified requirements. It includes existing ontologies available for reuse or re-engineering, software tools that enable ontology and probabilistic ontology development, mathematical languages that allow representation of entity attributes and their relationships, and databases of existing facts referenced for learning and knowledge base population. The purpose of the Support Layer is to facilitate probabilistic ontology development by identifying technological and semantic features specific to a particular inferential reasoning model. The four Support Layer components are discussed below.

3.2.3.1 Existing Ontologies

As previously discussed in Section 3.2.2.3, model reuse is a strength of the ontological engineering discipline and effort should be made to research and incorporate existing ontology material into new application areas. This will reduce overall effort and promote commonality among different products. Some suggested ontology repositories are listed in Section 3.2.2.3.2.

3.2.3.2 Modeling Languages

A modeling language is a graphical or textual representation used to express knowledge, information, processes or systems with a consistent set of rules and syntax. In the RAPOD, modeling languages serve three functions:

- System Architecture Representation
- Object Relationship Representation
- Ontology (and Probabilistic Ontology) Representation

A probabilistic ontology is an extension of an ontology which incorporates uncertainty while respecting its relational structure and domain specificity. The output of the RAPOD is a unique instantiated architecture for development of a domain-specific probabilistic ontology to meet an inferential reasoning requirement. The architecture includes models from each of the above representation categories and may be reused for development of new probabilistic ontologies in similar domains. The following sections describe the purpose of these representations and the instantiations used to create the MilShip PO.

3.2.3.2.1 System Architecture Representation

An architecture is a conceptual design that defines the structure and behavior of a system. There are two types of representations commonly employed: traditional and object-oriented, represented here by IDEF0 and UP.

- Icam Definition for Function Modeling (IDEF0) – IDEF0 is a process modeling technique that focuses on the functional model of a system. The model is expressed as a set of diagrams, often called pages. IDEF0 has been applied to the development of information systems, business processes and hardware systems [17].
- Unified Process (UP) – UP is an iterative, comprehensive development approach adapted to object oriented models, tools and techniques [140]. It was developed initially for software systems, but in recent years has been adapted to systems that include hardware and business processes.

IDEF0 is commonly associated with hardware systems and systems-of-systems, especially within the Department of Defense Architecture Framework (DODAF). Class hierarchies are fundamental to ontologies, and object oriented design is focused on modeling class hierarchies. Therefore an adaptation of UP is used to create the MilShip PO Architecture.

3.2.3.2.2 Object Relationship Representation

Object modeling languages are used to represent relationships at the system and object level of abstraction to enable clear, concise communication between Stakeholder Decision Maker and the PO Developer. While the specific choice of language is often left to the developer, object relationships are frequently represented using languages such as:

- Unified Modeling Language (UML) – UML is a graphical modeling language for the creation of object-oriented models used primarily for software engineering [140].
- Systems Modeling Language (SysML) – SysML extends UML language with semantic foundation for representing requirements, behavior, structure, and properties of systems and components [49][50].

There are many diagrams and representations appropriate to systems architecting available in both UML and SysML; the PO Developer should select and implement these tools to maximize clear communications with the Stakeholder Decision Maker. Diagrams associated with development of the MilShip PO are created in UML.

3.2.3.2.3 Ontology Representation

Ontology languages allow developers to create explicit, formal conceptualizations of domain models. The main requirements of an ontology language identified by Antoniou and Harmelen include [3]:

- Well-defined syntax
- Well-defined semantics
- Efficient reasoning support
- Sufficient expressive power
- Convenience of expression

Ontology languages are formal, declarative representations that allow compilation and organization of knowledge about a domain in formal knowledge structures with clearly defined semantics. Further, they include reasoning rules to represent relationships between knowledge classes. The literature contains many different ontology languages, some of which are optimized for specific domains. Some of the more common examples include [56]:

- Web Ontology Language (OWL) – Created by W3C, derived from DAML+OIL and builds on RDF(S).
- Resource Description Framework (RDF) – Created by W3C as a semantic network based language to describe web resources.
- Knowledge Interchange Format (KIF) (including OntoLingua) – Based on FOL with an underlying frame paradigm, overlaid by OntoLingua to simplify operator functionality.
- DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL) – Created by US and EU committee, an extension of RDF(S) with datatypes and nominals. DAML+OIL has been superseded by OWL.
- CycL – A declarative language used to represent the knowledge stored in the Cyc Knowledge Base [38].
- Common Logic (CL) – A FOL language for knowledge interchange approved and published as an ISO standard for representation and interchange of information and data among disparate computer systems [80].
- Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) – A FOL reference module of the Wonderweb Project adopted as a starting point for comparing and elucidating relationships between ontologies [79].

- Basic Formal Ontology (BFO) – An upper-level ontological framework used in support of domain ontologies developed for scientific research [78].

OWL has been selected by the World Wide Web Consortium (W3C) as the language of the Semantic Web and has therefore received broad attention in the research and development communities. Further, OWL is the ontology language used by the UnBBayes software tool, allowing evolution of an ontology to a probabilistic ontology without the need to recreate the classes, instances, and relationships in a new tool. Recall that PR-OWL expresses MEBN in OWL [29]. Of the above ontology languages, only OWL allows expression of probabilistic information along with an ontology through the PR-OWL extension. OWL is used throughout this work to create the MilShip Ontology that is later transformed into the MilShip PO.

3.2.3.2.4 Probabilistic Ontology Representation

As previously introduced in Chapter 2, probabilistic ontologies are used to comprehensively describe knowledge about a domain and the uncertainty embedded in that knowledge in a principled, structured and sharable way [29]. The probabilistic web ontology language (PR-OWL) and its successor (PR-OWL 2) provide a knowledge representation formalism with MEBN as the underlying semantics. A Multi-Entity Bayesian Networks represents knowledge about attributes of entities and their relationships as a collection of similar hypotheses organized into theories which satisfy consistency constraints ensuring a unique joint probability distribution over the random variables of interest [30]. MEBN uses context constraints to specify logical conditions that determine influence of one random variable over another and FOL sentences to incorporate evidence. The MilShip PO in the example problem is represented in MEBN throughout this work.

3.2.3.3 Software Tools

Modeling tools represent the software implementation packages used for development and implementation of architectures, ontologies, and probabilistic ontologies in the chosen modeling language. With the appropriate modeling tools, the entire ontology life cycle may be managed, including design, implementation, enhancement, and support.

A number of tools are available to capture data and model the components of a probabilistic ontology. During the Framing Activity of the PODM, the PO Developer selects software tools with the correct fidelity to represent relevant viewpoints and provide the desired communication and inferential reasoning representation. Three categories of tools are germane:

- General Purpose Modeling Tools
- Ontology Engineering Software Tools
- Probabilistic Ontology Engineering Software Tools

A combination of these categories gives the PO Developer flexibility in creating necessary views for communication, as well as operational ontology and probabilistic ontology models.

3.2.3.3.1 General Purpose Modeling Tools

Creation of a probabilistic ontology requires representation of many abstractions of data, processes, and relationships, each of which may be best represented in a different software application. However, to the extent possible, a single, general-purpose tool should be maximized to enhance readability and consistency. Below, two useful software tools are introduced. These are neither exclusive nor exhaustive in functionality, but provide a sample of the types of general tools that aid a developer in probabilistic ontology creation.

Microsoft Visio is a visual representation tool with templates for many languages including UML, IDEF, and SysML. Diagrams are used to simplify and communicate complex concepts. In the MilShip PO Architecture, Visio is employed to illustrate concept diagrams, class diagrams, context diagrams, taxonomies, use-case diagrams, and the overall architecture.

MagicDraw is a systems engineering tool that facilitates representation of complex object-oriented models (UML, SysML). Changes within a view are matriculated throughout the model, and updated in all views that share the object.

3.2.3.3.2 Ontology Engineering Software Tools

Ontological engineering tools capture the classes, properties, and instances of ontology entities in a hierarchical structure. Further, they describe their relationships, domains and ranges in a contextual environment. The most popular ontological engineering tool is Protégé, currently in version 4.1.0 (build 239). Protégé also has the advantage of integration with UnBBayes, which allows seamless implementation of uncertainty to establish the probabilistic ontology. Protégé is used to create the MilShip Ontology, as discussed below.

3.2.3.3.3 Probabilistic Ontology Engineering Software Tools

Few tools are able to model the complex integration of probability and ontologies. The most advanced is UnBBayes, an open source product developed by University of Brasilia and enhanced in collaboration with George Mason University. UnBBayes has a PR-OWL plug-in that ingests a Protégé ontology and allows the developer to represent uncertainty within its hierarchical structure through MEBN Fragments using the Probabilistic Web Ontology Language (PR-OWL 2). UnBBayes is used to evolve the MilShip Ontology into the MilShip PO by representing uncertainty in relationships between classes.

3.3 Military Ship Probabilistic Ontology Architecture

Creating an architecture for a given domain problem results in a reusable blueprint for similar designs that facilitates successful development from conceptualization to operation. Using the RAPOD, an architecture is instantiated for the MilShip PO example problem introduced in Chapter 1. Recall that the objective of the Stakeholder Decision Maker is to develop a DSS that assists in the determination of a class of warship given limited input information.

3.3.1 Decision Support Systems

A decision support system is an interactive, computer-based information system that supports business or organizational decision-making activities through compilation, processing and display of domain information. According to Marakas,

> *"A DSS is a system under control of one or more decision makers that assists in the activity of decision making by providing an organized set of tools intended to impose structure on portions of the decision-making situation and to improve the ultimate effectiveness of the decision outcome* [104].*"*

With the ever-increasing volume of information delivered to the decision maker, the ST community seeks to provide advanced decision support through data compilation, screening, transformation and probabilistic inference. Input may include raw data, documents, interviews, and mathematical models stored in databases, ontologies, and probabilistic ontologies. Because each DSS is domain-specific, it has a narrow focus of applicability and will only address a narrow set of decisions. In this construct, the DSS is the final product for the Stakeholder DM supported through implementation of the RAPOD.

3.3.2 Concept Diagram

The vision shown in Figure 2 illustrates the scope of the reference architecture supporting the Military Ship example problem in which a decision maker utilizes a DSS to produce a decision informed by available evidence regarding the current operational environment. The MilShip DSS example problem is used as a running example for demonstrative purposes. However, the flexibility of the RAPOD allows the PO Developer to design an architecture for any probabilistic ontology implementation.

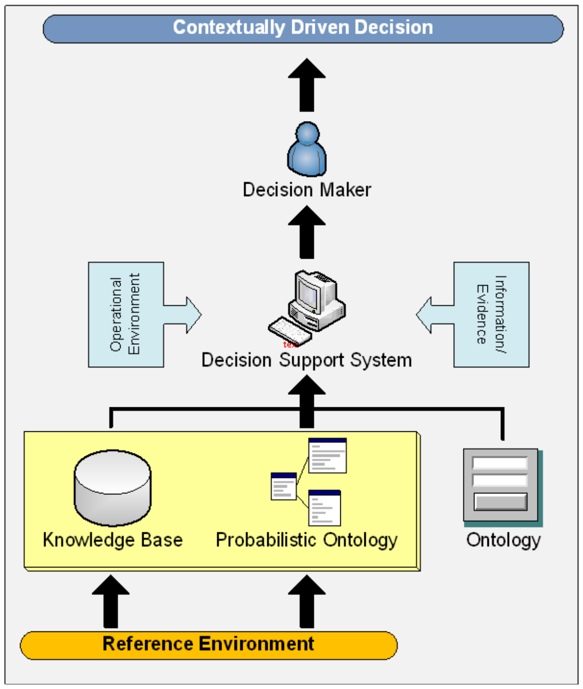


Figure 4 - Concept Diagram for MilShip PO DSS

An ontology of relevant, hierarchical relationships among anticipated classes is constructed or extended in relation to the current context. Then, uncertainty is introduced based on a reference environment representing a contextually relevant situation. Evidence from the available knowledge base is applied to the probabilistic ontology to provide the DSS with inferential reasoning support that is tailored for the operational domain. After implementation and during operations, the DSS continually receives updated information about the current operational situation and changes to the environment. These data will update the knowledge base, and therefore the probabilistic ontology. The end result is a DSS that produces contextually-driven decisions about the domain of interest using both historical and current evidence.

The MilShip PO is used throughout this dissertation as a running example. In Figure 5, the MilShip PO architecture is mapped to the RAPOD as an instantiation of the reference architecture.

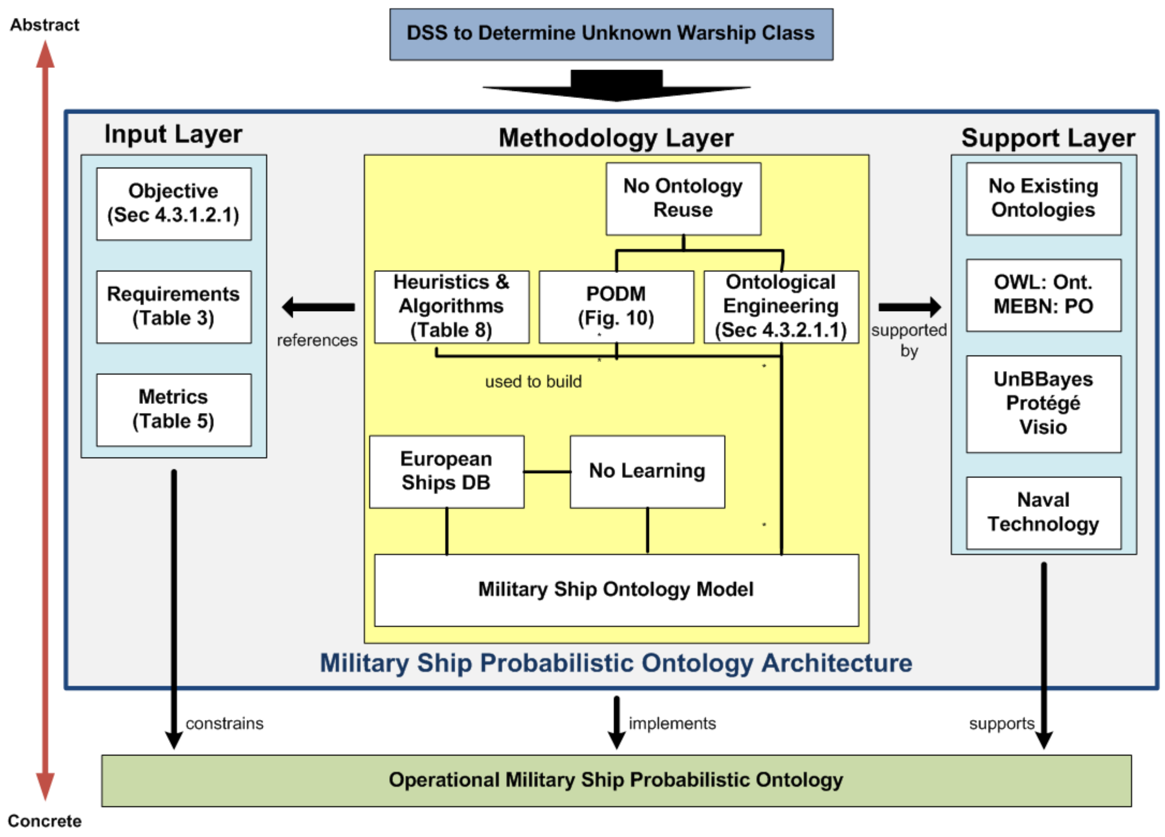


Figure 5 - Military Ship Probabilistic Ontology Architecture

The architecture in Figure 5 illustrates the MilShip PO Architecture from conceptualization as a DSS that is required to determine warship class to the operational implementation of a PO that performs that function. In the Input Layer, references to appropriate tables in Chapter 4 lead to specification of objectives, requirements, and metrics. Similarly, in the Methodology Layer, heuristics and algorithms, the PODM, and ontological engineering are linked with their appropriate figures in Chapter 4. There were no ontologies or existing empirical data sets available for reuse, eliminating reuse and learning functionality from the architecture. A database of selected European warships was constructed as the knowledge base, which captured several attributes for multiple ship classes from various nations. Ontological engineering was performed on this KB to create the Military Ship Ontology in Protégé. The Support Layer consists of technological artifacts highlighted by the OWL and MEBN languages used to represent the ontology and probabilistic ontology, respectively. Software included Microsoft Visio for graphical representation, Protégé for ontology modeling and UnBBayes for probabilistic ontology modeling. Finally, the Naval Technology Wiki is used to generate the European warship database. From this blueprint, the MilShip PO is developed using the PODM specified in Chapter 4. Details of individual tasks are given in Appendix A.

3.4 Summary

As shown in Section 3.3, use of a reference architecture facilitates design, implementation, and reuse of a domain-specific probabilistic ontology construction process by specifying the logical choices of components to create a blueprint for a contextual solution. The instantiated architecture is available for reuse to solve like problems in similar domains.

CHAPTER FOUR: PROBABILISTIC ONTOLOGY DEVELOPMENT METHODOLOGY

4.1 Background

At the heart of the Reference Architecture for Probabilistic Ontology Development lies the Probabilistic Ontology Development Methodology (PODM). PODM activities span the phases of the Systems Development Life Cycle, and are grounded in model-based systems engineering principles. The following figures demonstrate how the PODM is applicable to either the traditional Waterfall Development, or the Spiral Development Cycle. Development phases of a typical Waterfall SDLC are shown in Figure 6 [140]. The color codes associated with these phases are used consistently through the remainder of this work to aid in reader comprehension. A simple probabilistic ontology model may be completed using a single pass through this process.

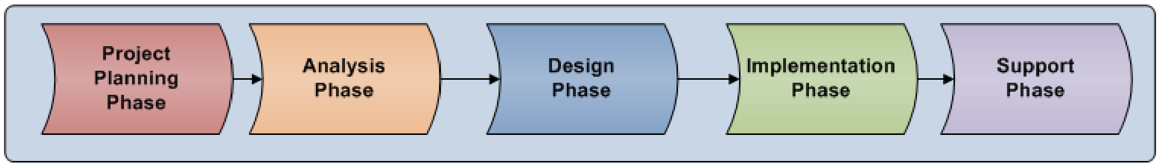


Figure 6 - Traditional Systems Development Life Cycle [140]

Complex development problems require a series of spiral cycles to bring the model from conceptualization to final operation. Spiral Development is suited to this method of development, with each spiral incorporating Planning, Analysis & Design, Development & Test, and Support phases. The Waterfall Development phases introduced in Figure 6, and their PODM-specific tasks, are overlaid onto a Spiral Development Cycle in Figure 7 [17][127][95][76].

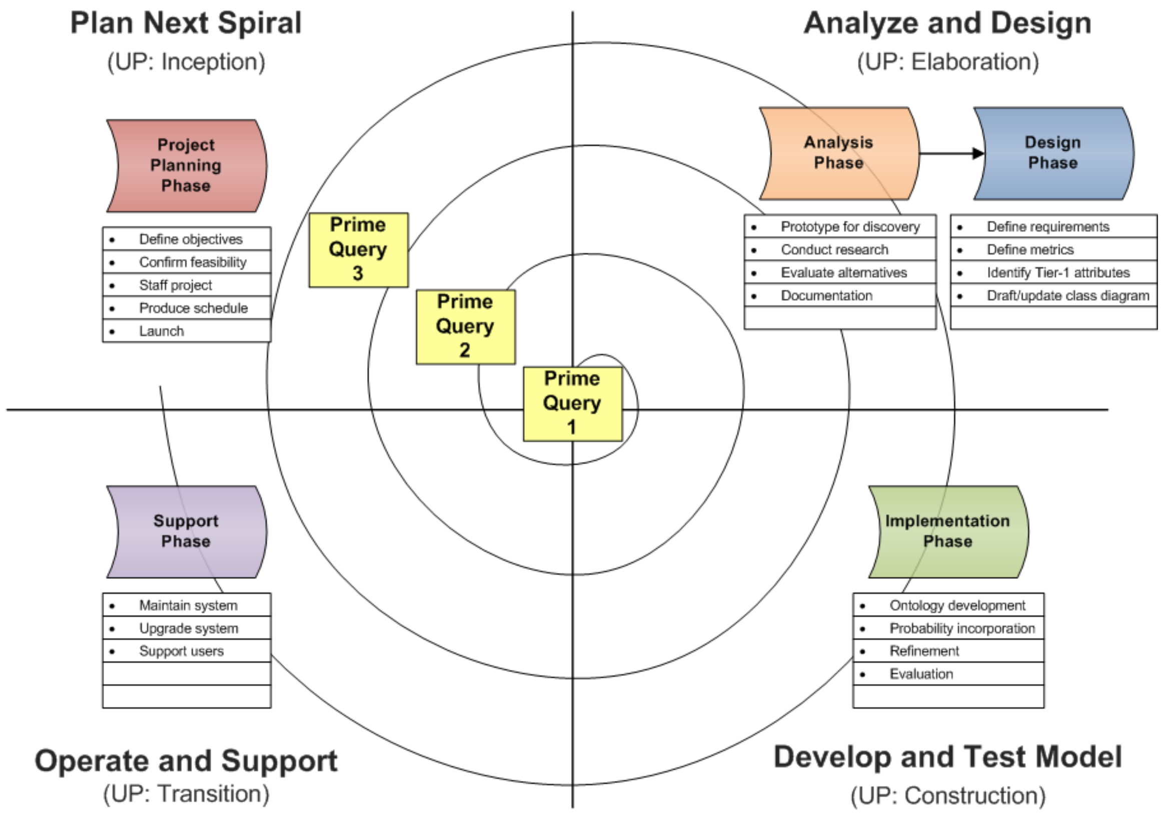


Figure 7 - PODM in Spiral Development Cycle

The Spiral Development Cycle phases also correspond to similar phases in the Unified Process (UP): Inception, Elaboration, Construction, and Transition [95]. The PODM is equally applicable to all three development processes, as captured in Table 2. PODM activities are specified in the left column and their mapping to the three processes is defined within each row. Further description of these activities is given below. The Waterfall process performs a single pass through each of the development activities in the table, top to bottom. Both Spiral and UP processes perform all of the development activities to some extent at each spiral, as discussed below.

Table 2 - PODM Development Cycle Alignment

	SDLC Phase [140]		
PODM Activity	**Waterfall**	**Spiral**	**Unified Process**
Frame	Planning Analysis Design	Plan Analyze & Design	Inception Elaboration
Ontology Development **Probability Incorporation** **Refinement** **Evaluation**	Implementation	Develop & Test	Construction
Operation	Support	Operate & Support	Transition

The remainder of this work assumes a Spiral Development Cycle is chosen to complete development of the probabilistic ontology.

Recursion is prevalent throughout the PODM. To alleviate confusion, the following terms are applied consistently throughout the remainder of the work.

Spiral: A spiral describes an iteration of the Spiral Development Cycle. Each spiral has a unique Objective Statement and one or more supporting Prime Queries. The Spiral Core Model contains the Prime Queries.

Iteration: An iteration describes a recursive step within the PO Construction Process. Iterations expand the Spiral Core Model by decomposing its attributes.

4.2 Overview of PODM in Spiral Development

A probabilistic ontology developed using the Spiral Development Cycle experiences all of the four phases in each spiral. While the first spiral has the heaviest emphases on Planning and Analysis & Design, it is also important to review and update these phases for each subsequent spiral to ensure the project continues to be aligned to the Stakeholder Decision Maker's objectives. Within the PODM, these two phases are collectively referred to as the Frame Activity because they frame the scope of the development for the spiral. The Frame Activity highlights those areas in the Planning and Analysis & Design phases that are updated under the assumption that the management tasks remain stable throughout development.

Within the Develop & Test Phase of the Spiral Development Cycle a Probabilistic Ontology Construction Process is incorporated that specifically addresses the evolution of requirements into an ontology that is probabilistically-integrated. This detailed process explicitly describes the iterative tasks required to produce a PO with in-situ evaluation steps to ensure continuous operation of a probabilistic ontology produced for inferential reasoning.

The Operate & Support phase encompasses fielding and operation of the system, as well as a means to conduct upgrades. Because the system is developed iteratively, the system will be in operation before it is complete and support must be provided to the users.

4.2.1 Plan Spiral (Project Planning Phase / UP: Inception)

Planning the Spiral of the Spiral Development Cycle is a crucial management step to ensure the project has the necessary support to complete the spiral. Before the first spiral is begun, the Stakeholder Decision Maker and PO Developer collaborate to establish the overall project objective, feasibility, staffing support, and schedule [140]. This is the project launch. Subsequently, the Planning phase only requires updates to specify new spiral objectives and their associated schedule revisions. These updates, combined with those from the Analyze & Design phase, make up the PODM Frame Activity.

Arguably one of the most important tasks within the pre-launch Planning phase is selection of the proper representation to meet the overall objective of the project. The breadth of solutions to satisfy individual stakeholder objectives is limitless, and not every project requires the commitment in time and effort associated with producing an ontology or probabilistic ontology. Figure 8 outlines one view that may be used to aid in determination of the appropriate modeling paradigm.

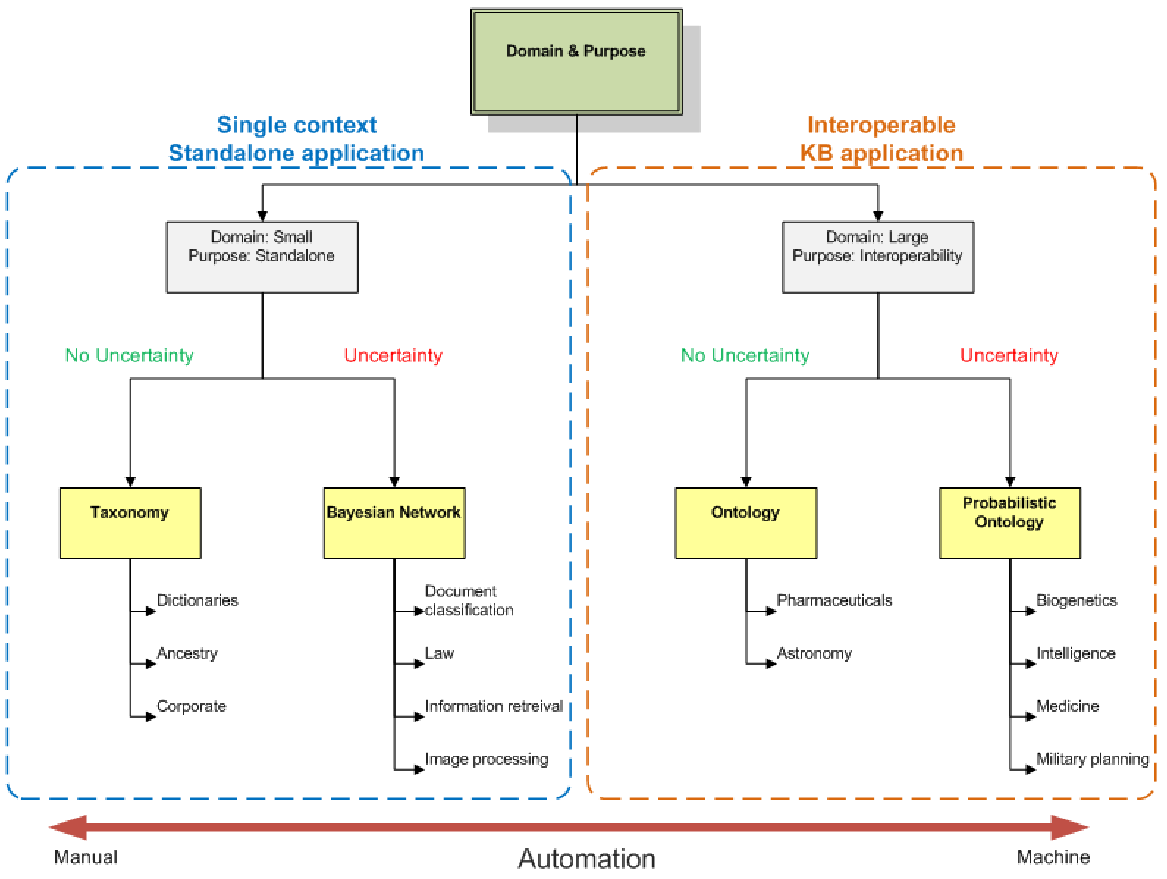


Figure 8 - Alternative Representations

The entering argument for selection of the appropriate representation is the overall domain of interest and the purpose of the model. The blue highlighted area captures representations for relatively simple problems that may be solved by one-of solutions, or those with a relatively small domain. Within this solution set, a taxonomy represents a hierarchical structure of a smaller domain without the need to represent uncertainty. On the other hand, a Bayesian network solution is appropriate if uncertainty must be introduced to a less-complex problem. The orange highlighted area captures representations of large domains of complex problems that often require interoperability between knowledge bases. If the solution requires classical logic in a closed system with an unambiguous vocabulary, then an ontology representation may be appropriate. However, if reasoning is required to resolve uncertainty within the large domain, then a probabilistic ontology is a better selection. Beneath each of the four representations in Figure 8 are some example applications from the literature.

At the bottom of the figure is a spectrum of automation that illustrates a proposed manual-machine scale for application of the representations. On the left (manual) end are solution representations created predominantly by manual techniques. Moving to the right increases the amount of automation until eventually the representations are machine-developed. As introduced in Chapter 3, current state of the art allows ontological learning of text corpuses and probabilistic learning through machine learning techniques. Recent research has introduced MEBN learning which can produce a learned probabilistic ontology [124][125].

4.2.2 Analyze & Design (Analysis and Design Phases / UP: Elaboration)

The Analyze & Design phase of the Spiral Development Cycle sets the stage for a successful spiral by thoroughly researching and documenting supporting information to achieve the spiral objectives and then detailing the requirements and metrics used to satisfy and measure those objectives. In the Analysis Phase, research is conducted to understand the problem domain as it relates to the project objective, including: observation, interview, document review, and existing system review. Satzinger et al. suggests prototyping as a method to understand requirements that may satisfy spiral objectives [140]. Partial systems represented by prototypes provide a venue for interaction with the Stakeholder Decision Maker and eventual users to elicit requirements. Information gathered throughout the analysis is documented for incorporation into the solution.

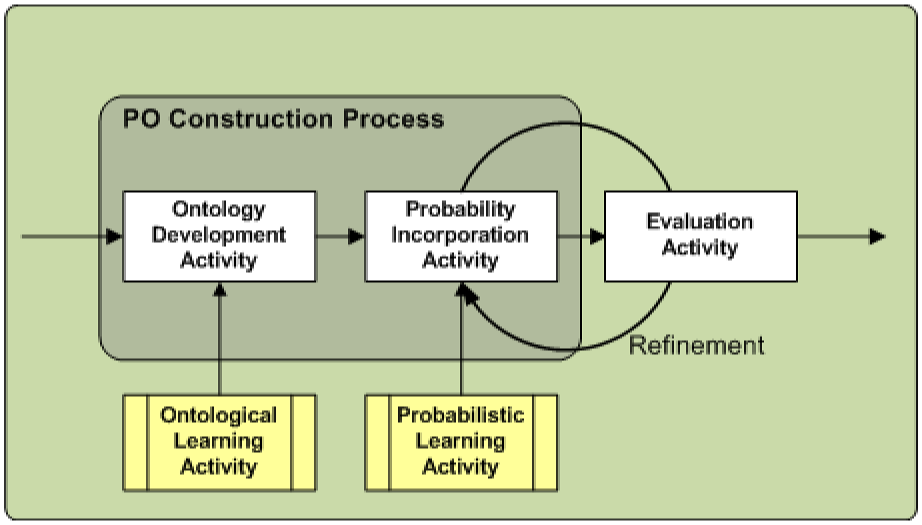
The Design Phase employs Stakeholder DM input and collected research material to create definitive requirements that the solution must satisfy to meet the objectives of the spiral. Metrics are developed that grade these requirements to quantitatively assess performance against the requirements. Finally, attributes and their relationships germane to the spiral are collected and captured in a class diagram from which development begins.

Combined with Planning the Spiral, the tasks within the Analyze and Design phase make up the PODM Framing Activity. After the project is launched, subsequent spirals of the development cycle require application of the Framing Activity to incorporate appropriate updates from these phases.

4.2.3 Develop & Test (Implementation Phase / UP: Construction)

The Develop & Test Phase includes the PODM activities of Ontology Development, Probability Incorporation and Evaluation. These activities comprise the heart of the PODM by defining the ontology, adding a representation of uncertainty and evaluating the model against requirements. Two recursive cycles exist within the Develop & Test Phase. The first begins with a simple probabilistic model that is incrementally expanded through refinement to establish a probabilistic ontology model that spans the entire scope of the decision problem, and is referred to as the Probabilistic Ontology

Construction Process. The second cycle involves evaluation of the model and further refinement to correct logic errors and unanticipated relationship effects.


PO Construction Process
Ontology Development Activity
Probability Incorporation Activity
Evaluation Activity
Refinement
Ontological Learning Activity
Probabilistic Learning Activity
Figure 9 - PODM Construction Process

The shaded area in Figure 9 delineates the iterative steps of the recursive PO Construction Process. During both initial PO construction and updates on subsequent spirals, multiple construction iterations may be performed to ensure each interim step is evaluated for valid relationships and correct logic. Similarly, recursion exists between the Evaluation and Probability Incorporation Activities. Errors discovered during the Evaluation Activity prompt refinement of the model for correction. These in turn may require further application of the Probability Incorporation Activity.

4.2.4 Operate & Support (Support Phase / UP: Transition)

The Operate & Support Phase includes three major functions: maintenance, improvement, and operational support [140]. Maintenance and operational support keep the current build in service and enable users to work through the continuing development process. Improvement identifies future increment capabilities that will be ranked for prioritization in the next Spiral Development Cycle. Upgrades can be as simple as adjusting probabilistic relationships, or as complex as adding additional evidential nodes to the overall reasoning process. All require some level of iterative refinement and evaluation to ensure model logic and consistency are maintained.

4.3 Probabilistic Ontology Development Methodology Details

The above phases are combined to form the PODM for a single spiral of the development cycle, illustrated in Figure 10. The remainder of this section delves further into the details of the PODM utilizing the previously introduced Military Ship Probabilistic Ontology as a running example. The activities of the PODM and their tasks are illustrated in Figures 11, 13, 18 and 33. Completion of these activities establishes a design solution to a specific decision problem grounded in an inclusive ontology representing its entities and incorporation of probability to represent uncertainty.

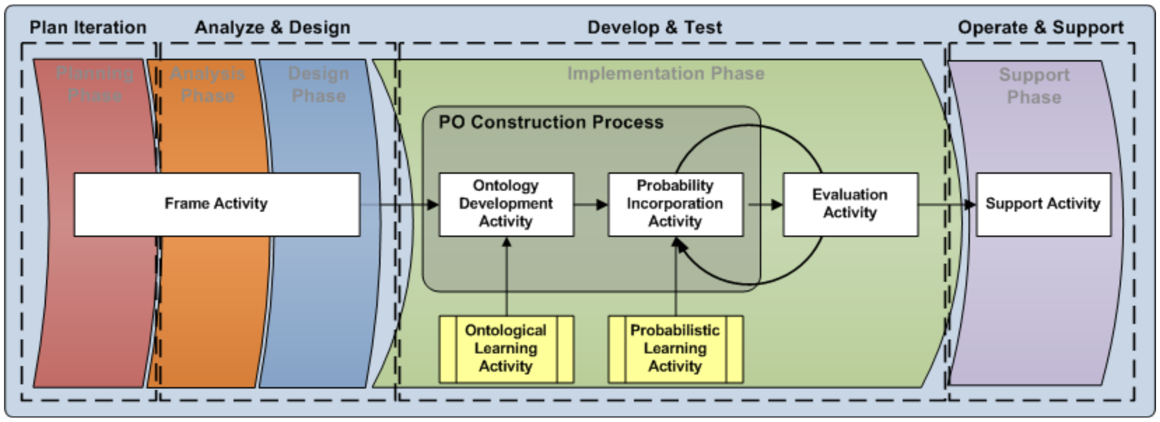


Figure 10 – Single Iteration of the PODM

A complete description of each activity and its component subtasks follows. Specific application of the PODM to the MilShip PO is included to aid in comprehension, where applicable. The interested reader will find a complete detailing of development of the MilShip PO using the PODM in Appendix A.

4.3.1 Frame Activity

For each spiral of the development cycle, the Frame Activity encompasses necessary tasks to scope the problem and its requirements based on the Objective Statement. The five tasks shown in Figure 11 culminate with an initial class diagram that is used to identify the probabilistic relationships of the Spiral Core Model before beginning the Probability Incorporation Activity.

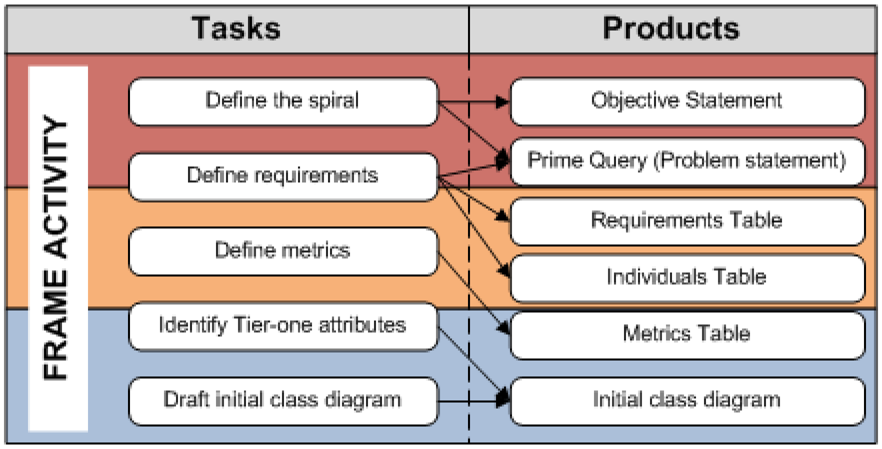


Figure 11 - Frame Activity

Key products of the Frame Activity include an Objective Statement defining the overall purpose of the spiral, one or more Prime Queries established to satisfy the objective of the Stakeholder DM, and Tier-one attributes that immediately affect the Prime Queries. The Prime Queries and their associated Tier-one attributes define the Spiral Core Model iterated in the PO Construction Cycle to create the PO used for inferential reasoning. Supporting and informing the Spiral Core Model are other traditional systems engineering products including detailed listings of requirements, individuals, and metrics.

4.3.1.1 Frame Activity Tasks

4.3.1.1.1 Define the Spiral

Defining the spiral establishes the overall objective of the spiral and identifies the Prime Queries that satisfy this objective. Each spiral of the development cycle has a single objective and one or more supporting Prime Queries. Working closely with the Stakeholder DM, the PO Developer crafts the Objective Statement and Prime Queries, as well as constraints on the model and its input to create a formal definition of the problem. Inferring a solution to the Prime Queries is the recurring theme maintained throughout model development.

4.3.1.1.2 Define Requirements

Grady defines a requirement as an essential attribute for a system or an element of a system, coupled by a relation statement with value and units information for the attribute [57]. Using either a top-down, bottom-up, or middle-out process, requirements are elicited that ensure satisfaction of the spiral Objective Statement and captured in a Requirements Table. The goal of this task is to capture attributes that should be controlled within the model in written requirement statements, to be validated by the Stakeholder DM and measured by the metrics. Several methods of requirement elicitation are available in the literature, introduced in Section 3.2.1.2.

4.3.1.1.3 Define Metrics

As described in Section 3.2.1.3, metrics are parameters or measures of quantitative assessment used for measurement, comparison or to track performance of the requirements against some benchmark established in collaboration with the Stakeholder DM. An initial set of metrics based on the requirements defined to support the spiral objective is developed and captured in a Metrics Table. It is best if there is at least one metric to support each requirement of the system. Metrics that do not support a requirement, either directly or indirectly, should be pruned from the metrics table. Armstrong defines a useful metric as one that takes a quantifiable form with a clear definition of the measure and associated units [137]. The metrics may be in the form of a confidence interval or a percentage of correct responses given a defined amount of information, or other suitable measurement.

4.3.1.1.4 Identify Tier-one Attributes

Attributes immediately affecting the Prime Queries establish the minimal probabilistic model that will support the decision of interest, and are referred to as Tier-one attributes. They form the core of the probabilistic ontology model and are expanded upon in the PO Construction Process to complete the entire inference network. The initial class diagram is created from these Tier-one attributes and the Prime Queries. The following steps lead to identification of the Tier-one attributes:

i. Based on the Prime Queries, identify the class of objects about which the reasoning will occur.
ii. Identify relationships that immediately affect the Prime Queries. These are the Tier-one attributes. Causal relationships are established by identifying variables that may cause a variable to take a particular state or prevent it from taking a particular state [89].
iii. At the most general level, identify the classes that are affected by these relationships. The established relationships and classes populate the initial class diagram.

4.3.1.1.5 Draft Initial Class Diagram

The initial class diagram enables the PO Developer to visualize the relationships directly affecting the Prime Queries via the Tier-one attributes. Later, this diagram is iteratively extended in the PO Construction Process to include all attributes and relationships in a systematic and comprehensive fashion. The class diagram shows classes and subclasses of objects that are instantiated in the model. As this is the initial class diagram, clarity is of great importance necessitating the inclusion of both cardinality and relationship information.

4.3.1.2 Frame Activity Products

4.3.1.2.1 Objective Statement

There are two types of objective statements employed in this development process. The Top-level Objective Statement describes the overall purpose of the PO in a manner understandable to both the Stakeholder DM and the PO Developer. The Spiral Objective Statements identify the purpose of each specific spiral and should support the Top-level Objective. Both should be specific, concise, and observable. The Top-level Objective Statement for the Military Ship PO is given as:

> *The Military Ship Probabilistic Ontology will aid the user in inferring the specific class of a warship for a contact of interest given the arrival of uncertain information about its sensors, weapons, nationality and physical characteristics.*

Completion of the MilShip PO requires only a single spiral of the development cycle. Therefore, for this example the Top-level Objective Statement is synonymous with the Spiral Objective Statement.

4.3.1.2.2 Prime Query

A Prime Query defines a principal area of focus for development in the spiral in the form of a question. The inference network seeks to answer this question at the completion of the Develop & Test Phase. Prime Query 1 for the first spiral of the Military Ship PO is given as:

Prime Query – 1: The unknown contact belongs to which of the warship classes in the AOR-specific library?

4.3.1.2.3 Requirements Table

The Requirements Table captures the validated requirements that represent behaviors, applications, constraints, properties, and attributes that directly support the Spiral Objective Statement. The table should include a title and brief descriptive statement for each requirement.

Table 3 – MilShip PO Requirements

Requirements Table		
ID	**Title**	**Description**
R1	**Determine warship class**	**PQ: Determine warship class from library of possible classes in the AOR**
R2	**Accept Reports**	**Incorporate uncertain information from arriving reports**
R2.1	Accept Type Reports	Incorporate reports about the type of warship detected (FF, FFG, CVN, Other)
R2.2	Accept Size Reports	Incorporate reports about size information for the detected ship (Small, Medium, Large)
R3	**Incorporate Class Descriptors**	**Incorporate a library of information about possible classes included in a Military Ship ontology that will be accessed for reasoning**
R3.1	Incorporate Nationality	Incorporate information about the nationality of a class of warship
R3.2	Incorporate Warship Type	Incorporate information about the type of warship for a given class
R4	**Incorporate Mission Information**	**Incorporate a library of information about the primary mission, and the sensors/weapons used to accomplish this mission**
R4.1	Incorporate Primary mission	Incorporate information about the primary mission of a class of warship
R4.2	Incorporate Ship Sensors	Incorporate information about sensors hosted on the class of warship
R4.3	Incorporate Weapon	Incorporate information about weapons carried on the class of warship
R5	**Incorporate Descriptive Information**	**Incorporate descriptive information to assist in classification of gross naval class and size**
R5.1	Incorporate Displacement	Incorporate class displacement data
R5.2	Incorporate Length	Incorporate class length data
R6	**Incorporate Performance Characteristics**	**Incorporate performance information related to deployed system hardware**
R6.1	Execute Quickly	Generate solution in t minutes or less
R6.2	Execute Efficiently	Compute solution on a PC (Intel 1.3 GHz)

Requirements definition requires close collaboration between the PO Developer, the Stakeholder DM, SMEs and users. Section 3.2.1.2 describes several methods for elicitation of requirements.

4.3.1.2.4 Individuals Table

Each class within the ontology contains individuals that are specific to the domain of interest. For example, the MilShip Ontology is comprised of selected European warships and their attributes. However, the same model could be employed for inferential reasoning support in the Pacific if the classes were instantiated with Asian warships. The PO will access the individuals instantiated in the ontology in response to a query to produce an inference result – typically a probability distribution on different answers to the query.

Table 4 - MilShip PO Individuals

Individuals Table	
Class	**Individuals**
Ship	ctc1, ctc2, ctc3, ctc4
SizeRpt	rs1, rs2, rs3, rs4, rs5
TypeRpt	rt1, rt2, rt3, rt4, rt5
Nation	Nation_(DE, ES, FR, Other)
ShipSize	Size_(Large, Small, Medium)

ShipSensor	RAS_(SPY-1D, LW08, DRBJ-11B, DRBV-15C)
	RFC_(Arabel, Castor_2J, DORNA, STIR180)
	RSS_(Aries, SMART-3D, DRBV-15C, DRBN-34)
	SHM_(1160-LF, DSQS-23BZ)

ShipWeapon	WNG_(DCNS, Giat_20F2, Mk45_Mod2, Mk75_OtoMelara)
	WNM_(Aster15, Mistral, MM38, MM40)
	LWT_Mk46

WarshipClass	Class_(AlvaroDeBazan, Brandenburg, CharlesDeGaulle, LaFayette, Unknown)
WarshipMission	Msn_(AAW, ASuW, ASW, Strike)
WarshipType	Type_(CVN, FF, FFG, Other)

This limited set of instances is designed to demonstrate feasibility of the methodology without stressing the UnBBayes software application. An operational application of the MilShip Ontology would have hundreds of instances.

4.3.1.2.5 Metrics Table

Through experience and stakeholder elicitation, performance goals and their associated metrics may be identified and captured for use in model evaluation. Many methods exist to elicit relevant metrics for a given domain. Armstrong proposes a brainstorming process during which rows of the metrics table are elicited by experts using the requirements table as a map. Table 5 shows a partial decomposition of the Metrics Table for the Military Ship PO. The complete Metrics Table is included in Appendix A.

Table 5 - MilShip PO Metrics

Metrics Table					
Requirement		**Metric**			
ID	**Name**	**ID**	**Name**	**Definition**	**Units**
R1	Warship class	M1	Model Accuracy	Correctly identify the warship class (≥ 85%)	Percent
R2 R3 R4 R5	Reports Descriptors Mission Descriptive	M2	Model Flexibility	Absorb/operate on ontology of Z items (max expected size)	Items
R6	Performance	M3	Execution Time	Generate solution in t minutes or less	Min
R6	Performance	M4	Model Efficiency	Compute solution on pc computer (Intel 1.3GHz)	Processor

These metrics are used to validate the spiral model against its requirements.

4.3.1.2.6 Tier-one Attributes

As previously introduced, the Tier-one Attributes have immediate effect on the Prime Queries for the spiral by virtue of their immediate proximity.

Table 6 - MilShip PO Tier-one Attributes

Tier-1 Attributes Table	
ID	**Tier-one Attribute**
T1	Ship Size
T2	Ship Type
T3	Nationality

The Tier-one Attributes are also the nearest neighbors to the Prime Queries' classes in the initial class diagram. Cementing the Spiral Core Model through thoughtful determination of both the Prime Queries and Tier-one Attributes ensures a model that is both relevant and coherent, meeting the spiral objective.

4.3.1.2.7 Initial Class Diagram

The initial class diagram establishes the core of the probabilistic ontology model and is iteratively expanded in the PO Construction Process to incorporate the full specification of requirements.

Figure 12 shows an initial class diagram for the first spiral of the Military Ship PO.

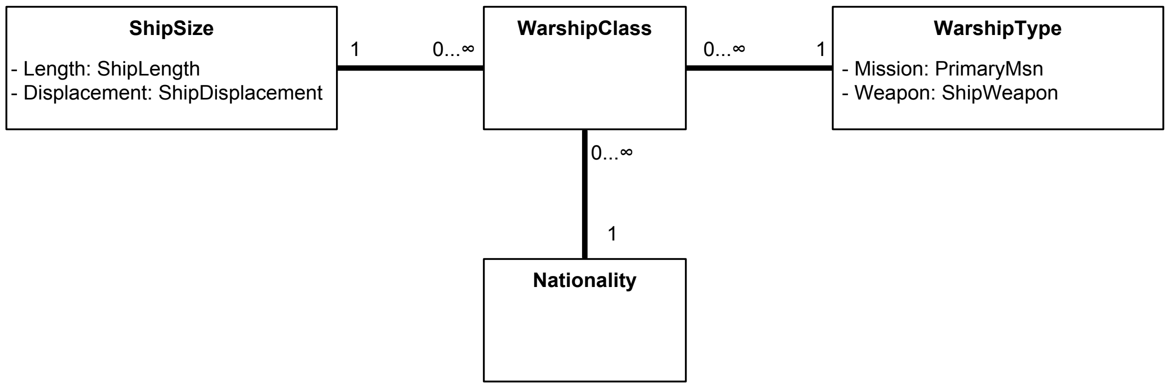


Figure 12 - MilShip PO Initial Class Diagram

At this point, known related classes may be included as attributes of Tier-one Attribute classes to clarify how this class diagram is to be expanded.

4.3.2 Ontology Development Activity

An ontology is used to capture consensual knowledge about a domain of interest [56]. The Ontology Development activity summarizes the non-trivial ontological engineering tasks required to produce a working ontology, shown in Figure 13. As discussed in Section 3.2.2, selection of the appropriate ontological engineering methodology is context dependent as is the required fidelity of the ontological model. However, there are tasks and products common to each of these processes summarized in Figure 13 and discussed below. Again, the Military Ship PO is used to provide relevant context as an example.

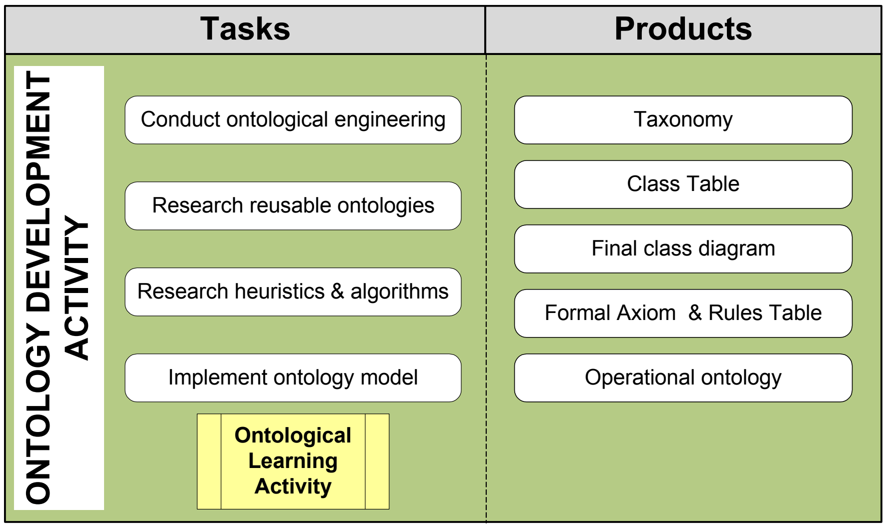

Tasks
Products
ONTOLOGY DEVELOPMENT ACTIVITY
Conduct ontological engineering
Research reusable ontologies
Research heuristics & algorithms
Implement ontology model
Ontological Learning Activity
Taxonomy
Class Table
Final class diagram
Formal Axiom & Rules Table
Operational ontology

Figure 13 - Ontology Development Activity

4.3.2.1 Ontology Development Activity Tasks

4.3.2.1.1 Conduct Ontological Engineering

Gomez-Perez et al. define ontological engineering as the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools, and languages for building ontologies [56]. With a clear understanding of the spiral objective and requirements, ontology construction begins following one of the community-accepted ontological development processes described by Gomez-Perez et al. [56]. Each of these methodologies includes both management and development activities to produce ontologies from conceptualization to implementation. In general, the proposed methods identify the tasks that need to be performed without regard for the order in which they are completed. The goal of the Ontology Development

Activity is to produce a working ontology that accurately represents the relationships of importance, focusing on the Prime Queries.

Terms and processes for development are as various as the application for which they are used. A generalized sequence of steps iteratively modeled for ontological engineering is proposed below as:

Ontological Engineering Process

i. **Identify Classes:** *what objects are acting or acted upon?*

ii. **Develop Context:** *where or when are the actions occurring?*

iii. **Identify Relationships:** *what objects are affected by an object?*

iv. **Identify States:** *in what condition may an object be found?*

Many of these steps are initialized in the Frame Activity and are continued here in the Ontology Development Activity. Through continuous refinement, classes, context, relationships, and states will be fully identified and ready for modeling within an appropriate software package.

4.3.2.1.2 Research Usable Ontologies

Before beginning construction of the ontology, it is useful to research existing ontologies in related domains to be reused and/or extended for the current problem, as discussed in Section 3.2.2.3. Model reuse is defined as the process by which available knowledge is used as input to generate new models. Reusing existing models may also require ontological re-engineering as described by Gomez-Perez et al. [56]. Similarly, ontology merging is a process by which a unique ontology is derived from two or more existing ontologies. There is an existing and ever-increasing body of knowledge regarding model reuse and extension that is beyond the scope of this work and includes methods such as ONIONS, FCA-Merge, and PROMPT. The interested reader may find these in Gomez-Perez et al. [56].

4.3.2.1.3 Research Heuristics and Algorithms

A heuristic is an experience-based technique for problem solving, learning, and discovery; an algorithm is a stepwise procedure for calculation of a problem solution. Heuristics and algorithms are used to express relationships between classes and individuals within ontologies and probabilistic ontologies. For the Ontology Development Activity, these heuristics and algorithms are used as bounding constraints to scope the model appropriately for the domain by capturing plain-language relationship statements in machine-readable format. Relevant heuristics and algorithms are regarded as Axioms which are propositions assumed without proof for the sake of studying the consequences that follow from it [41]. They are captured in a Formal Axiom & Rules Table for incorporation into the spiral model.

4.3.2.1.4 Implement Ontology Model

At this point the ontology is implemented in a suitable ontology building environment and evaluated for consistency using an appropriate evaluation methodology from the literature. Construction tools such as Protégé, Ontolingua, and OntoEdit aid in the key tasks of ontology implementation, consistency checking, and documentation [56]. For this project, the Protégé (Version 4.1) ontology development environment is used to capture military ship domain information [154]. This software is based on the Frames construct and supports first-order logic reasoning [56].

4.3.2.1.5 Ontological Learning Activity

There are several methods to aid in the knowledge acquisition process required to build an ontology. These include, but are not limited to ontological learning from texts, ontological learning from instances, ontological learning from schemata, and ontological learning for interoperability. Use of these techniques may aid in ontology development, as introduced in Section 3.2.2.5. The interested reader will also find further information on these topics in Gomez-Perez et al. [56].

4.3.2.2 Ontology Development Activity Products

The Ontology Development activity produces five products used to perform the Probability Incorporation Activity of the PODM, as shown in Figure 13 above. While completion of the PODM is feasible without these products, they provide significant aid in the documentation of the PO and reduce the likelihood of error during the iterative PO Construction Process.

4.3.2.2.1 Taxonomy and Relationships

A taxonomy is used to organize entity classes and instances through a hierarchical framework based on shared characteristics and serves as a baseline blueprint for the ontology framework. Objects are ordered into classes to define attribute inheritance and inter-class relationships. For this application, the taxonomy captures a complete dictionary of the actors in a natural relationship format. The taxonomy for the Military Ship Ontology is shown below in Figure 14.

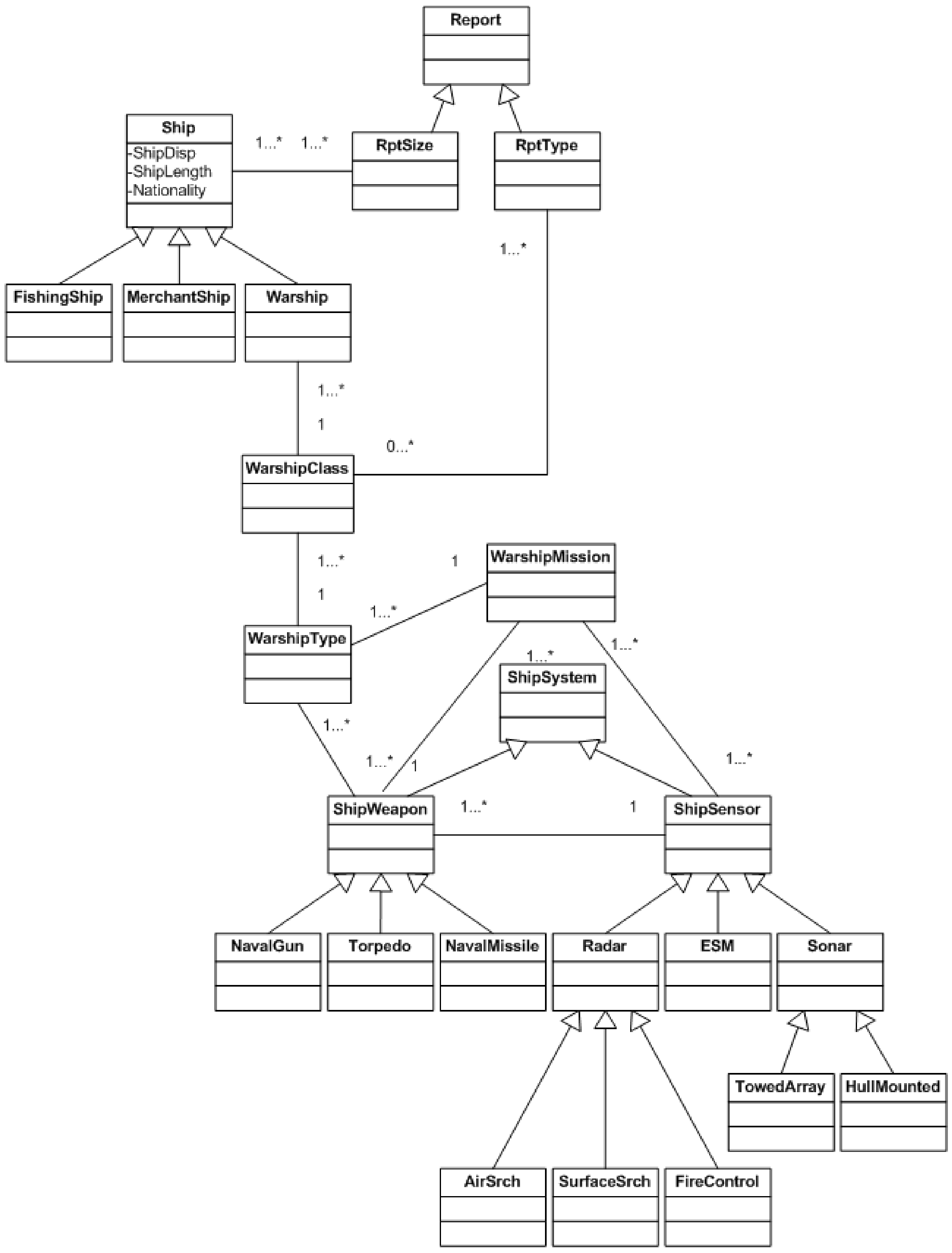
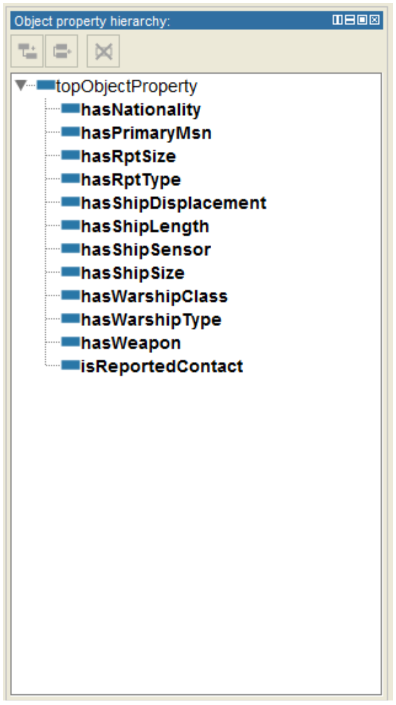


Figure 14 - MilShip Ontology Taxonomy

In the Military Ship domain, a warship is a type of Ship and inherits attributes of Displacement, Length, and Nationality. Ships have two types of systems, Weapons and Sensors. Weapons include Naval Guns, Torpedoes, and Naval Missiles. Sensors include several types of radar (Air Search, Surface Search, and Fire Control) and two types of Sonar, Towed-Array and Hull-Mounted. Electronic Support Measures are not included in the example model and is an area of future expansion to the Military Ship Ontology.



```
topObjectProperty
    hasNationality
    hasPrimaryMsn
    hasRptSize
    hasRptType
    hasShipDisplacement
    hasShipLength
    hasShipSensor
    hasShipSize
    hasWarshipClass
    hasWarshipType
    hasWeapon
    isReportedContact
```

Figure 15 - Object Properties (Relationships) for MilShip Ontology in Protege

Relationships between classes in the ontology are described by object properties, examples of which are shown in Figure 15 for the MilShip Ontology. In particular, the object properties `hasNationality`, `hasShipSize`, and `hasWarshipType` are used to infer the Prime Query, represented by `hasWarshipClass`. The other object properties illustrated in Figure 15 describe the relationships visible in the completed class diagram, Figure 16. Domain and Range information for object properties and data properties are compiled in the Class Table (Table 7).

4.3.2.2.2 Class Table

The Class Table captures the attributes and relations that describe all of the classes in the ontology. For each class the object properties, data properties, and their associated relations, domains, and ranges are collected. This compilation is used as a ready-reference throughout the ontological engineering process and aids the developer in maintaining consistency. The Class Table for the Military Ship Ontology is given in Table 7.

Table 7 - MilShip Ontology Class Table

Class Table				
Class	Associated Object Property	Associated Data Property	Domain	Range
Nation	hasNationality		WarshipClass	Nation
Report	isReportedContact		Ship	Report
Ship	hasNationality isReportedContact		Ship Ship	Nation Report
ShipSensor	hasReqSensor hasCueingSensor		WarshipMission ShipWeapon	ShipSensor ShipSensor
ShipSize	 hasRptSize hasShipSize	hasShipDisplacement hasShipLength	Ship Ship ShipSize WarshipClass	Float Float SizeReport ShipSize
ShipWeapon	hasWeapon hasReqWpn		WarshipType WarshipMission	ShipWeapon ShipWeapon
SizeRpt	hasRptSize		ShipSize	SizeReport
TypeRpt	hasRptType		WarshipType	TypeReport
Warship	hasWarshipClass		Warship	WarshipClass
WarshipClass	hasShipSize hasNationality hasWarshipType hasWarshipClass		WarshipClass WarshipClass WarshipClass Warship	ShipSize Nation WarshipType WarshipClass
WarshipMission	hasReqSensor hasPrimaryMsn hasReqWpn		WarshipMission WarshipType WarshipMission	ShipSensor WarshipMission ShipWeapon
WarshipType	hasWarshipType hasRptType hasWeapon hasPrimaryMsn		WarshipClass WarshipType WarshipType WarshipType	WarshipType TypeReport ShipWeapon WarshipMission

Alignment between Figures 14 and 16, and Table 7 indicate consistency in the model. A concise Class Table allows implementation of the ontology in one of several ontological packages introduced previously.

4.3.2.2.3 Complete Class Diagram

Class diagrams are a mainstay of object-oriented analysis and design. They identify the hierarchy of variables germane to the model. Relationships between variables that could cause another variable to change states are highlighted to capture causality between classes. Typically class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. The figure below completes the class diagram for the Military Ship Ontology.

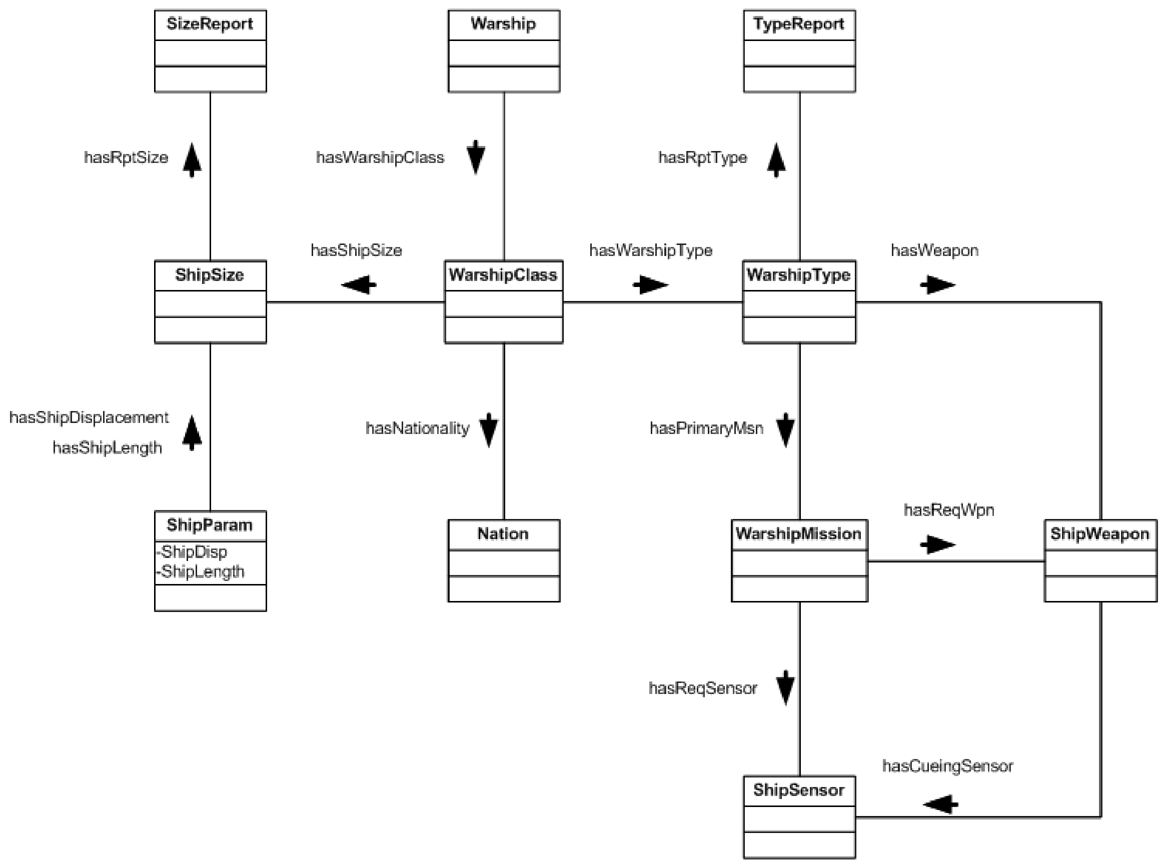


Figure 16 - MilShip Ontology Completed Class Diagram

4.3.2.2.4 Formal Axioms and Rules Table

Formal Axioms are first-order logical expressions that are always true. Rules are used to infer attribute values, or relation instances [56]. The Formal Axioms and Rules

Table also captures heuristics and algorithms that act as constraints for the model. Some rules for the Military Ship PO are shown in Table 8, below.

Table 8 - MilShip Ontology Formal Axioms & Rules

Formal Axioms & Rules Table					
Axiom	**Cueing**	**Mission**	**Configuration**	**Sensor**	**Weapon**
Description	A weapon is cued by a single sensor	A warship type is designed for a single primary mission	A warship type carries standard weapons	A mission requires specific sensors	A mission requires a specific weapon
Expression	NA	NA	NA	NA	NA
Classes	Ship Weapon Ship Sensor	Warship Type Warship Mission	Warship Type Ship Weapon	Warship Mission Ship Sensor	Warship Mission Ship Weapon
Relations	hasCueingSensor	hasPrimaryMsn	hasWeapon	hasReqSensor	hasReqWpn
Variables	NA	NA	NA	NA	NA

The entries in this table transform plain language constraints into formal, machine processable form. For example, in column 2 the rule, "A warship type is designed for a single primary mission," indicates a cardinality constraint equal to:

```
WarshipType 1…∞ -- 1 Mission
```

This constraint affects Warship Type, Warship, and Primary Mission classes through the `hasPrimaryMsn` object variable.

4.3.2.2.5 Operational Ontology

Finally, the operational ontology is created and is ready for evaluation. When seeking to answer a query about a specific domain of interest, the ontology can be considered a compilation of vocabulary and concepts used to frame the related entities. Recall from Gruber,

"An ontology is an explicit specification of a conceptualization [60]."

The working ontology serves as the relational framework for the PO when uncertainty is introduced. Construction tools and environments such as Protégé (Figure 17) aid in the key ontological engineering tasks of implementation, consistency checking, and documentation.

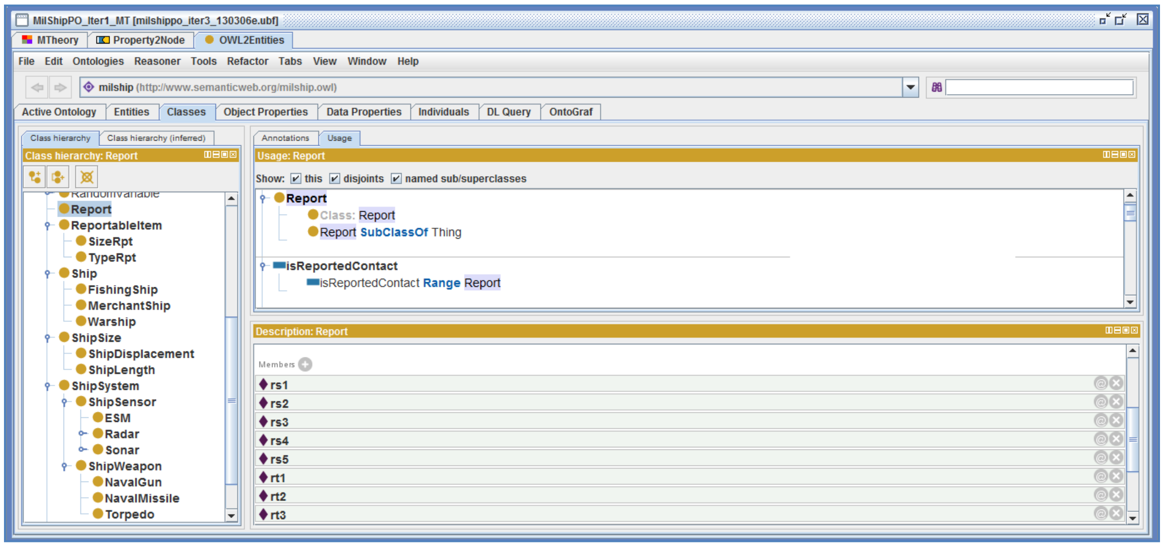


Figure 17 - Operational MilShip Ontology in Protégé

4.3.3 Probability Incorporation Activity

The Probability Incorporation Activity is the heart of the PODM, illustrated in Figure 18. Each spiral of the development cycle begins with creation of a Spiral Core Model based on the Prime Queries and their Tier-one attributes, as shown in the figure.

The Spiral Core Model is the keystone of development and is evaluated for correct operation and logic before the model is expanded to include additional attributes. After the Spiral Core Model generation tasks, the iterative PO Construction Process systematically decomposes each of the primary, secondary and tertiary attributes, evaluating model logic at each step. The spiral plan for the first Primary Query of the Military Ship PO is given in Figure 19.

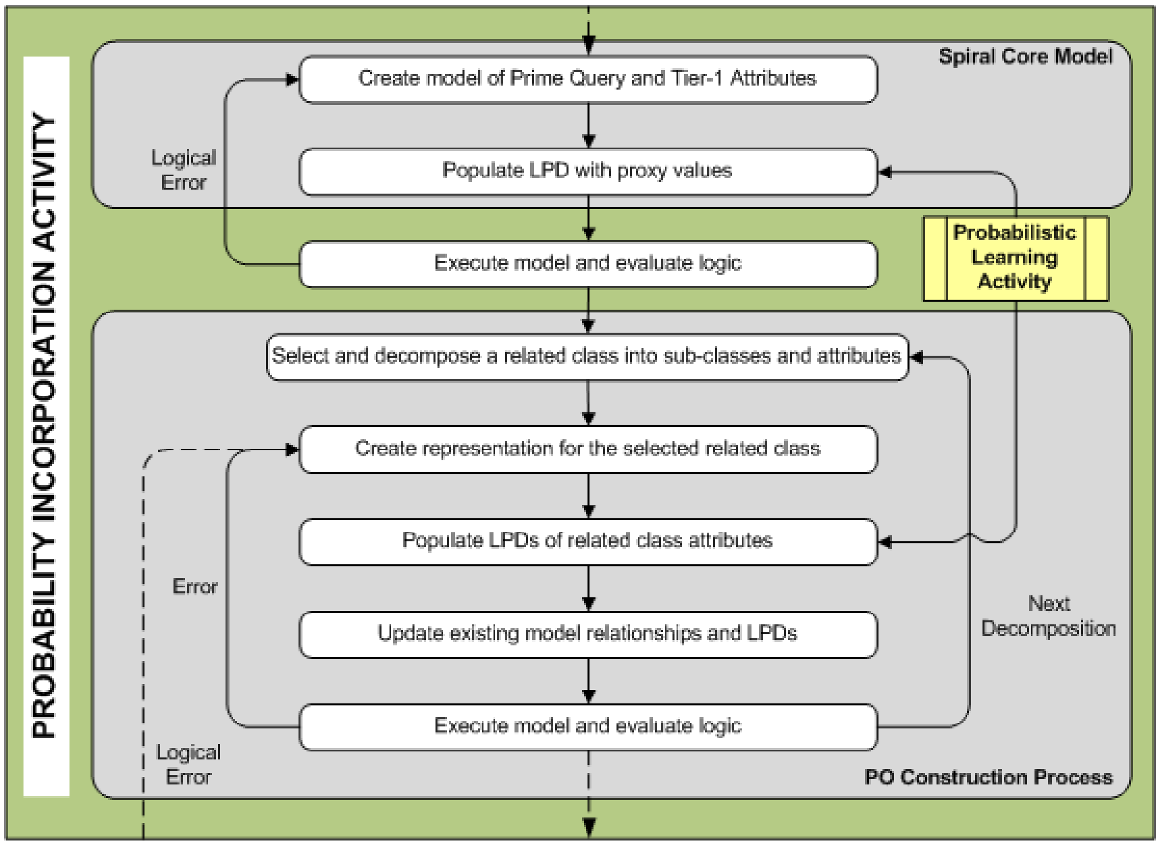

PROBABILITY INCORPORATION ACTIVITY
Spiral Core Model
Create model of Prime Query and Tier-1 Attributes
Logical Error
Populate LPD with proxy values
Probabilistic Learning Activity
Execute model and evaluate logic
Select and decompose a related class into sub-classes and attributes
Create representation for the selected related class
Populate LPDs of related class attributes
Error
Next Decomposition
Update existing model relationships and LPDs
Execute model and evaluate logic
Logical Error
PO Construction Process

Figure 18 - Probability Incorporation Activity and PO Construction Process

Multi-Entity Bayesian Networks is the modeling language used for the Military Ship PO, captured in the UnBBayes software tool. A MEBN represents knowledge about attributes of entities and their relationships as a collection of similar hypotheses organized into theories, which satisfy consistency constraints ensuring a unique joint probability distribution over the random variables of interest **[29]**. MEBN Theories can represent uncertainty about values of n-ary function and relations. UnBBayes is open source software for modeling, learning and reasoning upon probabilistic networks **[159][108][23]**. In the following sections, illustrations of appropriate MEBN components captured in the UnBBayes software tool are provided for clarification.

4.3.3.1 Core Model Generation

The initial PODM steps for incorporation of probability set the framework for the complete model and establish the spiral Prime Queries that will be serviced through the inferential reasoning model. The Core Model is established based on attributes immediately affecting the spiral Prime Queries, referred to as Tier-one attributes. Next, the Local Probability Distributions (LPDs) of this Core Model are populated with proxy probabilities that test the logic of the model without creating all of the related branch nodes resident in the final model. The proxy probabilities should be representative of expected posterior likelihoods delivered by individual Bayesian network branches under a completed model. The model is then executed and the logic examined. At this stage of development, errors are usually caused by illogical LPD values since the model architecture is quite simple. Establishing a strong foundation in this fashion eases debugging when the complexity of the model increases.

The PO Construction Process Iteration Plan (Figure 19) shows how the Spiral Core Model for the first spiral of the development process for the Military Ship PO is expanded to satisfy the Objective Statement. The first iteration of the PO Construction Process introduces uncertainty associated with the arrival of reports about the size of the contact of interest. Next, Iteration 2 expands the model to include uncertainty from the arrival of reports about the type of warship observed. Finally, Iteration 3 adds detail based on the sensors and weapons required to complete the Primary Mission tasked to a given class.

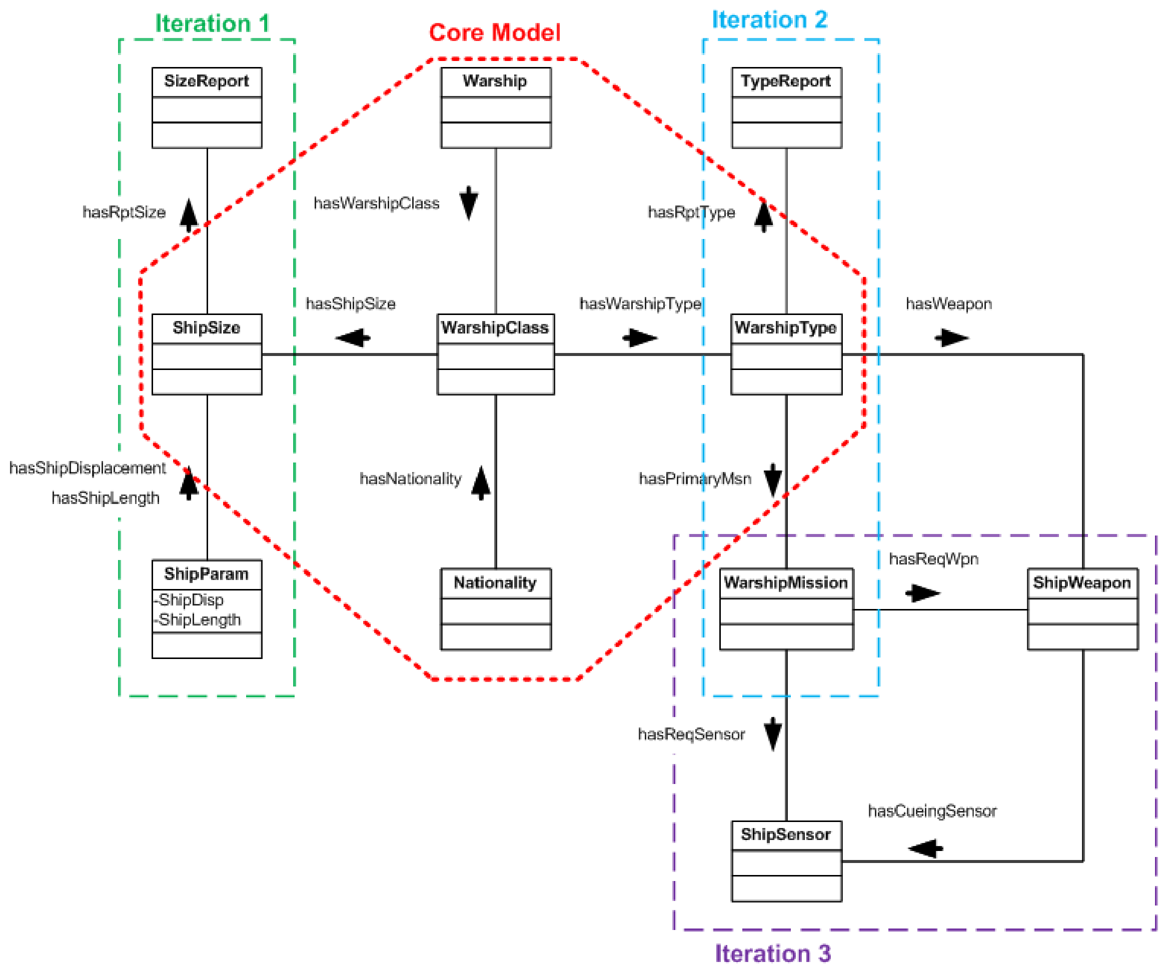


Figure 19 – PO Construction Process Iteration Plan

Upon completion of the third iteration of the PO Construction Process, a detailed model of the first spiral is complete which provides an inferred solution to the first spiral Prime Queries in the face of uncertainty.

4.3.3.1.1 Create Model of Prime Queries and Tier-one Attributes

The spiral Prime Query model is populated with conditional probabilities based on the Tier-one attribute relationships. LPDs for Prime Query nodes should have actual conditional probabilities. However, the Tier-one attribute LPDs use proxy values allowing testing of the Spiral Core Model logic before the model is iterated in the PO Construction Process.

The first two steps of the Probability Incorporation Activity produce the Spiral Core Core Model, illustrated in

Figure 12 and captured in the simple MEBN below (Figure 20).

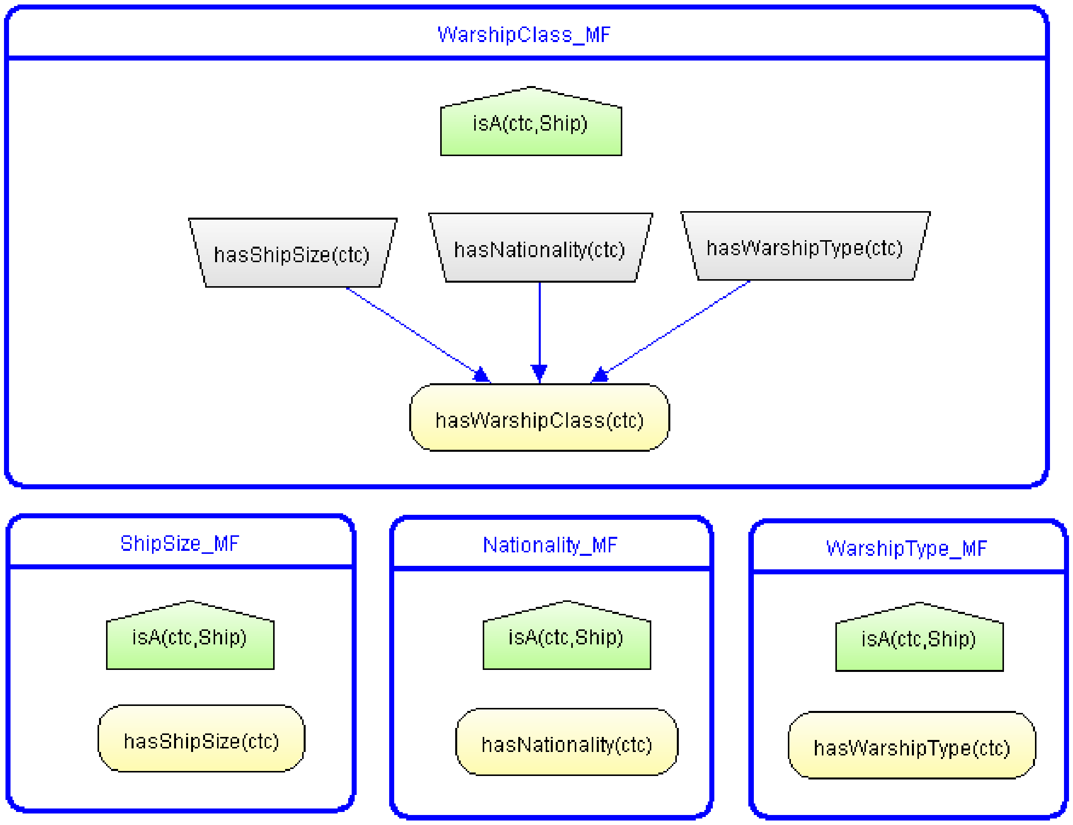


Figure 20 - MilShip PO Core MEBN Model

Each of the Tier-one Attributes of the Prime Query is represented by a MEBN Fragment (MFrag), and provides input to the Prime Query MFrag, `WarshipClass`.

4.3.3.1.2 Populate LPD with Proxy Values

The LPD of the Tier-one attribute nodes are populated with values that allow testing of the core model before all of the relationships are in place. The Tier-one LPDs use proxy values representing full network connectivity, and the Prime Query LPD is populated with appropriate conditional probabilities.

```
Warship Type
[
   Type_FFG = 0.40,
   Type_FF = 0.30,
   Type_CVN = 0.05,
   Type_Other = 0.25
]
```

```
Warship Nationality
[
   Nation_ES = 0.30,
   Nation_FR = 0.30,
   Nation_DE = 0.25,
   Nation_Other = 0.15
]
```

```
Warship Size
[
   SizeSmall = 0.60,
   SizeMedium = 0.35,
   SizeLarge = 0.05
]
```

Figure 21 - MilShip PO Core Model Proxy LPDs

Proxy values for the Tier-one Attributes are detailed in Figure 21.A complete listing of the Military Ship PO first Spiral Core Model LPD may be found in Appendix A.

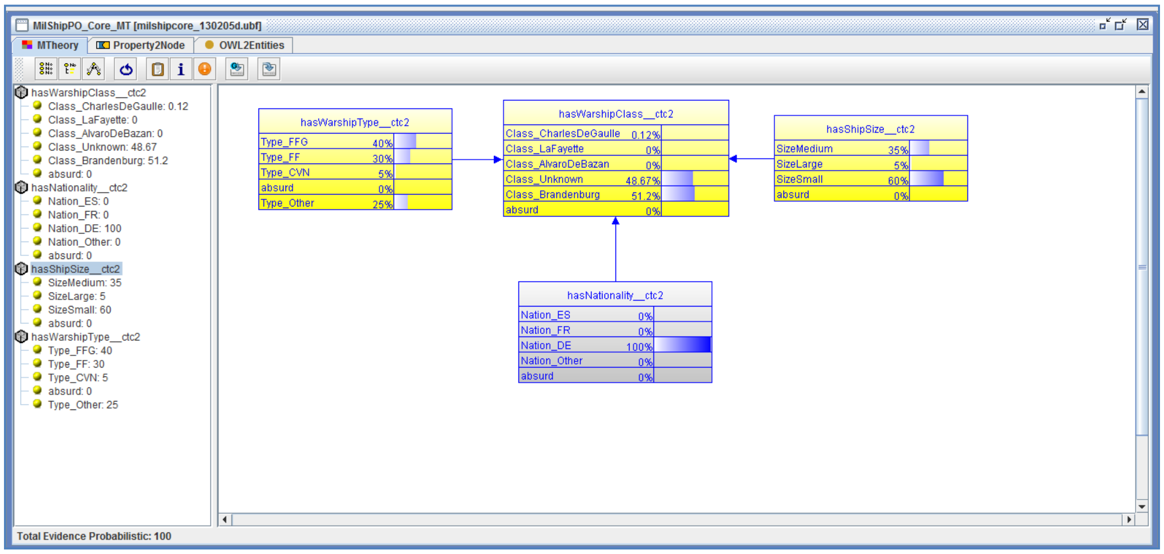
4.3.3.1.3 Execute Model and Evaluate Logic

Using the software tool, the model is executed with test evidence, an example of which is given in Table 9. It is useful to create a simple model using a Bayesian network software package to compare testing values. The simple examples used in the tutorial show that the Spiral Core Model logic operates correctly, and the knowledge engineer can be confident in moving forward with the PO Construction Process.

Table 9 - Evidence Table 1

Evidence Table 1	
Variable	**Evidence**
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE

Figure 22 shows the UnBBayes MEBN SSBN created when the Prime Query (`hasWarshipClass`) is evaluated for Contact 2 (`ctc2`).



```
hasWarshipClass__ctc2
    Class_CharlesDeGaulle: 0.12
    Class_LaFayette: 0
    Class_AlvaroDeBazan: 0
    Class_Unknown: 48.67
    Class_Brandenburg: 51.2
    absurd: 0
hasNationality__ctc2
    Nation_ES: 0
    Nation_FR: 0
    Nation_DE: 100
    Nation_Other: 0
    absurd: 0
hasShipSize__ctc2
    SizeMedium: 35
    SizeLarge: 5
    SizeSmall: 60
    absurd: 0
hasWarshipType__ctc2
    Type_FFG: 40
    Type_FF: 30
    Type_CVN: 5
    absurd: 0
    Type_Other: 25
```

```
hasWarshipType__ctc2
Type_FFG       40%
Type_FF        30%
Type_CVN        5%
absurd          0%
Type_Other     25%

hasWarshipClass__ctc2
Class_CharlesDeGaulle   0.12%
Class_LaFayette            0%
Class_AlvaroDeBazan        0%
Class_Unknown          48.67%
Class_Brandenburg       51.2%
absurd                     0%

hasShipSize__ctc2
SizeMedium     35%
SizeLarge       5%
SizeSmall      60%
absurd          0%

hasNationality__ctc2
Nation_ES       0%
Nation_FR       0%
Nation_DE     100%
Nation_Other    0%
absurd          0%
```

```
Total Evidence Probabilistic: 100
```

Figure 22 - SSBN of Core Model

The Bayesian network in Figure 23 was created using the Netica [115] software tool and verifies the logic of the simple SSBN. This process of querying the probabilistic ontology model and evaluating the model continues throughout the iterative development process. Test cases for the three model iterations are given in Appendix A.

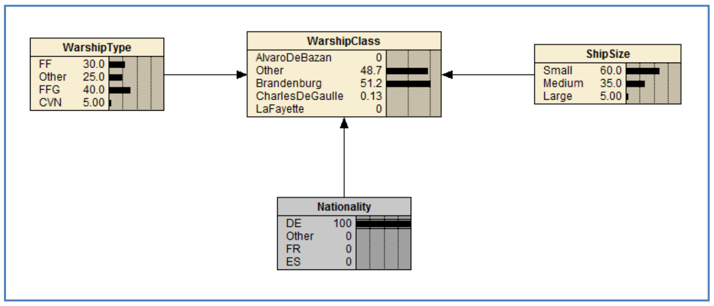


Figure 23 - Netica Bayesian Network to Test Core Model

4.3.3.1.4 Probabilistic Learning Activity

Probabilistic learning uses a relational schema assumed by the classes, their attributes, and relationships between classes to reduce the effort involved in establishing prior and conditional probabilities for domain entities. A training set in the form of a relational database of empirical data is utilized and the LPD parameters are determined using the likelihood function and an appropriate algorithm (e.g. Maximum Likelihood Parameter Estimation or Bayesian Parameter Estimation) [52]. An introduction to this topic is included in Section 3.2.2.5.

4.3.3.2 Probabilistic Ontology Construction Process

The iterative steps follow the PO Construction Process, shown in Figure 18 above, to systematically expand the initial model while ensuring coherent logic is maintained. In each iteration, a related class is selected and decomposed into its immediate sub-classes and their attributes. A representation is created for the related class, and LPDs are populated with proxy probabilities for the attributes if the node is non-terminal. Otherwise, appropriate prior probabilities are established in the LPD through research or Subject-Matter Expert elicitation. Next, the model created in previous steps is updated to include relationships with the newly created representation, and the LPDs are updated to capture any probabilistic relationships. The final step in the iteration is to execute the model and evaluate its logic using example data. Logic errors at this stage are likely caused by oversights and errors in the update of the existing model. When the executed model produces the desired logic, the next iteration is begun. The PO Construction Process is summarized as:

PO Construction Process

i. Select and decompose a related class into its sub-classes and attributes
ii. Create a representation for the selected related class
iii. Populate LPDs of related class attributes
iv. Update existing model relationships and LPDs
v. Execute model and evaluate logic

4.3.3.2.1 Select and Decompose a Related Class into its Sub-classes and Attributes

One of the related classes identified from the class diagram is decomposed into its subcomponents, classes and attributes. The sub-classes should be decomposed to the lowest possible level. The illustration below (Figure 24) shows the decomposition of the `Ship Size` class of the Military Ship PO to incorporate `Reports` about the size of the contact of interest as well as variables `for Ship Displacement` and `Ship Length`.

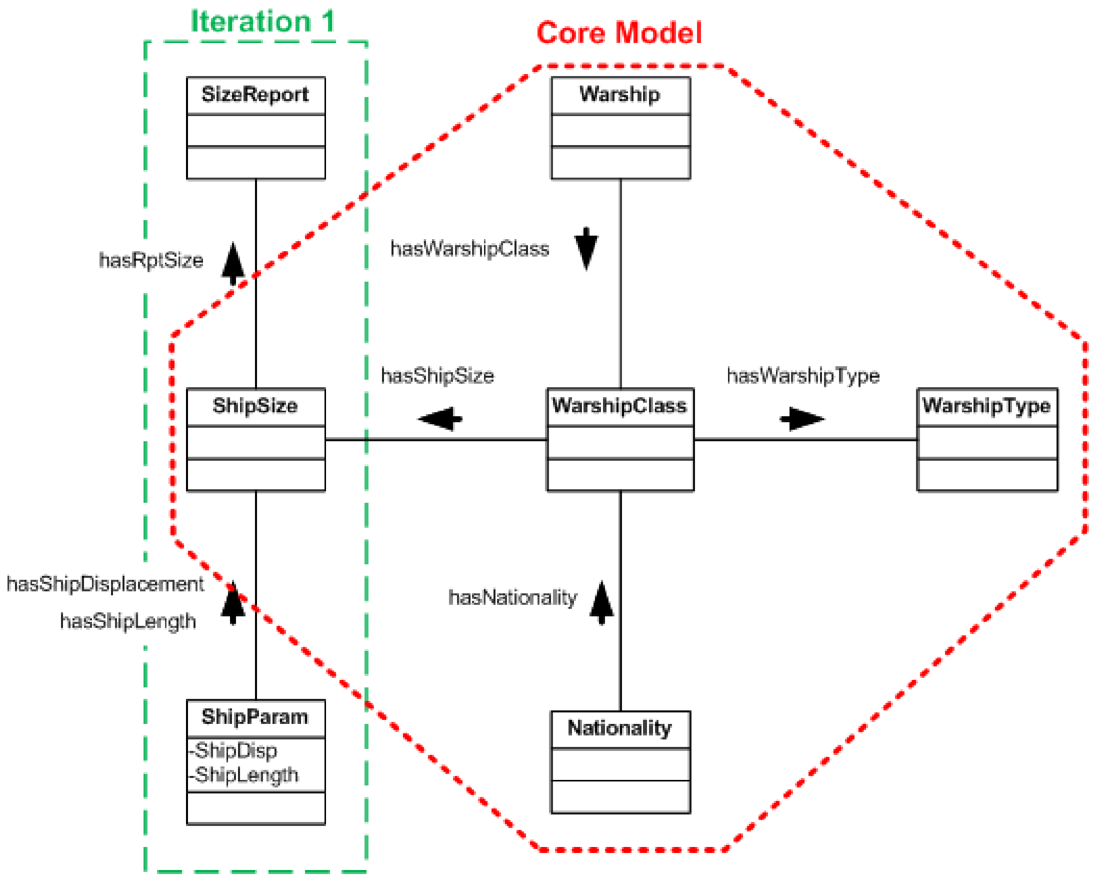


Figure 24 - Sub--class Decomposition: Iteration 1

Ideally, the displacement and length variables should be of type float to accommodate appropriate metric system values of ship classes. However, the current release of the UnBBayes software package requires individuals to be associated with object properties of classes. Therefore, both displacement and length have been binned into three categories (small, medium, large) that capture the differences in features of the classes within the library.

4.3.3.2.2 Create a Representation for the Selected Related Class

Using the information assembled in the decomposition, build a representation for the new class. Addition of a class influences one or more existing classes. Summarizing the relevant properties, domain, range and variables simplifies production of the model and discovery model logic.

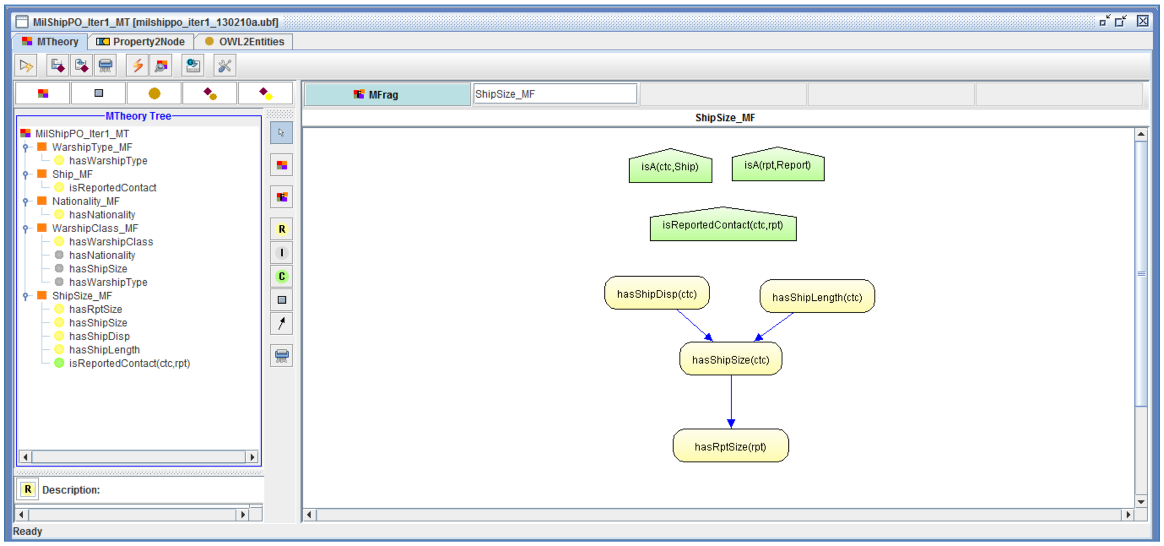


Figure 25 - Ship Size MFrag (Iteration 1)

In the MilShip PO, the UnBBayes MEBN MFrag for the Ship Size is shown above (Figure 25). Input Nodes for the `Ship Displacement` and `Ship Length` attributes are added to the MFrag to capture uncertainty for these properties as discussed above. Further, the actual ship size directly affects the `Reported Size` as can be seen below. The `Reported Size` is binned into the same three possible categorical states as the original `Ship Size` variable (small, medium, large).

4.3.3.2.3 Populate LPDs of Related Class Attributes

LPDs for the new nodes may be in the form of a conditional probability table

```
Reported Size
if any ctc have (hasShipSize = SizeSmall) [
   Rpt_SizeLarge = 0.10,
   Rpt_SizeMedium = 0.33,
   Rpt_SizeSmall = 0.60
] else if any ctc have (hasShipSize = SizeMedium) [
   Rpt_SizeLarge = 0.15,
   Rpt_SizeMedium = 0.60,
   Rpt_SizeSmall = 0.25
] else if any ctc have (hasShipSize = SizeLarge) [
   Rpt_SizeLarge = 0.60,
   Rpt_SizeMedium = 0.30,
   Rpt_SizeSmall = 0.10
] else
   Rpt_SizeLarge = 0.161,
   Rpt_SizeMedium = 0.475,
   Rpt_SizeSmall = 0.364
]
```

```
Ship Displacement
]
   Disp5kto10k = 0.35,
   DispLess5k = 0.60,
   DispGreater10k = 0.05
]
```

```
Ship Length
]
   LengthGreater150 = 0.05,
   Length125to150 = 0.35,
   LengthLess125 = 0.60
]
```

Figure 26 - New LPDs (Iteration 1)

(CPT) or logic statements, depending on the probabilistic ontology software utilized. If further decomposition is warranted, proxy values should be used for the nodes similar to those of the Tier-one Attributes above. Otherwise, terminal node likelihoods should be established via research or SME elicitation. In the `Ship Size` MFrag of the first Military Ship PO spiral, the `Ship Displacement` and `Ship Length` attributes are terminal data properties and have prior probabilities assigned as shown in Figure 26. The `Size` node is conditioned on these nodes and categorizes the ship into one of three possibilities based on these parameters. The `Reported Size` is a function of the actual size, but with additional uncertainty due to latency, proficiency, or equipment limitations. The LPD for `Reported Size` is also shown in Figure 26.

4.3.3.2.4 Update Existing Model Relationships and LPDs

All legacy nodes affected by addition of the new attribute must be updated to reflect conditional probabilities expressing the relationships associated with the new node. Elicitation of conditional probabilities within the LPD is accomplished through research, SME interview, or Bayesian learning techniques. A simplified Bayesian network may also aid the PO Developer in eliciting the prior values to use in this statement. The updated LPD provides a more realistic illustration of the uncertainty of judging dimensions with the addition of the size parameter nodes.

For the Military Ship PO, the Ship Size LPD is affected by the introduction of the two parameter nodes, Ship Displacement and Ship Length. Therefore, the Ship Size LPD is updated as shown in Figure 27.

```
Ship Size (updated)
if any ctc have ( hasShipDisp = DispLess5k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .10,
 SizeLarge = 0,
 SizeSmall = .90
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ]
] else if any ctc have ( hasShipDisp = Disp5kto10k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ]
] else if any ctc have ( hasShipDisp = DispGreater10k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ] else [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ]
] else [
 SizeMedium = .582,
 SizeLarge = .0639,
 SizeSmall = .354
]
```

Figure 27 - Updated Ship Size LPD (Iteration 1)

The updated MTheory for the Military Ship PO after the completion of the PO Construction Process Iteration 1 is shown in Figure 28. Note that there was no change to any of the four MFrags on the left. By focusing on a single attribute at a time, mistakes in updating logic are minimized and coherency is maintained as the model becomes increasingly complex.

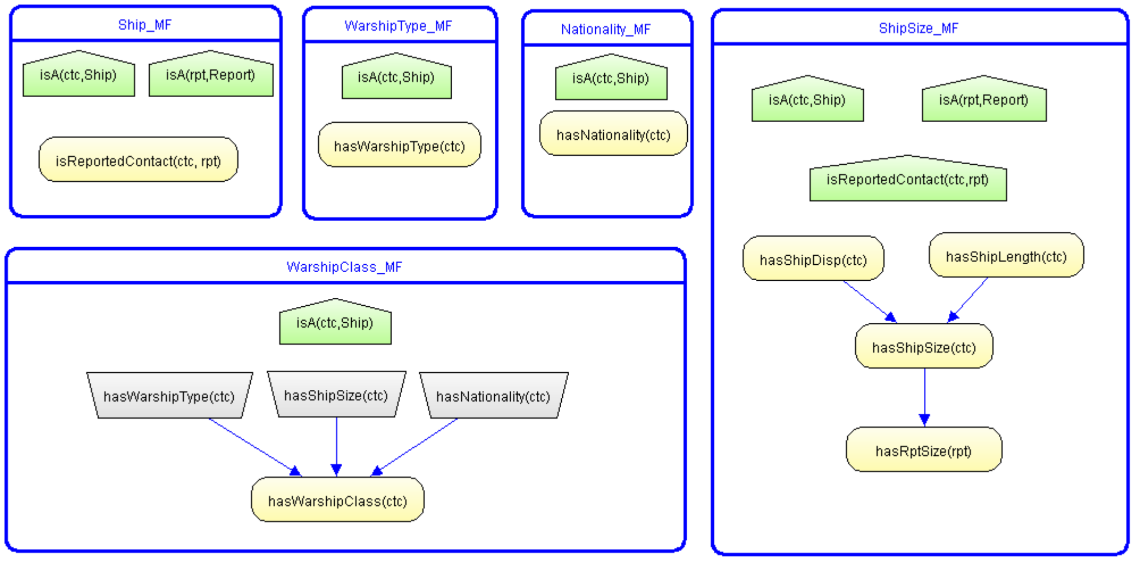


Figure 28 - Updated MilShip PO MTheory (Iteration 1)

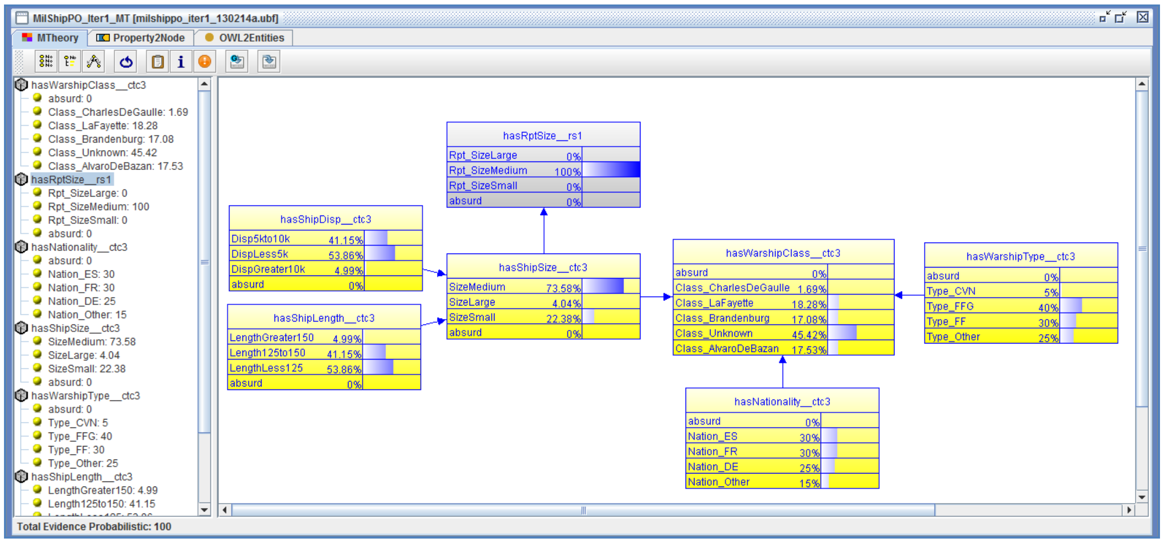
4.3.3.2.5 Execute the Model and Evaluate Logic

After each iteration of the PO Construction Process is complete, the model is executed to produce an inferred solution to a Prime Query. A simple test is devised that introduces evidence to the updated relationships to ensure model logic was maintained through the update. When feasible, it is advantageous to compare an instantiation with the appropriate Bayesian network model, as previously discussed. Table 10 is an expansion of the Evidence Table given above (Table 9) that includes a `Size Report` (`rs1`) and a new `Contact` associated with that report (`ctc3, rs1`).

Table 10 - Evidence Table 2

Evidence Table 2	
Variable	**Evidence**
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true

A simple test case should be sufficient to ensure that the logic is performing as expected; more elaborate test cases are used in the Evaluation Activity. Figure 29 is the SSBN created for the Prime Query using `rs1` and `ctc3` as input evidence.



hasWarshipClass__ctc3
- absurd: 0
- Class_CharlesDeGaulle: 1.69
- Class_LaFayette: 18.28
- Class_Brandenburg: 17.08
- Class_Unknown: 45.42
- Class_AlvaroDeBazan: 17.53

hasRptSize__rs1
- Rpt_SizeLarge: 0
- Rpt_SizeMedium: 100
- Rpt_SizeSmall: 0
- absurd: 0

hasNationality__ctc3
- absurd: 0
- Nation_ES: 30
- Nation_FR: 30
- Nation_DE: 25
- Nation_Other: 15

hasShipSize__ctc3
- SizeMedium: 73.58
- SizeLarge: 4.04
- SizeSmall: 22.38
- absurd: 0

hasWarshipType__ctc3
- absurd: 0
- Type_CVN: 5
- Type_FFG: 40
- Type_FF: 30
- Type_Other: 25

hasShipLength__ctc3
- LengthGreater150: 4.99
- Length125to150: 41.15

hasRptSize__rs1	
Rpt_SizeLarge	0%
Rpt_SizeMedium	100%
Rpt_SizeSmall	0%
absurd	0%

hasShipDisp__ctc3	
Disp5kto10k	41.15%
DispLess5k	53.86%
DispGreater10k	4.99%
absurd	0%

hasShipLength__ctc3	
LengthGreater150	4.99%
Length125to150	41.15%
LengthLess125	53.86%
absurd	0%

hasShipSize__ctc3	
SizeMedium	73.58%
SizeLarge	4.04%
SizeSmall	22.38%
absurd	0%

hasWarshipClass__ctc3	
absurd	0%
Class_CharlesDeGaulle	1.69%
Class_LaFayette	18.28%
Class_Brandenburg	17.08%
Class_Unknown	45.42%
Class_AlvaroDeBazan	17.53%

hasWarshipType__ctc3	
absurd	0%
Type_CVN	5%
Type_FFG	40%
Type_FF	30%
Type_Other	25%

hasNationality__ctc3	
absurd	0%
Nation_ES	30%
Nation_FR	30%
Nation_DE	25%
Nation_Other	15%

Total Evidence Probabilistic: 100

Figure 29 - MilShip PO Simple Test SSBN for rs1 and ctc3

The SSBN shown in Figure 29 matches the simple BN shown in Figure 30, below. Testing the different combinations of evidence demonstrates that the model logic is sound.

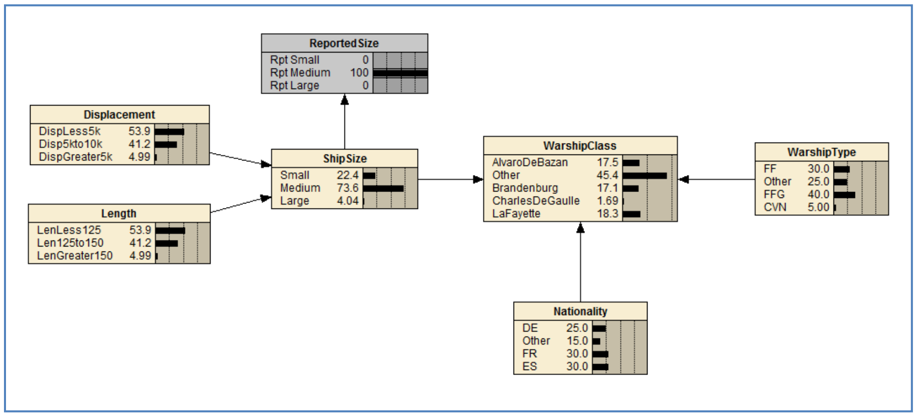


Figure 30 - Bayesian Network Test for rs1 and ctc3

4.3.3.3 Completed Military Ship Probabilistic Ontology

Three iterations of the model are conducted to complete the first spiral of the Military Ship PO as shown in Figure 19 and discussed above. The completed MTheory model is shown in Figure 31.

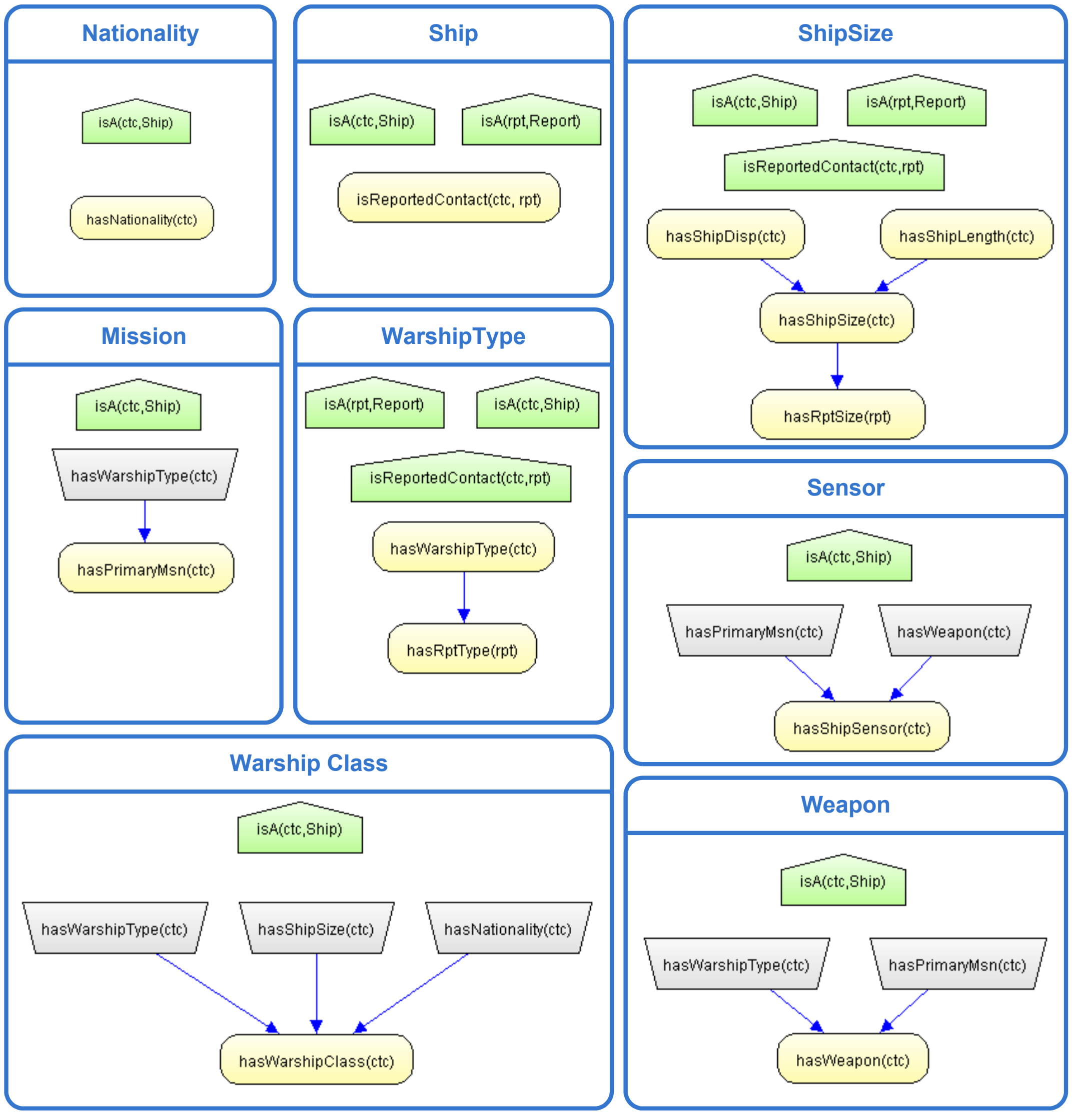


Figure 31 - Completed MilShip PO MTheory

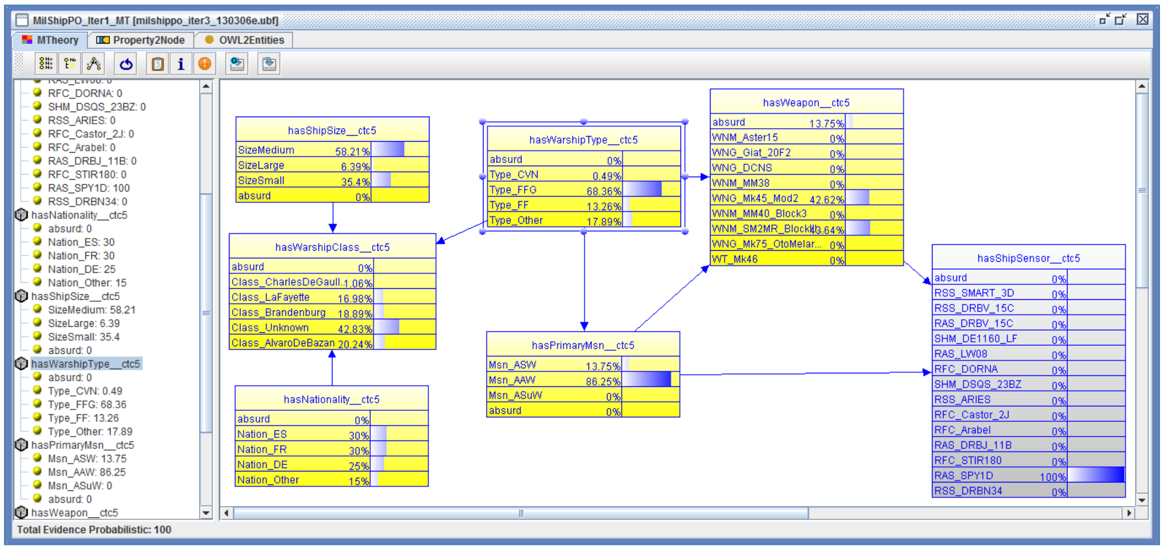
Evidence of the iterative process is clear in the above figure. First, note that hasShipSize is conditional on hasShipDisp and hasShipLength, which correspond to input parameters. hasRptSize is conditional on hasShipSize indicating that a reported size is dependent on actual size. Next, hasRptType is conditional on hasWarshipType, demonstrating that a type report is affected by the actual type of warship observed. Similarly, hasPrimaryMsn is conditional on hasWarshipType, indicating that certain warship types have associated likelihoods of conducting specific missions. Finally, both hasShipSensor and hasWeapon are conditional on hasPrimaryMsn, indicating that missions require specific types of weapons and their associated sensors. The hasWeapon variable is also conditional on hasWarshipType because not all types of warships carry every type of weapon.

Throughout the iterative process, the Evidence Table has continued to expand, the final version of which is given in Table 11.

Table 11 - Evidence Table after Iteration 3

Evidence Table 3	
Variable	**Evidence**
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc4, rt1	isReportedContact(ctc4(Ship), rt1(Report))=true
ctc5	hasShipSensor(ctc5(Ship))=RAS_SPY1D

Figure 32 shows the UnBBayes MEBN SSBN for Contact 5 (ctc5) that provides the final validation of model logic for the third iteration.



```
RFC_DORNA: 0
SHM_DSQS_23BZ: 0
RSS_ARIES: 0
RFC_Castor_2J: 0
RFC_Arabel: 0
RAS_DRBJ_11B: 0
RFC_STIR180: 0
RAS_SPY1D: 100
RSS_DRBN34: 0
hasNationality__ctc5
  absurd: 0
  Nation_ES: 30
  Nation_FR: 30
  Nation_DE: 25
  Nation_Other: 15
hasShipSize__ctc5
  SizeMedium: 58.21
  SizeLarge: 6.39
  SizeSmall: 35.4
  absurd: 0
hasWarshipType__ctc5
  absurd: 0
  Type_CVN: 0.49
  Type_FFG: 68.36
  Type_FF: 13.26
  Type_Other: 17.89
hasPrimaryMsn__ctc5
  Msn_ASW: 13.75
  Msn_AAW: 86.25
  Msn_ASuW: 0
  absurd: 0
hasWeapon__ctc5
Total Evidence Probabilistic: 100
```

hasShipSize__ctc5	
SizeMedium	58.21%
SizeLarge	6.39%
SizeSmall	35.4%
absurd	0%

hasWarshipType__ctc5	
absurd	0%
Type_CVN	0.49%
Type_FFG	68.36%
Type_FF	13.26%
Type_Other	17.89%

hasWeapon__ctc5	
absurd	13.75%
WNM_Aster15	0%
WNG_Giat_20F2	0%
WNG_DCNS	0%
WNM_MM38	0%
WNG_Mk45_Mod2	42.62%
WNM_MM40_Block3	0%
WNM_SM2MR_Block	43.64%
WNG_Mk75_OtoMelar...	0%
WT_Mk46	0%

hasWarshipClass__ctc5	
absurd	0%
Class_CharlesDeGaull...	1.06%
Class_LaFayette	16.98%
Class_Brandenburg	18.89%
Class_Unknown	42.83%
Class_AlvaroDeBazan	20.24%

hasPrimaryMsn__ctc5	
Msn_ASW	13.75%
Msn_AAW	86.25%
Msn_ASuW	0%
absurd	0%

hasNationality__ctc5	
absurd	0%
Nation_ES	30%
Nation_FR	30%
Nation_DE	25%
Nation_Other	15%

hasShipSensor__ctc5	
absurd	0%
RSS_SMART_3D	0%
RSS_DRBV_15C	0%
RAS_DRBV_15C	0%
SHM_DE1160_LF	0%
RAS_LW08	0%
RFC_DORNA	0%
SHM_DSQS_23BZ	0%
RSS_ARIES	0%
RFC_Castor_2J	0%
RFC_Arabel	0%
RAS_DRBJ_11B	0%
RFC_STIR180	0%
RAS_SPY1D	100%
RSS_DRBN34	0%

Figure 32 - SSBN for Prime Query of Contact 5

Note that inclusion of the SPY-1D Air Search Radar (`RAS_SPY1D`) increased the likelihood of the primary mission being Anti-Air Warfare (`Msn_AAW`) and the associated weapons being the Mk-45 naval gun (`WNG_Mk45_Mod2`) and the SM-2-MR Standard Surface to Air Missile (`WNM_SM2MR`). These features imply that `contact 5` is most likely of type Guided Missile Frigate (`Type_FFG`). However, without the benefit of additional evidence, it is not possible to determine the specific class (`Class_Alvaro de Bazan` or `Class_Brandenburg`), so the SSBN returns a `Class_Unknown` class. Inclusion of a nationality (ES, DE) would sway the inference toward one of these classes. The interested reader may find the entire iterative process to complete the Military Ship PO in Appendix A.

4.3.4 Evaluation Activity

The Evaluation Activity completes the PODM as shown in Figure 33 by two methods introduced in Section 3.2.2.1.4. First, an elicitation review is conducted by the PO Developer and SMEs. Then, a sequence of increasingly difficult test cases is applied to test the model across the spectrum of expected performance. Results are evaluated against existing models or by the development team. A case that results in erroneous logic is returned to the PO Construction Process at the decomposition task to rebuild the representation. A successful case is documented and followed by the next case study.

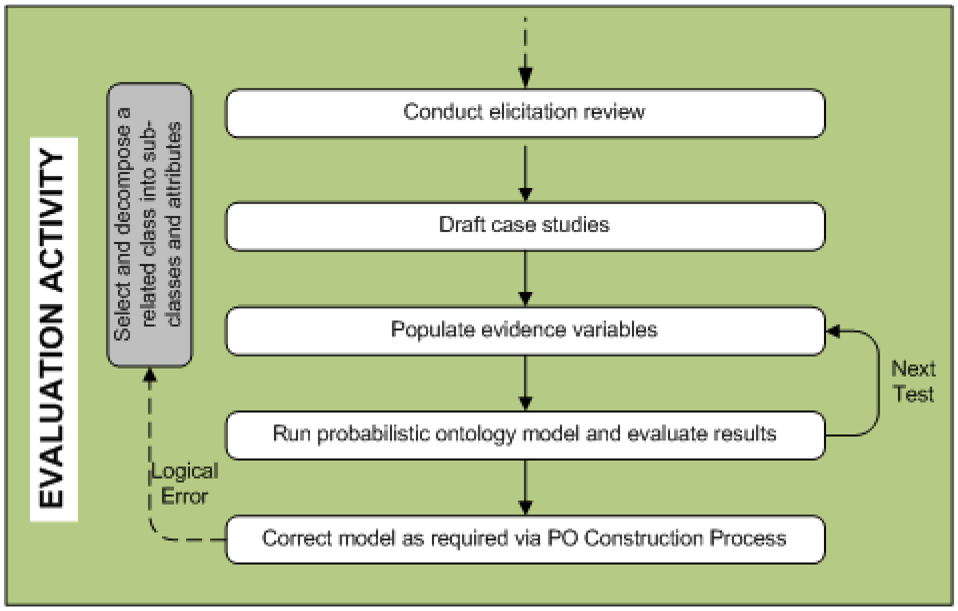

EVALUATION ACTIVITY
Select and decompose a related class into sub-classes and attributes
Conduct elicitation review
Draft case studies
Populate evidence variables
Run probabilistic ontology model and evaluate results
Next Test
Logical Error
Correct model as required via PO Construction Process

Figure 33 - Evaluation Activity

4.3.4.1 Conduct Elicitation Review

An elicitation review is a holistic review of the probabilistic ontology to ensure it is consistent with the spiral objective and Top-level Objective Statement. Laskey and Mahoney describe the elicitation review as an overall review of node definitions, state definitions, independence assumptions, and probability distributions [99]. This is a qualitative assessment provided by expert reviewers including the Stakeholder DM, SMEs, and users.

4.2.4.2 Draft Case Studies

A series of increasingly complex test cases is developed to test model logic and coherence in an operational context. Test cases are designed to test the spectrum of inference tasks expected to be encountered during operations within the Operate & Support Phase of the SDLC. The complete set of cases should fully examine the model and specify assumptions, input parameters, and expected output. Evidence collected throughout the modeling process (Table 11) and captured within the model may be useful in formulating these cases. Each test case is evaluated in the spiral PO and evaluated by expert reviewers. Deficiencies are meticulously documented to aid in model correction.

4.2.4.3 Populate Evidence Variables

For each case study, the appropriate evidence is incorporated into the PO model using FOL statements. For example, in the first case study used to evaluate the MilShip PO (below), there is evidence that contact one is reported to be a warship of medium size.

The associated evidence statement for the MilShip PO model is:

`hasRptSize(sensorReport(Report)) = Rpt_SizeMedium`.

4.2.4.4 Run PO Model and Evaluate Results

Once all of the evidence for the case is loaded into the PO KB, the Prime Queries are executed and PO results are evaluated by expert reviewers to identify potential logical or relationship errors. Cases producing incorrect results return to the PO Construction Process for refinement (Figure 18 and Figure 33). A test case that performs as expected is documented, and the next test case is applied.

4.2.5.5 Correct Model as Required via PO Construction Process

Logical and relational errors necessitate a return to PO Construction Process, as discussed above. A review of the model should identify which sub-class or attribute representation is causing the error, thereby focusing the correction. The PO Construction Process is re-run for that related class beginning with task two (Create representation for selected class) and completed through task five. Upon successful creation and evaluation of the executed model, the Evaluation Activity is resumed.

4.2.4.6 Military Ship Probabilistic Ontology Case Study Evaluation

The Military Ship PO is used throughout this section to demonstrate a stylized example of the case study evaluation process. In this series of examples, an unknown warship is detected and designated as a contact of interest (COI). The Commander (Stakeholder DM) desires to know what class of ship it represents to establish the risk it poses to his own ship.

4.2.4.6.1 Test Case 1

The baseline evidence provides little information to the Commander. Table 12 contains the evidence for Test Case 1 that is used to instantiate the first SSBN shown below.

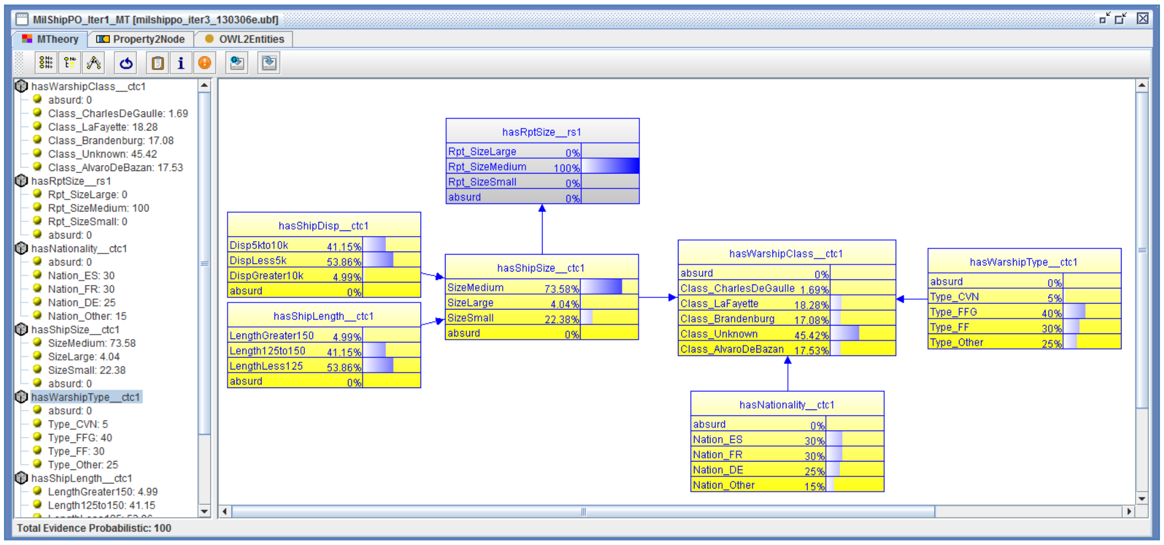
Table 12 - Evidence Table for Test Case 1

Evidence Table Case 1	
Variable	**Evidence**
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true

A summary paragraph for Test Case 1 reads:

> *There is a surface ship contact of interest (*`ctc1`*) that is reported to be a warship of medium size.*

This minimal information is captured in the SSBN depicted in Figure 34.



```
hasWarshipClass__ctc1
    absurd: 0
    Class_CharlesDeGaulle: 1.69
    Class_LaFayette: 18.28
    Class_Brandenburg: 17.08
    Class_Unknown: 45.42
    Class_AlvaroDeBazan: 17.53
hasRptSize__rs1
    Rpt_SizeLarge: 0
    Rpt_SizeMedium: 100
    Rpt_SizeSmall: 0
    absurd: 0
hasNationality__ctc1
    absurd: 0
    Nation_ES: 30
    Nation_FR: 30
    Nation_DE: 25
    Nation_Other: 15
hasShipSize__ctc1
    SizeMedium: 73.58
    SizeLarge: 4.04
    SizeSmall: 22.38
    absurd: 0
hasWarshipType__ctc1
    absurd: 0
    Type_CVN: 5
    Type_FFG: 40
    Type_FF: 30
    Type_Other: 25
hasShipLength__ctc1
    LengthGreater150: 4.99
    Length125to150: 41.15
Total Evidence Probabilistic: 100
```

hasRptSize__rs1	
Rpt_SizeLarge	0%
Rpt_SizeMedium	100%
Rpt_SizeSmall	0%
absurd	0%

hasShipDisp__ctc1	
Disp5kto10k	41.15%
DispLess5k	53.86%
DispGreater10k	4.99%
absurd	0%

hasShipLength__ctc1	
LengthGreater150	4.99%
Length125to150	41.15%
LengthLess125	53.86%
absurd	0%

hasShipSize__ctc1	
SizeMedium	73.58%
SizeLarge	4.04%
SizeSmall	22.38%
absurd	0%

hasWarshipClass__ctc1	
absurd	0%
Class_CharlesDeGaulle	1.69%
Class_LaFayette	18.28%
Class_Brandenburg	17.08%
Class_Unknown	45.42%
Class_AlvaroDeBazan	17.53%

hasWarshipType__ctc1	
absurd	0%
Type_CVN	5%
Type_FFG	40%
Type_FF	30%
Type_Other	25%

hasNationality__ctc1	
absurd	0%
Nation_ES	30%
Nation_FR	30%
Nation_DE	25%
Nation_Other	15%

Figure 34 - MilShip PO SSBN for Test Case 1

The SSBN shows that describing a contact of interest as a medium warship provides less than 20% likelihood that it is one of the identified ships in the library of interest. The heavy inference that an "Unknown" class is observed could be a class not identified in the library, or insufficient information about the known classes. Additional evidence is needed for a more accurate determination. Note that the SSBN does not explicitly show nodes (leaves) beyond `hasWarshipType`. As no evidence has been applied to these nodes, their inferential weight is rolled up into the `hasWarshipType` node.

4.2.4.6.2 Test Case 2

Test Case 2 continues the scenario with the addition of a report that the type of ship is believed to be (i.e. reported as) a guided-missile frigate (FFG). Accumulated evidence on `ctc1` is compiled in Table 13.

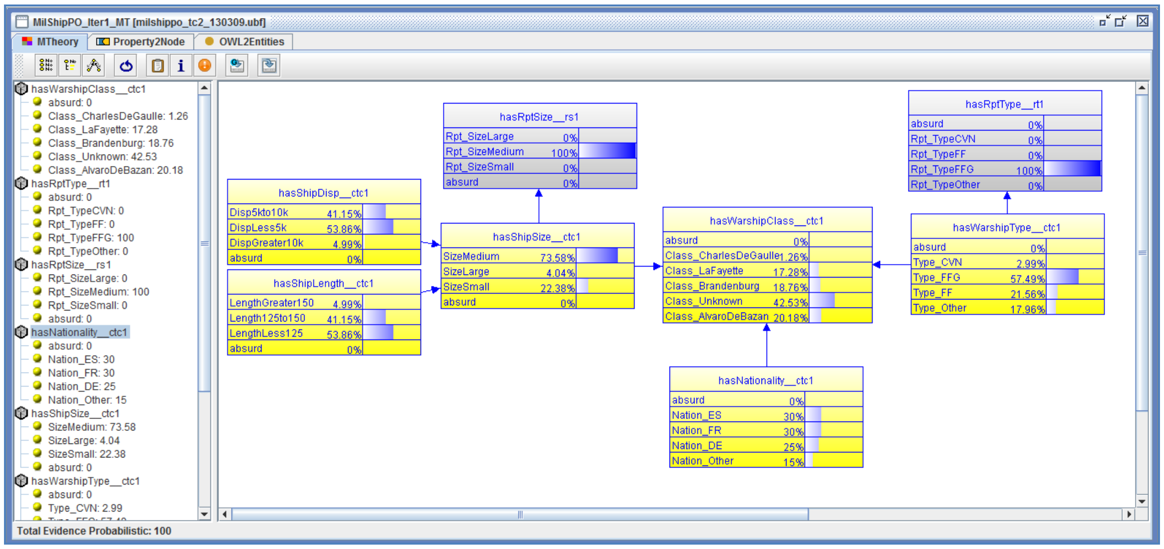
Table 13 - Evidence Table for Test Case 2

Evidence Table Case 2	
Variable	**Evidence**
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true

Test Case 2 can be summarized as:

> *The warship contact of interest (*`ctc1`*) is reported to be a medium-sized guided-missile frigate (FFG).*

This compilation of evidence to date is instantiated in an SSBN depicted in Figure 35.



```
hasWarshipClass__ctc1
    absurd: 0
    Class_CharlesDeGaulle: 1.26
    Class_LaFayette: 17.28
    Class_Brandenburg: 18.76
    Class_Unknown: 42.53
    Class_AlvaroDeBazan: 20.18
hasRptType__rt1
    absurd: 0
    Rpt_TypeCVN: 0
    Rpt_TypeFF: 0
    Rpt_TypeFFG: 100
    Rpt_TypeOther: 0
hasRptSize__rs1
    Rpt_SizeLarge: 0
    Rpt_SizeMedium: 100
    Rpt_SizeSmall: 0
    absurd: 0
hasNationality__ctc1
    absurd: 0
    Nation_ES: 30
    Nation_FR: 30
    Nation_DE: 25
    Nation_Other: 15
hasShipSize__ctc1
    SizeMedium: 73.58
    SizeLarge: 4.04
    SizeSmall: 22.38
    absurd: 0
hasWarshipType__ctc1
    absurd: 0
    Type_CVN: 2.99
Total Evidence Probabilistic: 100
```

```
hasRptSize__rs1
Rpt_SizeLarge      0%
Rpt_SizeMedium   100%
Rpt_SizeSmall      0%
absurd             0%

hasRptType__rt1
absurd             0%
Rpt_TypeCVN        0%
Rpt_TypeFF         0%
Rpt_TypeFFG      100%
Rpt_TypeOther      0%

hasShipDisp__ctc1
Disp5kto10k     41.15%
DispLess5k      53.86%
DispGreater10k   4.99%
absurd             0%

hasShipLength__ctc1
LengthGreater150  4.99%
Length125to150   41.15%
LengthLess125    53.86%
absurd              0%

hasShipSize__ctc1
SizeMedium      73.58%
SizeLarge        4.04%
SizeSmall       22.38%
absurd             0%

hasWarshipClass__ctc1
absurd                    0%
Class_CharlesDeGaulle  1.26%
Class_LaFayette       17.28%
Class_Brandenburg     18.76%
Class_Unknown         42.53%
Class_AlvaroDeBazan   20.18%

hasWarshipType__ctc1
absurd             0%
Type_CVN        2.99%
Type_FFG       57.49%
Type_FF        21.56%
Type_Other     17.96%

hasNationality__ctc1
absurd             0%
Nation_ES         30%
Nation_FR         30%
Nation_DE         25%
Nation_Other      15%
```

Figure 35 - MilShip PO SSBN for Test Case 2

In this case, the SSBN shows a slight reduction in the likelihood of an unknown class, and of the Charles de Gaulle class nuclear aircraft carrier. However, because of the strong similarity in the La Fayette, Brandenburg, and Alvaro de Bazan classes, the model does not strongly favor one over the other. La Fayette is slightly lower because it is a frigate without guided missiles (FF). The `hasWarshipType` node indicates that 79% of the likelihood is captured in the guided-missile (FFG) and non-guided-missile (FF) frigates. Again, the heavy inference that an "Unknown" class is observed could be a class not identified in the library, or insufficient information about the known classes. At this point, the DM can rule out the Charles de Gaulle nuclear aircraft carrier (CVN). Currently, there is no mission evidence available to the DM, which also confuses the inference of warship class. Guided-missile frigates are capable of performing Anti-Air Warfare (AAW), but frigates are not. This piece of evidence would go a long way toward establishing an accurate warship class.

4.2.4.6.3 Test Case 3

As the scenario continues, electronic sensing equipment has detected the presence of a SPY-1D air search radar emitting from the contact of interest. Accumulated evidence on `ctc1` is compiled in Table 14.

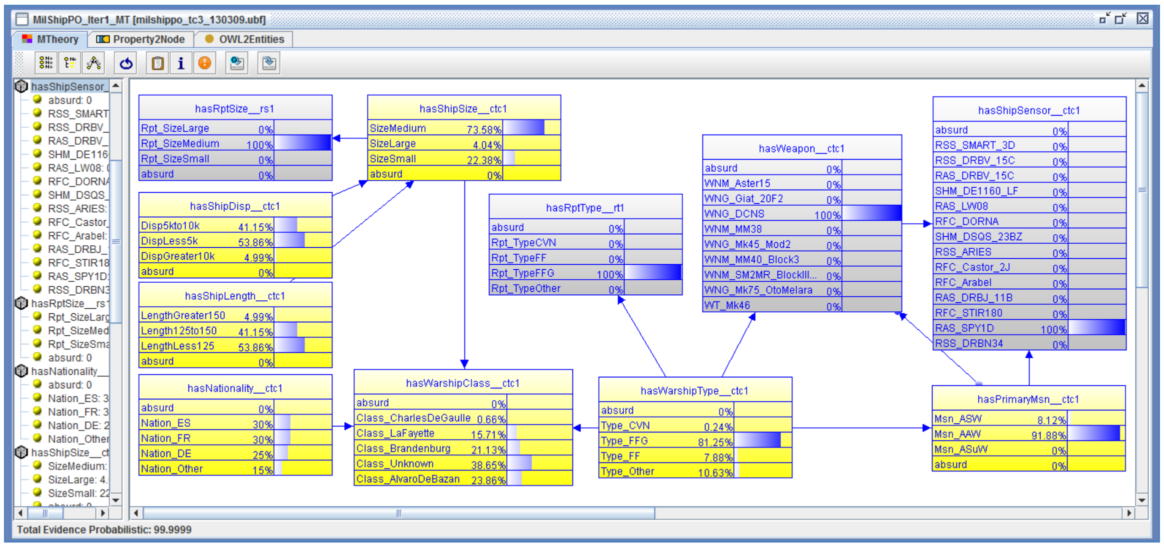
Table 14 - Evidence Table for Test Case 3

Evidence Table Case 3	
Variable	**Evidence**
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true
ctc1	hasSensor(ctc1(Ship))=RAS_SPY1D
ctc1	hasNationality(ctc1((Ship))=Nation_DE

The summary paragraph for this case reads:

> *The warship contact of interest (ctc1), reported to be a medium-sized guided-missile frigate, is radiating a SPY-1D air search radar.*

This compilation of evidence to date is instantiated in an SSBN depicted in Figure 36.



```
hasShipSensor_
  absurd: 0
  RSS_SMART
  RSS_DRBV_
  RAS_DRBV_
  SHM_DE116
  RAS_LW08: (
  RFC_DORNA
  SHM_DSQS_
  RSS_ARIES:
  RFC_Castor_
  RFC_Arabel:
  RAS_DRBJ_
  RFC_STIR18
  RAS_SPY1D:
  RSS_DRBN3
hasRptSize__rs1
  Rpt_SizeLarg
  Rpt_SizeMed
  Rpt_SizeSma
  absurd: 0
hasNationality_
  absurd: 0
  Nation_ES: 3
  Nation_FR: 3
  Nation_DE: 2
  Nation_Other
hasShipSize__ct
  SizeMedium:
  SizeLarge: 4.
  SizeSmall: 22
```

hasRptSize__rs1	
Rpt_SizeLarge	0%
Rpt_SizeMedium	100%
Rpt_SizeSmall	0%
absurd	0%

hasShipSize__ctc1	
SizeMedium	73.58%
SizeLarge	4.04%
SizeSmall	22.38%
absurd	0%

hasShipDisp__ctc1	
Disp5kto10k	41.15%
DispLess5k	53.86%
DispGreater10k	4.99%
absurd	0%

hasShipLength__ctc1	
LengthGreater150	4.99%
Length125to150	41.15%
LengthLess125	53.86%
absurd	0%

hasNationality__ctc1	
absurd	0%
Nation_ES	30%
Nation_FR	30%
Nation_DE	25%
Nation_Other	15%

hasWarshipClass__ctc1	
absurd	0%
Class_CharlesDeGaulle	0.66%
Class_LaFayette	15.71%
Class_Brandenburg	21.13%
Class_Unknown	38.65%
Class_AlvaroDeBazan	23.86%

hasRptType__rt1	
absurd	0%
Rpt_TypeCVN	0%
Rpt_TypeFF	0%
Rpt_TypeFFG	100%
Rpt_TypeOther	0%

hasWeapon__ctc1	
absurd	0%
WNM_Aster15	0%
WNG_Giat_20F2	0%
WNG_DCNS	100%
WNM_MM38	0%
WNG_Mk45_Mod2	0%
WNM_MM40_Block3	0%
WNM_SM2MR_BlockIII...	0%
WNG_Mk75_OtoMelara	0%
WT_Mk46	0%

hasWarshipType__ctc1	
absurd	0%
Type_CVN	0.24%
Type_FFG	81.25%
Type_FF	7.88%
Type_Other	10.63%

hasShipSensor__ctc1	
absurd	0%
RSS_SMART_3D	0%
RSS_DRBV_15C	0%
RAS_DRBV_15C	0%
SHM_DE1160_LF	0%
RAS_LW08	0%
RFC_DORNA	0%
SHM_DSQS_23BZ	0%
RSS_ARIES	0%
RFC_Castor_2J	0%
RFC_Arabel	0%
RAS_DRBJ_11B	0%
RFC_STIR180	0%
RAS_SPY1D	100%
RSS_DRBN34	0%

hasPrimaryMsn__ctc1	
Msn_ASW	8.12%
Msn_AAW	91.88%
Msn_ASuW	0%
absurd	0%

Total Evidence Probabilistic: 99.9999

Figure 36 - MilShip PO SSBN for Test Case 3

The SPY-1D air search radar is typically associated with Anti-Air Warfare. Therefore there is a considerable increase in likelihood associated with the two guided-missile frigates in the library, Brandenburg and Alvaro de Bazan. The Charles de Gaulle class is further reduced. While clear evidence of an associated mission would be stronger evidence, the relationship between the air search radar and the AAW mission is sufficient to sway the inference toward FFG classes. Brandenburg and Alvaro de Bazan classes are very similar in style, capability and mission. Without further evidence, it may be impossible to clearly delineate the two.

4.2.4.6.4 Test Case 4

Finally, a helicopter reports visual confirmation of a German flag flying on board the COI. All of the scenario data is compiled in Table 15.

Table 15 - Evidence Table for Test Case 4

Evidence Table Case 4	
Variable	**Evidence**
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true
ctc1	hasSensor(ctc1(Ship))=RAS_SPY1D
ctc1	hasNationality(ctc1((Ship))=Nation_DE

The summary for the scenario reads:

The warship contact of interest (ctc1), reported to be a medium-sized German guided-missile frigate, is radiating a SPY-1D air search radar.

All of the scenario evidence is instantiated in an SSBN depicted in Figure 37.

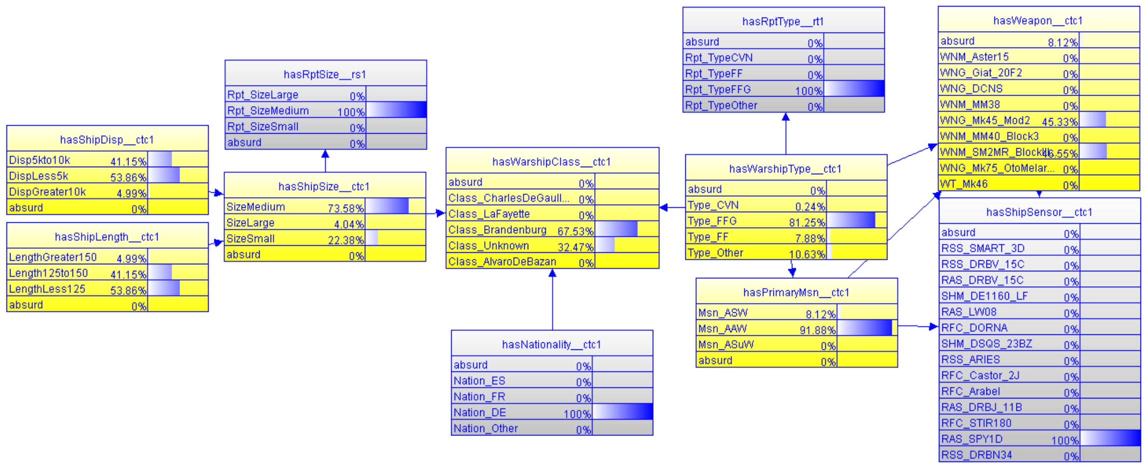


Figure 37 - MilShip PO SSBN for Test Case 4

With the addition of the nationality, there is strong likelihood that the COI is a member of the German guided-missile frigate Brandenburg class. A slight chance of an unknown class remains due to two pieces of the instantiated evidence coming from reports, which are less reliable than organic information. The Commander may now be fairly confident that the unknown ship is of the Brandenburg class, and plan accordingly.

From the four test cases, it is apparent that the PO model performs as expected in this scenario. The four classes of warships from three countries are a small snapshot of the overall domain that may be captured in a complete Military Ship Ontology. Similarly, each class has several more weapons and sensors that may be detected and ingested into the inference problem.

4.3.5 Support Activity

As introduced in Section 4.2.4, the Operate & Support Phase of the SDLC includes three functions: maintenance, improvement, and operational support [140]. In the context of the PODM, improvement is the germane task in this set. Once the PO is implemented and operating, periodic updates may be desired. Simple refinements of relationships enter the PO Construction Process at task two (Create representation for the selected related class). From this point the process proceeds with one or more iterative cycles, until the modification is complete. More elaborate improvements are assigned to a prioritized update list for entry into planning for the next spiral in the SDLC. This type of update receives the full PODM process. Improvements must be followed by an evaluation to ensure continued model functionality.

4.4 Summary

The Probabilistic Ontology Development Methodology introduced in this chapter provides a specific, guided methodology to implement the reference architecture introduced in Chapter 3. It is widely applicable across multiple systems development process styles and ontological domains. In the following chapter, the efficiency, effectiveness and teachability of the PODM are demonstrated through a user case study.

4.4.1 Applicability

The Probabilistic Ontology Development Methodology is applicable across the spectrum of ontology domains where representation of uncertainty is required. Meticulous, structured decomposition of complex problems ensures relationships are established and updated while maintaining model logic. The Military Ship PO example illustrated PO from conceptualization to operation, but the PODM is equally useful incorporating and testing uncertainty into an existing ontology.

4.4.2 Scalability of the PODM

The PODM is scalable to ontologies of varying sizes by decomposing the model into manageable tasks or conducting multiple iterations of a Spiral Development Cycle. The Military Ship example developed a usable PO on a small subset of the complete military ship domain. However, it would be a straightforward task to increase the individuals within the existing framework by populating the ontology with additional warship classes and their characteristics. This would provide a powerful decision support tool in an expansive domain. Further, additional iterations of the PO Construction Process would allow the introduction of additional relationships among sensors, weapons, and missions or alternate report types.

CHAPTER FIVE: CASE STUDY OF PROBABILISTIC ONTOLOGY DEVELOPMENT

5.1 Background

The above Probabilistic Ontology Development Methodology is an early attempt within the ST community to define and evaluate a specific process for development of an efficient, repeatable and teachable PO. To demonstrate its utility and teachability, a case study of Computer Science (CS) and Systems Engineering and Operations Research (SEOR) participants was employed to demonstrate increased efficiency and an improved final product. The overarching objectives of this study were twofold:

1. Evaluate the effectiveness of the methodology
2. Evaluate the teachability of the methodology

A group of Systems Engineering and Operations Research and Computer Science graduate students enrolled in the Volgenau School of Engineering at George Mason University was the test population for the analysis. A brief demographic survey offered at the commencement of the study captured student personal information. At the conclusion of the project, each participant completed a post-project survey to capture individual recommendations and comments about the methodology. Additional data available from the study includes the example probabilistic ontology models created by the participants, and help desk information accumulated in a database grouped by category (technical, knowledge, software, etc.).

The case study was completed by three CS/SEOR graduate students. Each was provided with materials, training, and a statement of work described in Appendix E.

Yin identifies four common applications of case studies [165]:

1. To explain the presumed causal links in real-life interventions that are too complex for surveys or experiments;
2. To describe an intervention and the real-life context in which it occurred;
3. To illustrate certain topics within an evaluation in the descriptive mode;
4. To enlighten those situations in which the intervention being evaluated has no clear, single set of outcomes.

This case study focused on explaining causal links (#1) using the process shown in Figure 38. For this case study, there is a hypothesized linkage between the PODM and better probabilistic ontologies; "better" is defined below. The difference between Participant Groups 1 and 2 is the order in which the example problems are delivered, one of which is performed before introduction to the PODM. This order was enacted to identify possible bias induced by perceived problem difficulty or unfamiliarity with the topic. Additionally, a qualitative assessment of each participant's solution was performed to illustrate the improvement in clarity and efficiency introduced with the PODM. Finally, the case study helped to describe the representation of value in a domain for which there is no standard. Here, because PO development is in its infancy as an engineering discipline, there is no standard against which the PODM can be evaluated. The case study highlights those attributes necessary to demonstrate utility as a viable engineering methodology, effectiveness and efficiency.

The case study process shown in Figure 38 illustrates the flow of activities for the two groups of participants and the Investigator. After each participant is assigned to a group, he is provided with introductory material consisting of software tutorials that introduce Protégé and UnBBayes. Then each participant completes an example problem of homework assignment difficulty using any methodology of his choosing. At this time the participant is given access to the Probabilistic Ontology Development Methodology and instructed to use the methodology to complete a second problem of similar difficulty. Following the PODM requires creation of an OWL ontology and then iterative incorporation of probability to create a probabilistic ontology. Finally, each participant is required to adequately document the model and complete an exit survey. Using output from each participant's model and documentation, the tutorials and PODM were updated as required.

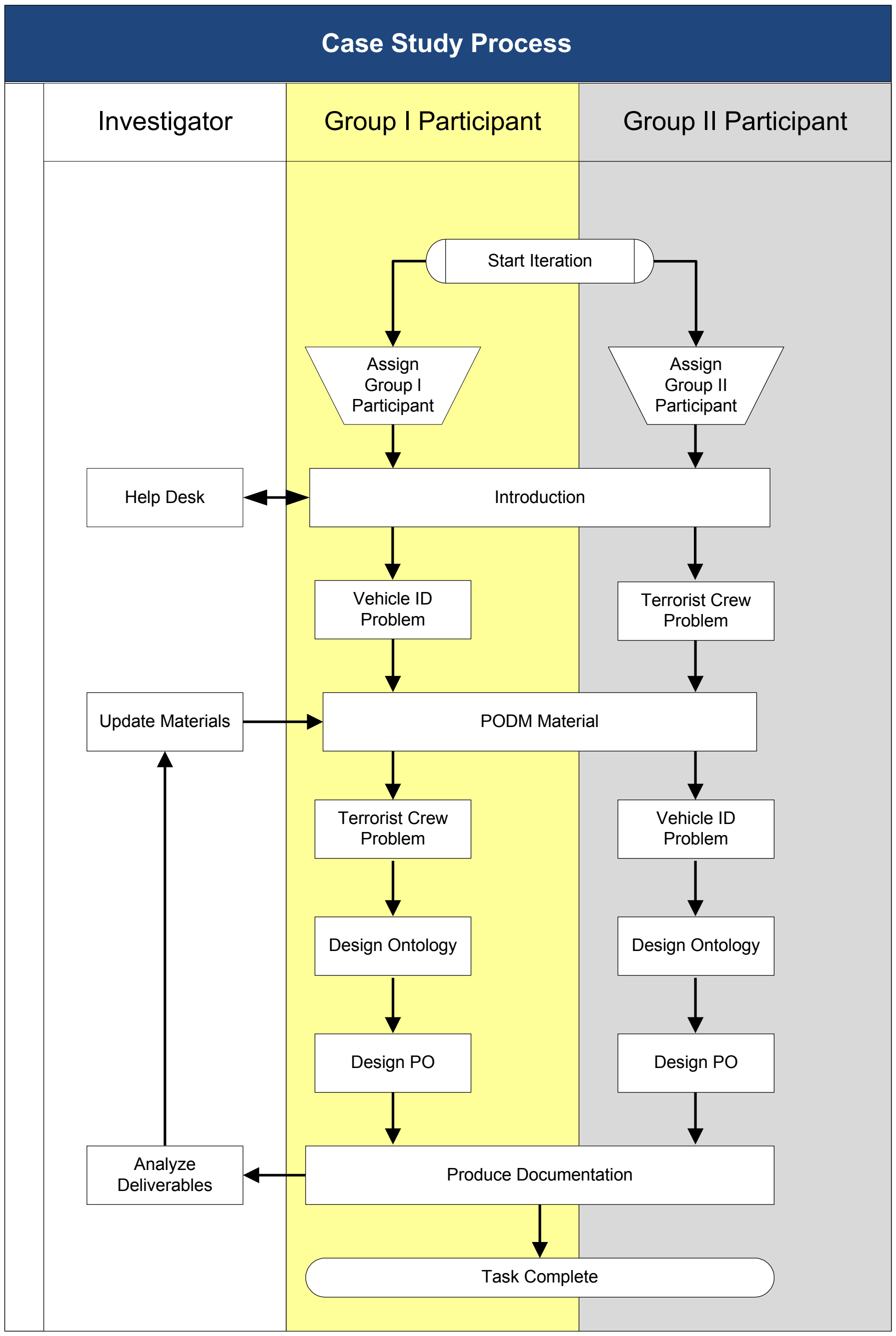


Figure 38 - Case Study Process

The case study team consisted of Dr. Kathryn Laskey in the role of Evaluator and Richard Haberlin in the role of Investigator. The primary role of the Evaluator was to provide oversight on the methodology delivery to ensure a positive academic experience was afforded the participants and the interests of the university were maintained. The Investigator executed the case study protocol described in Appendix F, including compilation of all documentation, delivery of the methodology and materials, manning the help desk, and compilation/analysis of data produced by the study. The Investigator maintained regular correspondence with the Evaluator to ensure she was cognizant of all participant issues and concerns related to the project and progress of the study.

5.2 Components of the Case Study

Yin describes five components of a successful case study: study questions, propositions, units of analysis, logic linking data to propositions, and criteria for interpreting findings [165]. Details of these components for the Probabilistic Ontology Case Study are described below.

5.2.1 Study Questions

There are three primary questions answered by this case study which align with the objectives introduced in Section 5.1, above. Connotative definitions of the applicable measures of effectiveness (MOE) for these questions are captured in Table 16. The three primary study questions are:

1. Does the methodology produce "better" probabilistic ontologies than those produced without using the PODM?
2. Does the methodology allow "more efficient" development of probabilistic ontologies than development without the PODM?

3. Is the methodology "teachable" to a population of graduate students?

Table 16 - Case Study Measures of Effectiveness

"Better"
Fewer logical errors
Fewer relational omissions
Better documentation
Runs test cases correctly
"More Efficient"
Less time to complete
More focused effort
Fewer false starts
"Teachable"
Appropriately employ methodology

The study questions describe the utility of a methodology by demonstrating the value in error reduction and reduced development time. Also, a methodology is not useful if it is complex beyond the capability of its intended user group, evaluated through the "teachability" MOE. Specific metrics for these MOEs are given in Table 17 and discussed below.

5.2.2 Propositions

Case study propositions are topics that should be examined within the scope of the study. The following propositions were introduced as factors examined within the case study and are addressed through qualitative or quantitative evidence in the Results section, below.

- The methodology improves efficiency

- The methodology improves consistency in models
- The methodology reduces the likelihood for relational errors
- The methodology reduces the likelihood of logical errors
- The methodology encourages better documentation practices
- The methodology is teachable to an applicably educated population

These propositions helped to narrow the scope of the study and identified targeted areas of inquiry for the questionnaires and discussion of results. The effect of the PODM on each of these propositions is qualitatively evaluated in the results, below.

5.2.3 Units of Analysis

Both qualitative and quantitative units of analysis were used to assess this case study. Table 17 compiles the metrics used to evaluate the MOEs of improvement, efficiency, and teachability with quantifiable terms. For the qualitative items, a simplified scale of Negative, Neutral or Positive performance in the given area was used to minimize interpretation while yielding a visualization performance.

Table 17 - Model Evaluation Criteria

"Better"	
Attribute	**Metrics**
Fewer logical errors	Number of logical errors
Fewer relational omissions	Number of relational errors Number of relational omissions
Better documentation	Qualitative documentation scale
Runs test cases correctly	Successful test cases
Overall PO quality	Expert assessment
"More Efficient"	
Less time to complete	Hours to complete the task
More focused effort	Unused material
Fewer false starts	Number of false starts
"Teachable"	
Appropriately employ methodology	Expert assessment

Most of the metrics in Table 17 are self explanatory, but a few require further discussion. Relational errors are existing, yet incorrect, representations of relationships between classes. This differs from a relational omission, which is a lacking relationship where one should exist for proper model function. Documentation is valued through qualitative assessment based on the utility of what is provided using Positive, Neutral and Negative as descriptors. Generally, adequate documentation allows a non-developer to understand the model and its functionality. Similarly, the expert assessment of overall quality is a subjective evaluation of the model based on the requirements set forth in the problem statements. Unused materials and false starts are similar in that they both represent wasted developer effort. Specifically, unused material includes model pieces, sub-models, diagrams, pseudo-code, etc. that was unnecessary to the creation of the final model. On the other hand, the more egregious false start indicates near-complete abandonment of a model in progress to begin in another direction. This issue may indicate a lack of focus or problem comprehension. Finally, the expert assessment of teachability is a holistic view of the modeling process in the context of the case study, participant aptitude, timeline, and example problem to evaluate comprehension of the PODM by the participant. The simplistic scale of Positive, Neutral, and Negative is used for all of the subjective evaluations to minimize distraction involved with qualitative grading bias.

5.2.4 Logical Linking of Data to Propositions

The logic linking technique used in this case study was explanation building. With each participant performing an example problem before and after access to the PODM, improvement to the PO product and efficiency in its development could be attributed to the structured framework and methodological procedure applied to development. Similarly, completion of the PODM tasks specified within the activities provided a measure of detailed documentation that assists in reconstruction of the developer's methodology. Figure 39 illustrates the linkage from MOE through proposition to data.

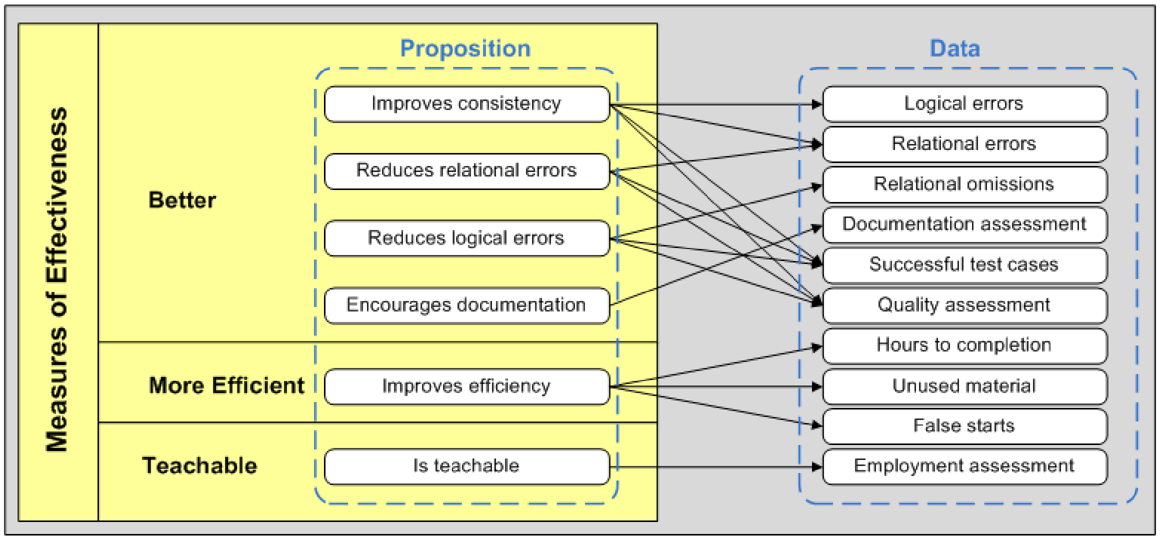


Figure 39 - Logical Linking

The three MOEs were represented by six propositions which were then mapped to the ten data points collected for each participant. The first six data points were linked to the first four propositions which support the “Better” MOE. Similarly, the next three data points were linked to the “Improves efficiency” proposition and MOE. Finally, the last data point supports the “Teachable” MOE through an expert assessment of participant success in employing the PODM.

Data were tabulated as individuals, and also rolled-up into overall results, summarized in Section 5.4.2. Using the mapping in Figure 39 performance was observed to evaluate the MOEs. The data suggest the existence of a causal linkage between the PODM and better, more efficient POs. Insights gleaned from the process and results are summarized in the results, Section 5.5.

5.2.5 Criteria for Interpreting Findings

There are possible rival explanations for under-performance of participants prior to implementation of the PODM including immature/buggy software, skill and expertise level of participants, and participant motivation. Each of these is considered and discussed appropriately in the comparison of results.

5.3 Case Study Protocol

5.3.1 Characteristics of the Intended Sample

The PODM is primarily intended for utilization by academic and professional users with baseline expertise in the area of applicability, probabilistic inference. It was therefore assumed that those participating in the study had knowledge and interest in both probabilistic ontology development and its application. The Evaluator and Investigator gauged the background knowledge available by each of the participants and the likelihood of success within the desired timeline. A complete description of the Case Study Protocol is included as Appendix F.

5.3.2 Design and Methodology

As introduced above, this case study provided participants with a methodology to facilitate development of probabilistic ontologies. Explicit details of participant requirements and tasks are summarized in Table 18 and detailed in Appendix E.

The *Entry Survey* was designed to capture basic demographic information and was administered to participants during the introduction to the study. Upon completion of the project, participants completed the *Exit Survey* to capture issues and recommendations relating to the methodology.

Participants installed two shareware software products from online websites before beginning two tutorials. The first tutorial explicitly led the participant through construction of an ontology using the Protégé software tool. The second tutorial builds on this solution by applying uncertainty to the created ontology in FOL MEBN using UnBBayes to create a probabilistic ontology. The tutorials taught software implementation rather than inferential reasoning or modeling skills. The example used for both tutorials is the well-known Patient Diagnosis academic problem. Next, each participant was tasked to create a probabilistic ontology for an example problem by any means they chose. Each of the two groups was given a different problem: Group 1 modeled the Vehicle Identification problem and Group 2 modeled the Terrorist Crewmember problem. These problems and their solutions are provided as Appendices C and D. With their first problem complete, the participants were given a detailed description of the PODM and asked to implement it to solve a second problem. In this case, each group was given the example they did not have for the pre-PODM assignment (Group 1: Terrorist Crewmember, Group 2: Vehicle ID).

Participants were required to follow the SOW. Any models, documentation, false starts, etc. produced were returned to the Investigator to be used in the overall evaluation of quality and efficiency. Issues and queries relating to the software or the methodology were submitted via the help desk. The Investigator responded to these queries within 24 hours of receipt and collected information in the help desk database.

5.3.3 Confidentiality

The preferred method of communication between participants and the Investigator was via email so that query data could be captured and catalogued in the help desk database. Email correspondence between participants and the Investigator was also stored in a project email database maintained by the Investigator. These materials will be turned over to the Evaluator upon completion of this research.

There were no physical, psychological, social, or legal risks to the participants associated with use of the methodology, nor were participants recorded in any manner, other than correspondence with the Investigator. Use of the methodology and access to the help desk were designed to be straightforward without deception. Any confusion experienced by participants was unintentional and corrected by the Investigator at the earliest opportunity.

5.3.4 Data Collection Instruments

Several collection instruments were used to collect data for the case study. These documents are included in the Appendices, and are briefly described below.

A. *Entry Survey*. This questionnaire was distributed to participants to capture basic information regarding education, ethnicity and age.

B. *Informed Consent Form*. This form was signed by all participants acknowledging participation and that they were to receive monetary compensation for successful completion of the SOW. A digital copy was provided to each participant.

C. *Exit Survey.* This questionnaire was distributed to participants at the completion of the study to capture issues and recommendations about the proposed methodology and the overall project experience.

D. *Help Desk.* This database captured queries and allowed collection of data from the study. Participants submitted software and technical queries to the Investigator via the help desk.

E. *Tutorials Describing Probabilistic Ontology Development.* These tutorials generally followed a simplified example using the Probabilistic Ontology Development Methodology described in the SOW.

F. *Probabilistic Ontology Development Methodology.* Participants were provided with this methodology, which was followed to complete the SOW. Issues and concerns with the methodology were submitted to the Investigator via the help desk or made known via the exit survey.

5.3.5 Cooperating Organizations

This case study was conducted completely under the cognizance of George Mason University staff and students. No outside organizations had access to the participants or collected data.

5.3.6 Tasking

Participants completed tasks according to a statement of work provided at the commencement of the study. Specific SOW tasks for each group are listed in Table 18. Explicit procedures for these tasks are included in Appendix E.

Table 18 - Participant Statement of Work (SOW)

Statement of Work				
	Group I		Group II	
0	Complete Informed Consent Form	E.5	Complete Informed Consent Form	E.5
I	**Setup and Preparation**		**Setup and Preparation**	
I.1	Install Protégé 4.1	E.6.1	Install Protégé 4.1	E.6.1
I.2	Install UnBBayes 4.11.4	E.6.2	Install UnBBayes 4.11.4	E.6.2
I.3	Complete entry survey	E.7	Complete entry survey	E.7
II	**Training**		**Training**	
II.1	Complete Protégé Tutorial	E.8.1	Complete Protégé Tutorial	E.8.1
II.2	Complete Patient Diagnosis Tutorial	E.8.2	Complete Patient Diagnosis Tutorial	E.8.2
II.3	Review required documentation	E.9	Review required documentation	E.9
III	**Development**		**Development**	
III.1	Create probabilistic ontology for Vehicle Identification Problem	E.10 App C	Create probabilistic ontology for Terrorist Crewmember Problem	E.11 App D
III.2	Review the PODM	E.12 Ch 4	Review the PODM	E.12 Ch 4
III.3	Create probabilistic ontology for Terrorist Crewmember Problem	E.11 App D	Create probabilistic ontology for Vehicle Identification Problem	E.10 App C
IV	**Documentation**		**Documentation**	
IV.1	Capture task hours and Feedback	E.9.2	Capture task hours and Feedback	E.9.2
IV.2	Complete exit survey	E.13	Complete exit survey	E.13

Group SOWs differed only in the order of example problems completed by the participants.

5.3.7 Materials

Participants were given electronic access to all materials necessary to complete the assigned tasks, including documents, forms, surveys, software, and an explicit statement of work. Each participant had access to a personal computer with Internet access and administrator permissions required to install and operate the Protégé and UnBBayes software tools.

5.3.7.1 Tutorials

Each participant was required to complete a tutorial in the Protégé ontology modeling and UnBBayes probabilistic ontology modeling software packages using the Patient Diagnosis example problem. These two instructional examples allowed a CS/SEOR participant to become familiar with the software tools. They are provided below as Appendix B.

5.4 Evidence

Yin identifies six primary sources of evidence: documentation, archival records, interviews, direct observation, participant-observation, and physical artifacts [165]. Multiple sources of evidence from multiple participants create a more robust case study. The evidence for each participant is summarized in Section 5.4.2, primarily consisting of documentation and physical artifacts (models).

5.4.1 Sources

This work relied on evidence provided primarily through documentation in the form of the example problem solutions and survey data provided by each participant.

The documentation was used to quantify the difference in participant performance solving example problems before and after introduction to the PODM. Surveys were used to capture overall opinion of the PODM methodology and recommendations for improvement. There was also the availability of physical artifacts in the form of the models created for each of the problems. These too, offered insight into the development process executed by each participant. The specific data established by each instrument is listed below.

5.4.1.1 Entry Survey

The entry survey was used to elicit demographic data to identify possible barriers to success caused by cultural/language difficulties, age, or unfamiliarity with the subject matter. To establish these factions, questions were posed regarding:

- Birth year
- Education and major
- Ethnicity
- Primary language
- Ontology familiarity
- Probabilistic ontology familiarity

There were also some general questions regarding marital status and employment status that were not used in processing the results.

5.4.1.2 PODM Effect

Documentation from the example problem solutions, as well as the model representations, was used to establish the effect produced by utilization of the PODM for PO development. In some cases this was a quantitative effort, counting differences in the number of:

- Logical errors,
- Relational errors,
- Relational omissions, and
- Time to complete.

Establishing the PODM effect in these areas was a relatively simple case of counting values before and after introduction to the methodology, as shown in the evidence tables, below. There was also the subjective assessment of the actual models and their documentation used to establish:

- Documentation quality,
- Test success/fail,
- Unused material, and
- Overall model quality.

These areas used the simplistic scale of Negative, Neutral, and Positive to illustrate performance results from use of the PODM.

5.4.1.3 Exit Survey

The exit survey was used to elicit general thoughts about the overall value of PODM, its strengths and weaknesses, and areas for future improvement. Participants were also offered an opportunity to critique the materials, including software, the tutorials, and the methodology.

5.4.2 Participants

Five participants began the case study, and three completed the statement of work. Group 1 (Participants 4 and 13) completed the Vehicle Identification Problem first, and Group 2 (Participant 12) completed the Terrorist Crewmember problem first. An abbreviated statement of work that excluded the requirement for a working UnBBayes model was offered to Participants 13 and 20 when it was determined that the immaturity of the software was becoming a distraction from the purpose of the case study. The individual PODM Effect data for the three participants providing case study input is summarized in Tables 19-21, with identity protection provided by assignment of a unique participant number. Quantitative values are assigned to logical errors, relational errors, relational omissions, time and false starts. Qualitative evaluations are characterized by negative (-), neutral (N) or positive (+) assessment. Shaded areas indicate areas of non-performance or no data.

5.4.2.1 Participant 4 (Group 1)

Participant 4 entered the study as an SEOR student with limited background in ontologies and probabilistic ontologies. He had significant trouble with the software and tutorials, expending a great deal of time on their completion. Also, he used the PODM to complete both problems, eliminating the ability to evaluate performance results from its implementation, indicated by the shaded column in Table 19. Neither model was of sufficient maturity to operate within UnBBayes.

Table 19 - Participant 4 Summary Data

"Better"				
Attribute	**Units of Analysis**	**Vehicle**	**Terrorist**	**Eval**
Fewer logical errors	Number of logical errors	2	0	
Fewer relational omissions	Number of relational errors	0	0	
	Number of relational omissions	3	6	
Better documentation	Qualitative documentation scale	+	+	
Runs test cases correctly	Successful test cases			
Overall PO quality	Expert assessment	-	N	
"More Efficient"				
Less time to complete	Hours to complete the task			
More focused effort	Unused material	N	N	
Fewer false starts	Number of false starts	0	0	
"Teachable"				
Appropriately employ methodology	Expert assessment	+	+	

Ultimately Participant 4 did not produce working software models for either example problem, but provided a detailed development model of both problems in the form of documentation, tables, and graphics. Unfortunately, it is likely that his inexperience with the subject matter had a negative effect on his ability to successfully complete the tasks, as evidenced by the significant number of errors in modeling. As previously mentioned, Participant 4 used the PODM for both problems, so there is no trend data available to show improved efficiency or effectiveness. However, he did follow the PODM explicitly, thereby demonstrating its teachability.

5.4.2.2 Participant 12 (Group 2)

Participant 12 was a CS graduate student who was very comfortable with both software tools and had no problems with the tutorials. He experienced a decline in performance due to some unused material in his post-PODM example problem caused by limited familiarity with ontologies which caused confusion in identifying classes.

Table 20 - Participant 12 Summary Data

"Better"				
Attribute	**Units of Analysis**	**Terrorist**	**Vehicle**	**Eval**
Fewer logical errors	Number of logical errors	0	0	🟨
Fewer relational omissions	Number of relational errors	0	0	🟨
	Number of relational omissions	2	1	✓
Better documentation	Qualitative documentation scale	-	+	✓
Runs test cases correctly	Successful test cases	+	+	🟨
Overall PO quality	Expert assessment	N	+	✓
"More Efficient"				
Less time to complete	Hours to complete the task	9	7	✓
More focused effort	Unused material	N	-	✗
Fewer false starts	Number of false starts	0	0	🟨
"Teachable"				
Appropriately employ methodology	Expert assessment	▨	+	✓

Similarly, as a Computer Science major he had limited prior knowledge of Bayesian inference, and therefore used the tutorials as lesson guides to properly build the BN required for testing. In both models he experienced one or more relational omissions by not providing context variables linking the class of interest to another ordinary variable. In the Terrorist Crewmember problem, one context variable establishes the person of interest as a member of a particular group, and the other links a specific friend or relative to a crewmember. The crewmember may be a member of many organizations or have many relatives, but not all are necessarily associated with terrorism. In the Vehicle ID problem the context variable locates a particular target in a region which establishes the weather and terrain it experiences.

Overall, Participant 12 was successful in completing the SOW and creating two operational probabilistic ontologies. While he commented that the PODM was not necessary to his success, his documentation improved from nonexistent to significant after proper use of the PODM. Further, the example problems were intentionally simplistic to allow success across a broad spectrum of participant capability. It is likely that Participant 12 would experience success with a more difficult PO problem if armed with the PODM. Also, despite experiencing some issues with class identification, he was able to complete the problem in a shorter amount of time when the PODM was utilized. However, there is also the possibility that completing two problems in a short amount of time provided sufficient experience to decrease production time due to the learning effect, as described below.

5.4.2.3 Participant 13 (Group 1)

Participant 13 was an SEOR graduate student assigned to perform an abbreviated case study in which models were created in the form of documentation, tables and graphics, but not encoded in UnBBayes. Although his primary area of focus is systems engineering, he had moderate familiarity with ontologies through work and school-related research.

Table 21 - Participant 13 Summary Data

"Better"				
Attribute	**Units of Analysis**	**Vehicle**	**Terrorist**	**Eval**
Fewer logical errors	Number of logical errors	0	0	🟨
Fewer relational omissions	Number of relational errors	0	0	🟨
	Number of relational omissions	0	0	🟨
Better documentation	Qualitative documentation scale	N	N	🟨
Runs test cases correctly	Successful test cases	▨	▨	▨
Overall PO quality	Expert assessment	+	+	🟨
"More Efficient"				
Less time to complete	Hours to complete the task	4.3	1.3	✓
More focused effort	Unused material	N	N	🟨
Fewer false starts	Number of false starts	0	0	🟨
"Teachable"				
Appropriately employ methodology	Expert assessment	▨	+	✓

Overall, Participant 13 was successful in completing the SOW. He noted that not using a software package for modeling made it difficult to avoid extending the model beyond the scope required to complete the assigned problem. The software would have allowed him to check his model in-situ and stop when the problem was complete. That said, his logic and structure were well-founded and could be used to implement either PO. From the tabulated data, it appears that the PODM provided Participant 13 with limited benefit other than reducing the development time. However, it is possible that this could also be attributed to him finding the Terrorist Crewmember problem easier. He successfully followed the PODM steps for the second problem, which implies teachability.

5.5 Results

5.5.1 Analytic Strategy

Performance analysis before and after introduction of the methodology is used to establish a causal linkage between use of the PODM and better probabilistic ontologies produced more efficiently. Further, the results are viewed both in the context of problem order and by problem type to determine if one problem was generally more difficult. The PODM is also evaluated against the propositions introduced in Section 5.2.2. and mapped in Figure 39. Finally, some rival explanations are offered that could explain the observed data, and provide topics for future evaluation.

5.5.2 Problem Order

The PODM Effect displayed in Table 22 indicates improving performance in three of the categories, two of which are quantitative. First, there was a slight reduction in relational errors from the first to the second problem, and all of the averaged errors were produced by the same participant. So, he benefited from the PODM, learned from the first problem, or found the second problem less difficult. Next, there was significant improvement in documentation when using the PODM, since its activities specify generation of a number of tables to compile information regarding the problem space. All participants required less time to produce the models when following the defined methodology.

Table 22 - Problem Order Results

“Better”										
		First Problem				Second Problem				Eval
Attribute	**Units of Analysis**	**4**	**12**	**13**	**μ†**	**4**	**12**	**13**	**μ†**	
Fewer logical errors	Number of logical errors	2	0	0	0	0	0	0	0	🟨
Fewer relational omissions	Number of relational errors	0	0	0	0	0	0	0	0	🟨
	Number of relational omissions	3	2	0	1	6	1	0	.5	✓
Better documentation	Qualitative documentation scale	+	-	N	-	+	+	N	+	✓
Runs test cases correctly	Successful test cases	▨	+	▨	+	▨	+	▨	+	🟨
Overall PO quality	Expert assessment	-	N	+	+	N	+	+	+	🟨
“More Efficient”										
Less time to complete	Hours to complete the task		9	4.3	6.6		7	1.3	4.2	✓
More focused effort	Unused material	N	N	N	N	N	-	N	-	✗
Fewer false starts	Number of false starts	0	0	0	0	0	0	0	0	🟨
“Teachable”										
Appropriately employ methodology	Expert assessment	+ *	▨	▨	▨	+ *	+	+	+	✓

†Participant 4 used the PODM for both problems, so his data is not included in the average scoring.
** Participant 4 completed both problems using the PODM, demonstrating successful employment in doing so.*

The negative evaluation in the unused material category is a result of one participant having difficulty identifying the correct classes for his second problem. Finally, all participants were able to execute the PODM, implying that it is indeed teachable to a population of graduate students. It is reasonable to assume that a population of industry experts would be equally successful in following its design.

5.5.3 Problem Example

Participants were consistent in their completion of the two example problems regardless of the order in which they were assigned. Data in Table 23 indicate no deviations from the trends noted above in Table 22.

Table 23 – Problem Example Results

"Better"									
		Vehicle Problem				**Terrorist Problem**			
Attribute	**Units of Analysis**	**4**	**12**	**13**	**μ**	**4**	**12**	**13**	**μ**
Fewer logical errors	Number of logical errors	2	0	0	0.7	0	0	0	0
Fewer relational omissions	Number of relational errors	0	0	0	0	0	0	0	0
	Number of relational omissions	3	1	0	1.1	6	2	0	2.1
Better documentation	Qualitative documentation scale	+	+	N	+	+	-	N	N
Runs test cases correctly	Successful test cases	▨	+	▨	+	▨	+	▨	+
Overall PO quality	Expert assessment	-	+	+	+	N	N	+	+
"More Efficient"									
Less time to complete	Hours to complete the task		7	4.3	5.7		9	1.3	5.2
tMore focused effort	Unused material	N	-	N	-	N	N	N	N
Fewer false starts	Number of false starts	0	0	0	0	0	0	0	0
"Teachable"									
Appropriately employ methodology	Expert assessment	+	+	▨	+	+	▨	+	+

These results indicate that the participants found the problems to be of similar complexity, averaging five hours to complete and 1-2 logical or relational errors.

5.5.4 Proposition Analysis

Recall that the propositions introduced in Section 5.2.2 are statements about the PODM that were to be evaluated by the case study. Using the logical linking illustrated in Figure 39, a qualitative assessment for success against the propositions can be made for each participant. Again, the assessment is characterized by a negative (-), neutral (N) or positive (+) score. The summary of these qualitative grades is shown in Table 24.

Table 24 - Proposition Results

Proposition					
	4	12	13	μ	
Improves efficiency	▨	N	+	+	✓
Improves consistency	▨	+	N	+	✓
Reduces relational errors	▨	+	N	+	✓
Reduces logical errors	▨	+	N	+	✓
Encourages documentation	+	+	N	+	✓
Teachable	+	+	+	+	✓

The table shows positive results for all six propositions. Quantitative values for error consistency and error reduction may be neutral due to the relative simplicity of the problem and the experience level of the participant group. Apparently, these problems were well within the capability of the group, minimizing the chance of relational or logical error, even before introduction to the PODM. However, documentation was generally improved along with overall model clarity. For two of the three participants, the PODM reduced development time supporting the claim of improved efficiency.

Finally, all participants successfully employed the methodology indicating that it was indeed teachable with minimal background experience.

5.5.5 Rival Explanations

Rival Explanations seek to elucidate additional causes of the effects represented in the results. In this case, these Rival Explanations need to illustrate why, other than due to the PODM, the participants produced better models more efficiently after being introduced to the methodology.

5.5.5.1 Learning Effect of second problem

The most likely Rival Explanation is that of the learning effect due to performing multiple problems within a short time period. This would manifest as evidence in the form of improved efficiency and a better product as the participant became more comfortable with the software and the domain. An effort to eliminate this effect was established by using examples from significantly different genres.

5.5.5.2 Education Level of Participants

Another possible Rival Explanation for success in the case study is the experience level of the participant group. Because all were graduate students, it is not unreasonable that they are primed to find any such methodology within their capability, making the PODM no more teachable than another methodology. Also, there is a possibility that a participant was previously exposed to one or both of the example problems during their coursework. An effort to limit this explanation was made by using relatively new models for the example problems.

5.5.5.3 Simplicity of Problem

It is also possible that one or both of the problems was merely too simplistic for the participants which would characterize a better solution and a more efficient process. Little could be done to limit this Rival Explanation without running the risk that some participants would be unable to complete the SOW. However, the data captured in Table 23 indicate that the problems were of similar complexity.

5.6 Summary

Establishing a value in a discipline for which no existing baseline is in place is a difficult task. The case study analysis of CS/SEOR participants suggests that the Probabilistic Ontology Development Methodology is indeed teachable to a student population and aids in the production of better POs, in less time.

CHAPTER SIX: CONCLUSIONS

The overarching objective of this research effort is to provide a structured methodology, based in model-based systems engineering principles, to develop probabilistic ontologies more efficiently and effectively. The key tasks in achieving this purpose are a probabilistic ontology reference architecture, and a probabilistic ontology development methodology. Within the framework of the RAPOD the iterative steps of the PODM lead the developer through a concise, thoughtful process scalable for development of any size probabilistic ontology.

Building toward filling the identified void in the current ST body of knowledge regarding PO development, Chapter 2 summarized current literature on reference architectures, ontological engineering, evidential reasoning, knowledge engineering, and case study development. From this review several points become clear. First, architectures and architecting are a recognized requirement for many industries including software, manufacturing, and information technology. Reference architectures encourage consistency and provide for reuse, minimizing redundant effort and reducing production cost. Ontological engineering and ontology development are matured domains with many contributors and an ever-increasing body of knowledge to improve ontology development processes. It is therefore wise to leverage advances in this area for the production of probabilistic ontologies. Evidential reasoning is also an established domain whose principles are applied to ontological engineering to produce probabilistic ontologies. Finally, a review of knowledge engineering provides insight into inferential reasoning capability of knowledge-based systems through the use of first-order logic of graphical probability models and Expressive Probabilistic Languages.

Chapter 3 described creation of a blueprint for probabilistic ontologies in a given domain in the form of the Reference Architecture for Probabilistic Ontology Development. This construct defines integral components of the PO, clarifies language, provides a reference for evaluation, and establishes specifications. In this way, effort expended for the creation of a PO can be leveraged in the development of subsequent models in similar domains through reuse. Use of a reference architecture aids in the design, implementation and reuse of domain-specific probabilistic ontologies by specifying coherent choices of components to create a template for an appropriate solution.

Chapter 4 detailed the Probabilistic Ontology Development Methodology, a systematic construction construct for probabilistic ontology design from conceptualization to implementation. The iterative process provides sufficient flexibility for application to numerous design methodologies including agile, spiral, and waterfall constructs. The PODM provides a specific, guided methodology to implement the reference architecture developed for the domain.

The application of a reference architecture and development of a probabilistic ontology using the new methodology is illustrated through a running example in which a military ship probabilistic ontology is developed for use in a decision support system.

The MilShip PO is used to infer the most probable class of military warship from a set of ship classes, given a varied amount of reported information. First, a reference architecture is developed for the model. Then, the PO is created by following the PODM, detailed in Appendix A. The result is an operational probabilistic ontology scalable to application across the naval military domain. The PODM is also used to solve two example problems (Vehicle Identification and Terrorist Crewmember) used by case study participants. Solutions are given in Appendices C and D, respectively. The first describes development of a PO to infer the type of vehicle given uncertain incoming reports, and the second describes development of a PO to establish the likelihood a particular merchant ship crewmember is a terrorist given his relationships, influences and organizational memberships. These, as well as a Patient Diagnosis Probabilistic Ontology Development Tutorial, provide the reader with multiple application examples at varying levels of complexity.

With minimal existing literature on the subject of PO development, it was necessary to establish a baseline of value for the provided methodology. A case study was performed on George Mason University CS and SEOR graduate students to validate the effectiveness, efficiency, and teachability of the PODM. Results indicated possible benefit from the application of the methodology, and further case studies of an iteratively improved PODM will further strengthen the value of the methodology within the ST community. As part of the case study, two tutorials were developed that initiated the user to the process of building a probabilistic ontology from an ontology using the Protégé and UnBBayes software tools. These provide step-by-step instruction to allow users familiar with ontologies and Bayesian networks to advance into development of functioning probabilistic ontologies at the academic level. These two tutorials are available to the ST community through a Volgenau School of Engineering at George Mason University portal.

Most importantly, together the RAPOD and the PODM fill a void in the ST community for the construction of probabilistic ontologies across the breadth of development, from simple to complex. Using solid model-based systems engineering principles and a spiral development process, an efficient methodology enables developers to reuse existing ontologies and leverage previous work for the production of probabilistic ontologies.

6.1 Summary of Contributions

In summary, this research provides the following contributions to the body of knowledge in the Semantic Technology community:

1. Reference Architecture for Probabilistic Ontology Development (RAPOD): provides unification of effort by identifying concepts, processes, languages, theories and tools for designing and maintaining probabilistic ontologies.
2. Probabilistic Ontology Development Methodology (PODM): provides iterative activities and tasks to produce a probabilistic ontology from conceptualization to implementation with in-situ evaluation steps to ensure coherence.
3. Probabilistic Ontology Case Study Evaluation: demonstrates the utility of the Probabilistic Ontology Development Methodology by evaluating its effectiveness, efficiency, and teachability.

4. Military Ship Probabilistic Ontology (MilShip PO): provide a running example for the RAPOD and PODM resulting in an operational probabilistic ontology for inference about military ships, scalable to any maritime theater of interest.
5. Patient Diagnosis Protégé and UnBBayes Tutorials: provide introductory instruction for ontology development in Protégé and subsequent incorporation of uncertainty to produce a probabilistic ontology in UnBBayes.

6.2 Future Research Areas

A natural step in the evolution of this work is to continue extension of the body of knowledge and leverage the work across the domain. Four areas of interest for future work are introduced.

Of greatest interest is continued refinement of the PODM to ensure it keeps pace with the rapidly evolving probabilistic ontology domain. At time of printing, the PODM was updated using input from the PO Case Study Evaluation results. It would be beneficial to conduct additional case study assessments using the updated version of the PODM and example problems of increasing complexity. A process of continuous iteration and evolution will ensure the relevance and maturity of this tool are maintained.

As domains increase in complexity, so does the amount of work required of the developer. A necessary area of focus is the automation boundary of the PODM. With the exception of machine learning for probabilistic parameters and the underlying network structure, the entire PODM is completed by hand. Investigation of automation for a portion of the process would allow extension beyond academic problems and into the realm of extremely large and complex ontologies. Similarly, the UnBBayes tool is state of the art for PO development, but is still immature. Advancement of the desktop software will open the utility of PO application to a greater number of users.

Learning is discussed as part of ontological engineering and knowledge engineering. However, these processes would be conducted external to the PODM and their results incorporated. An updated PODM should include incorporation of learning into the iterative construction process. Automation of the PODM will provide an opportunity for this synergy to be realized.

Finally, the Military Ship PO should be extended to include a more intricate differentiation between organic and inorganic sensors and reports, allowing different pedigrees to be associated with more trustworthy sources. Also, for the purpose of illustration the MilShip PO contains only a few ship classes out of the dozens available in the Western European domain. These and their attributes should be extended to increase the utility for decision support. A further extension to the overall MilShip PO to include merchant and fishing ships would create a maritime PO useful across the domain.

APPENDIX A: MILITARY SHIP PROBABILISTIC ONTOLOGY

A.1 Introduction

The Military Ship PO introduced in Chapter 1 and developed in Chapter 4 is detailed below in a comprehensive running example. The MilShip PO was completed in a single spiral in response to a single Prime Query. Completion of these activities establishes a framed solution to a specific decision problem grounded in an inclusive ontology representing its entities and incorporation of probabilities to represent uncertainty. Portions of this work are included in the PODM descriptions of Chapter 4.

A.2 Frame Activity

The Frame Activity encompasses necessary tasks to scope the problem and its requirements based on the Objective Statement. Framing captures the tasks of bounding the problem space, defining metrics, identifying important attributes, drafting the class diagram and selecting modeling tools.

A.2.1 Define the Spiral

Defining the spiral establishes the overall objective of the spiral and identifies the Prime Query that satisfies this objective.

A.2.1.1 Objective Statement

The objective statement clearly describes the purpose of the PO in a manner understandable to both the stakeholder and the modeler.

> ***Objective**: The Military Ship Probabilistic Ontology will aid the user in inferring the specific class of a warship for a contact of interest given the arrival of uncertain information about its sensors, weapons, nationality and physical characteristics*

A.2.1.2 Prime Query

The Prime Query defines the principal topic of interest to the stakeholder in the form of a question that supports the objective.

> ***Prime Query – 1**: The unknown contact belongs to which of the warship classes in the AOR-specific library?*

A.2.2 Define Requirements

The goal of this task is to capture attributes that should be controlled within the model in written requirement statements, to be validated by the Stakeholder DM and measured by the metrics.

A.2.2.1 Requirements Table

The Requirements Table captures the validated requirements that represent behaviors, applications, constraints, properties, and attributes that directly support the Spiral Objective Statement.

Requirements Table		
ID	**Title**	**Description**
R1	**Determine warship class**	**PQ: Determine warship class from library of possible classes in the AOR**
R2	**Accept Reports**	**Incorporate uncertain information from arriving reports**
R2.1	Accept Type Reports	Incorporate reports about the type of warship detected (FF, FFG, CVN, Other)
R2.2	Accept Size Reports	Incorporate reports about size information for the detected ship (Small, Medium, Large)
R3	**Incorporate Class Descriptors**	**Incorporate a library of information about possible classes is included in a Military Ship ontology that will be accessed for reasoning**
R3.1	Incorporate Nationality	Incorporate information about the nationality of a class of warship
R3.2	Incorporate Warship Type	Incorporate information about the type of warship for a given class
R4	**Incorporate Mission Information**	**Incorporate a library of information about the primary mission, and the sensors/weapons used to accomplish this mission**
R4.1	Incorporate Primary mission	Incorporate information about the primary mission of a class of warship
R4.2	Incorporate Ship Sensors	Incorporate information about sensors hosted on the class of warship
R4.3	Incorporate Weapon	Incorporate information about weapons carried on the class of warship
R5	**Incorporate Descriptive Information**	**Incorporate descriptive information assists in classification of gross naval class and size**
R5.1	Incorporate Displacement	Incorporate class displacement data
R5.2	Incorporate Length	Incorporate class length data
R6	**Incorporate Performance Characteristics**	**Incorporate performance information related to deployed system hardware**
R6.1	Execute Quickly	Generate solution in t minutes or less
R6.2	Execute Efficiently	Compute solution on a PC (Intel 1.3 GHz)

A.2.2.2 Individuals Table

Each class within the ontology contains individuals that are specific to the domain of interest. Individuals of each class define the scope of the ontological library that is accessed to produce the probabilistic ontology.

Individuals Table	
Class	Individuals
Ship	ctc1, ctc2, ctc3, ctc4
SizeRpt	rs1, rs2, rs3, rs4, rs5
TypeRpt	rt1, rt2, rt3, rt4, rt5
Nation	Nation_(DE, ES, FR, Other)
ShipSize	Size_(Large, Small, Medium)

ShipSensor	RAS_(SPY-1D, LW08, DRBJ-11B, DRBV-15C)
	RFC_(Arabel, Castor_2J, DORNA, STIR180)
	RSS_(Aries, SMART-3D, DRBV-15C, DRBN-34)
	SHM_(1160-LF, DSQS-23BZ)

ShipWeapon	WNG_(DCNS, Giat_20F2, Mk45_Mod2, Mk75_OtoMelara)
	WNM_(Aster15, Mistral, MM38, MM40)
	LWT_Mk46

WarshipClass	Class_(AlvaroDeBazan, Brandenburg, CharlesDeGaulle, LaFayette, Unknown)
WarshipMission	Msn_(AAW, ASuW, ASW, Strike)
WarshipType	Type_(CVN, FF, FFG, Other)

A.2.3 Define Metrics

Metrics are parameters or measures of quantitative assessment used for measurement, comparison or to track performance of the requirements against some benchmark established in collaboration with the Stakeholder DM.

A.2.3.1 Metrics Table

Through experience and stakeholder elicitation, performance goals and their associated metrics may be identified and captured for use in model evaluation.

Metrics Table					
Requirement		Metric			
ID	Name	ID	Name	Definition	Units
R1	Determine warship class	M1	Model Accuracy	Correctly identify the warship class (≥ 85%)	Percent
R2	Reports Descriptors Mission Descriptive	M2	Model Flexibility	Absorb/operate on ontology of 500 entities (max expected size)	Items
R6	Performance	M3	Execution Time	Generate solution in t minutes or less	Min
R6	Performance	M4	Model Efficiency	Compute solution on pc computer (Intel 1.3GHz)	Processor

A.2.4 Identify Tier-one Attributes

Attributes immediately affecting the Prime Query establish the minimal probabilistic model that will support the decision of interest, and are referred to as Tier-one attributes.

A.2.4.1 Tier-one Attributes Table

The Tier-one Attributes have immediate effect on the Prime Query by virtue of their immediate proximity.

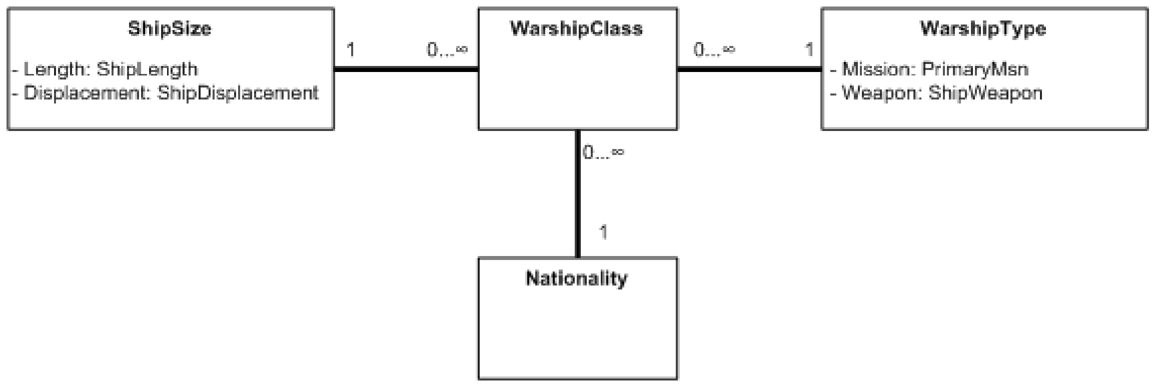
Tier-1 Attributes Table	
ID	Tier-one Attribute
T1	Ship Size
T2	Ship Type
T3	Nationality

A.2.5 Draft Initial Class Diagram

The initial class diagram enables the modeler to visualize the relationships directly affecting the Prime Query via the Tier-one attributes.

A.2.5.1 Initial Class Diagram

The initial class diagram enables the PO Developer to visualize the relationships directly affecting the Prime Query via the Tier-one attributes. It establishes the core of the probabilistic ontology model and is iteratively expanded to incorporate the full specification of requirements.



A.3 Ontology Development Activity

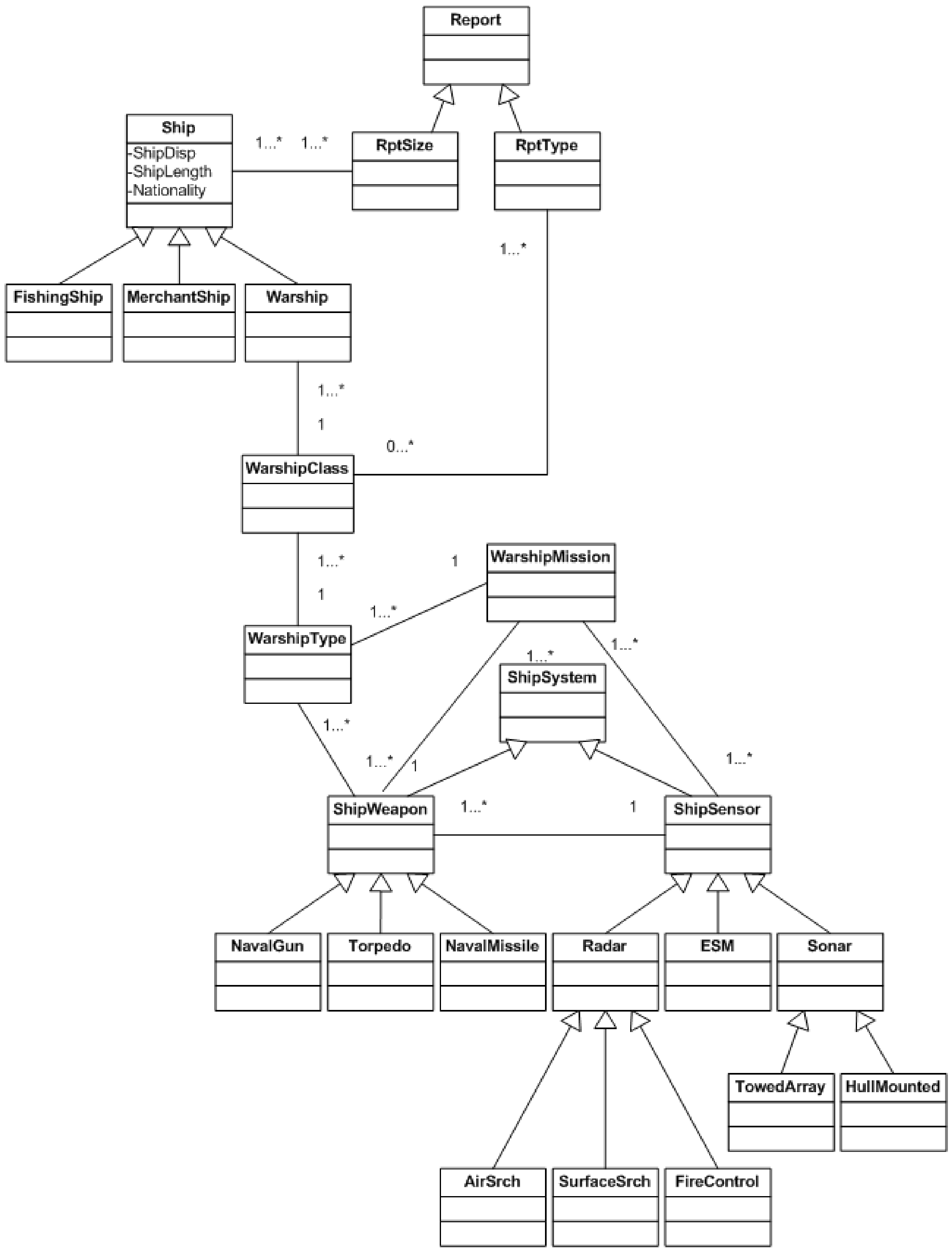
The Ontology Development activity summarizes the non-trivial ontological engineering tasks required to produce a working ontology.

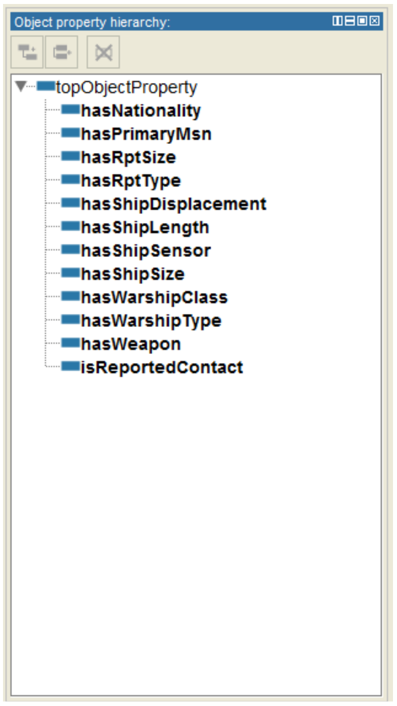
A.3.1 Conduct Ontological Engineering

The set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools, and languages for building ontologies.

A.3.1.1 Taxonomy and Relationships.

A taxonomy is used to organize entity classes and instances through a hierarchical framework based on shared characteristics and serves as a baseline blueprint for the ontology framework.


Report
Ship
-ShipDisp
-ShipLength
-Nationality
1...*
1...*
RptSize
RptType
1...*
FishingShip
MerchantShip
Warship
1...*
1
0...*
WarshipClass
1...*
1
1
WarshipMission
1...*
WarshipType
1...*
1...*
ShipSystem
1...*
1...*
1
1...*
ShipWeapon
1...*
1
ShipSensor
NavalGun
Torpedo
NavalMissile
Radar
ESM
Sonar
TowedArray
HullMounted
AirSrch
SurfaceSrch
FireControl



topObjectProperty
- hasNationality
- hasPrimaryMsn
- hasRptSize
- hasRptType
- hasShipDisplacement
- hasShipLength
- hasShipSensor
- hasShipSize
- hasWarshipClass
- hasWarshipType
- hasWeapon
- isReportedContact

A.3.2 Research Usable Ontologies

Model reuse is defined as the process by which available knowledge is used as input to generate new models.

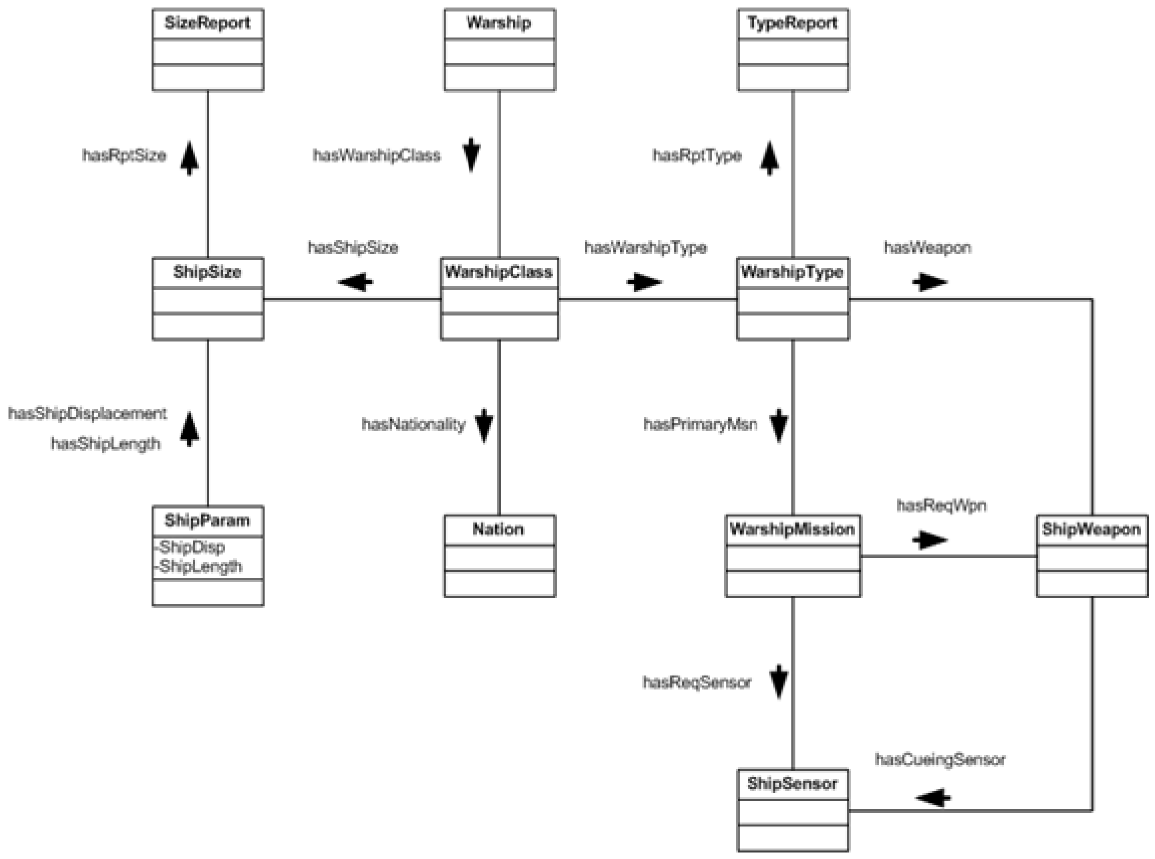
A.3.2.1 Class Table.

The Class Table captures the attributes and relations that describe all of the classes in the ontology.

Class Table				
Class	Associated Object Property	Associated Data Property	Domain	Range
Nation	hasNationality		WarshipClass	Nation
Report	isReportedContact		Ship	Report
Ship	hasNationality isReportedContact		Ship Ship	Nation Report
ShipSensor	hasReqSensor hasCueingSensor		WarshipMission ShipWeapon	ShipSensor ShipSensor
ShipSize	hasRptSize hasShipSize	hasShipDisplacement hasShipLength	Ship Ship ShipSize WarshipClass	Float Float SizeReport ShipSize
ShipWeapon	hasWeapon hasReqWpn		WarshipType WarshipMission	ShipWeapon ShipWeapon
SizeRpt	hasRptSize		ShipSize	SizeReport
TypeRpt	hasRptType		WarshipType	TypeReport
Warship	hasWarshipClass		Warship	WarshipClass
WarshipClass	hasShipSize hasNationality hasWarshipType hasWarshipClass		WarshipClass WarshipClass WarshipClass Warship	ShipSize Nation WarshipType WarshipClass
Warship Mission	hasReqSensor hasPrimaryMsn hasReqWpn		WarshipMission WarshipType WarshipMission	ShipSensor Warship Mission ShipWeapon
Warship Type	hasWarshipType hasRptType hasWeapon hasPrimaryMsn		WarshipClass WarshipType WarshipType WarshipType	WarshipType TypeReport ShipWeapon WarshipMission

A.3.2.2 Complete Class Diagram.

Class diagrams are the mainstay of object-oriented analysis and design and identify the hierarchy of variables germane to the model.

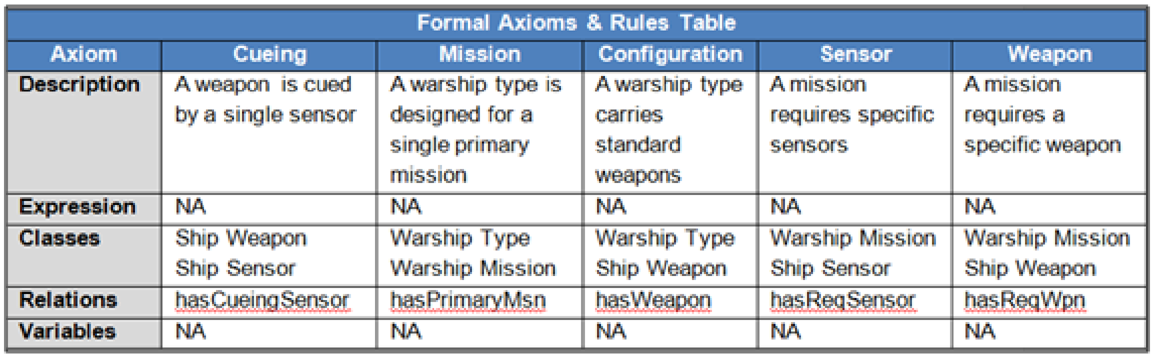


A.3.3 Research Heuristics and Algorithms

Heuristics and algorithms are used to express relationships between classes and individuals within ontologies and probabilistic ontologies.

A.3.3.1 Formal Axiom & Rules Table.

Formal axioms are first-order logical expressions that are always true.



Formal Axioms & Rules Table

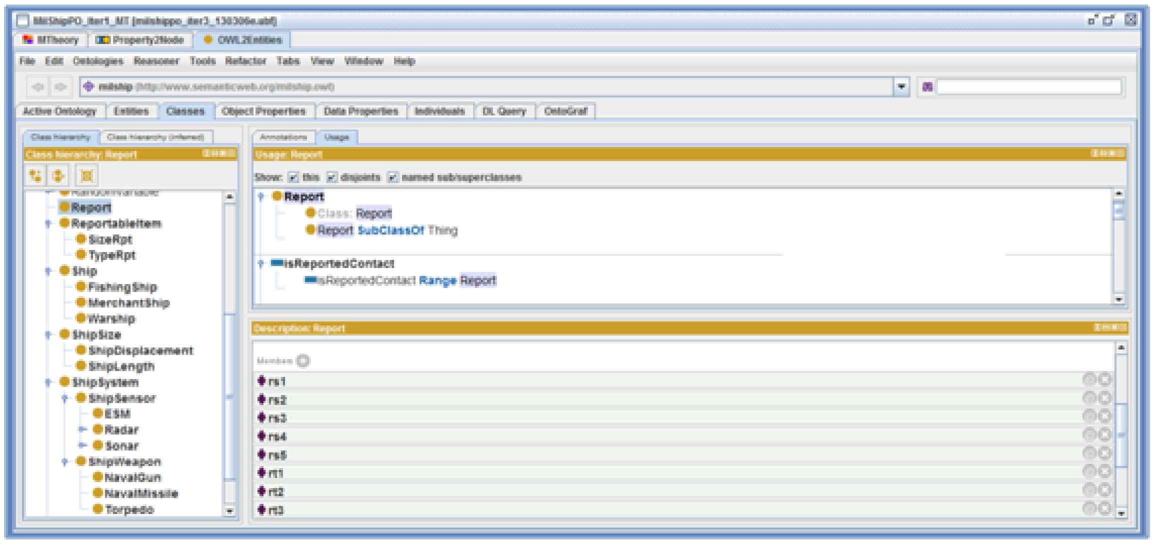
Axiom	Cueing	Mission	Configuration	Sensor	Weapon
Description	A weapon is cued by a single sensor	A warship type is designed for a single primary mission	A warship type carries standard weapons	A mission requires specific sensors	A mission requires a specific weapon
Expression	NA	NA	NA	NA	NA
Classes	Ship Weapon Ship Sensor	Warship Type Warship Mission	Warship Type Ship Weapon	Warship Mission Ship Sensor	Warship Mission Ship Weapon
Relations	hasCueingSensor	hasPrimaryMsn	hasWeapon	hasReqSensor	hasReqWpn
Variables	NA	NA	NA	NA	NA

A.3.4 Implement Ontology Model

At this point the ontology is implemented in a suitable ontology building environment and evaluated for consistency using an appropriate evaluation methodology from the literature.

A.3.4.1 Operational Ontology.

The working ontology serves as the relational framework for the PO when uncertainty is introduced.

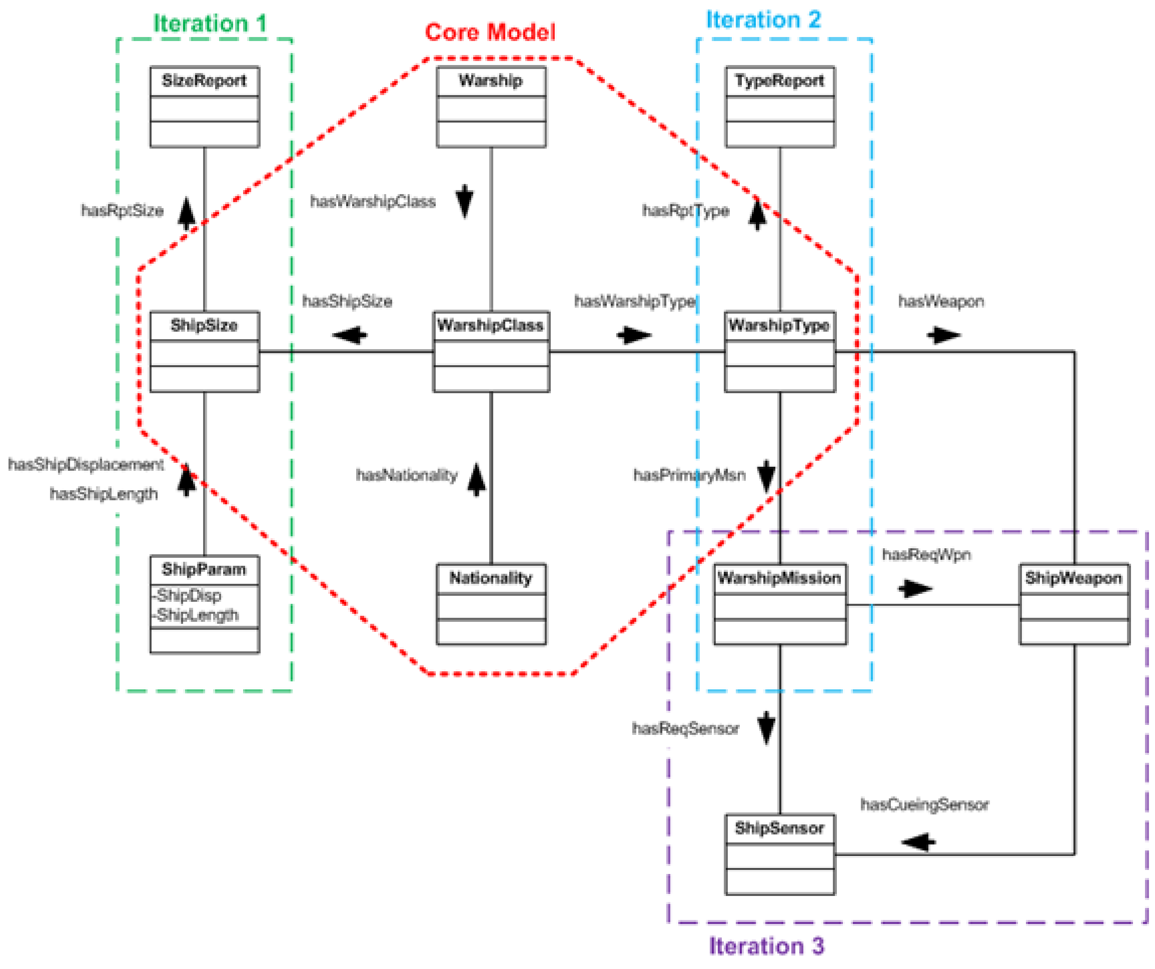


A.4 Probability Incorporation Activity

The Probability Incorporation Activity is the heart of the PODM.

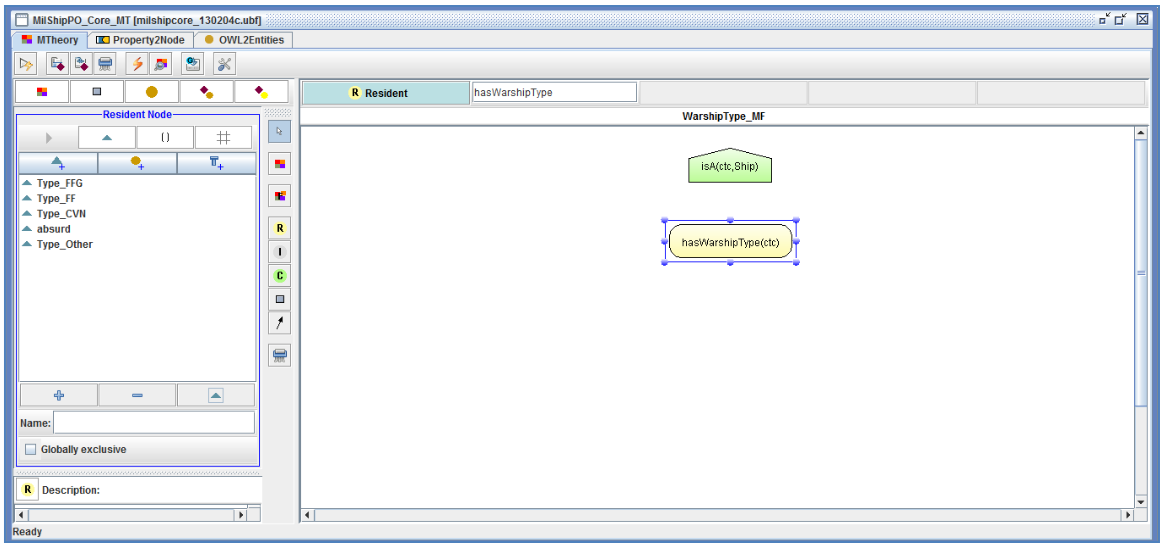
A.4.1 Core Model Generation

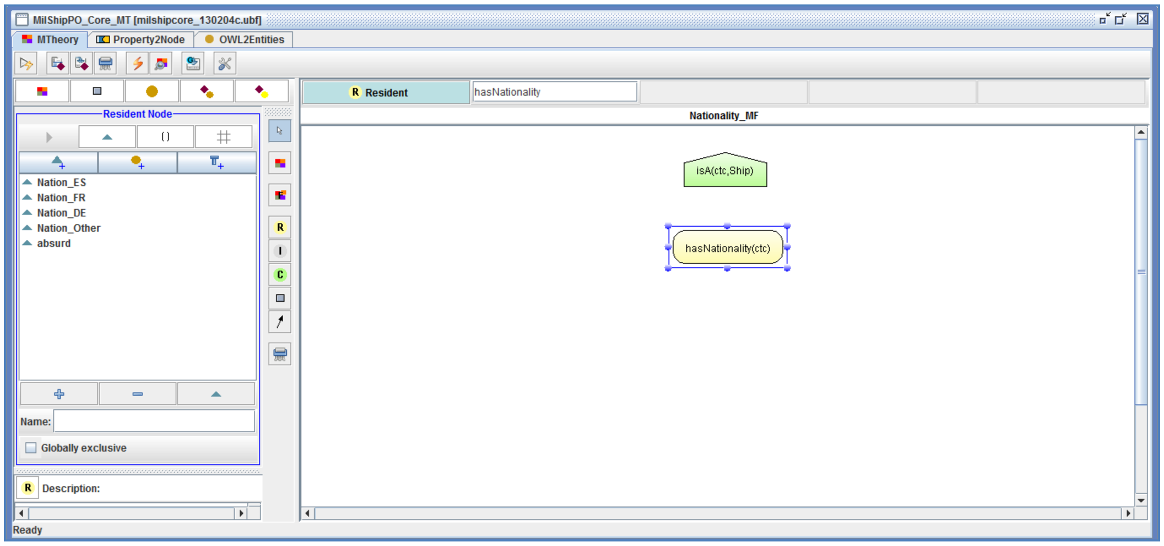
The initial PODM steps for incorporation of probability set the framework for the complete model and establish the spiral Prime Query that will be serviced through inferential reasoning on the Situation-Specific Bayesian Network produced by the model. The PO Construction Process Iteration Plan (Figure 19) shows how the Spiral Core Model for the first spiral of the development process for the Military Ship PO is expanded to satisfy the Objective Statement.

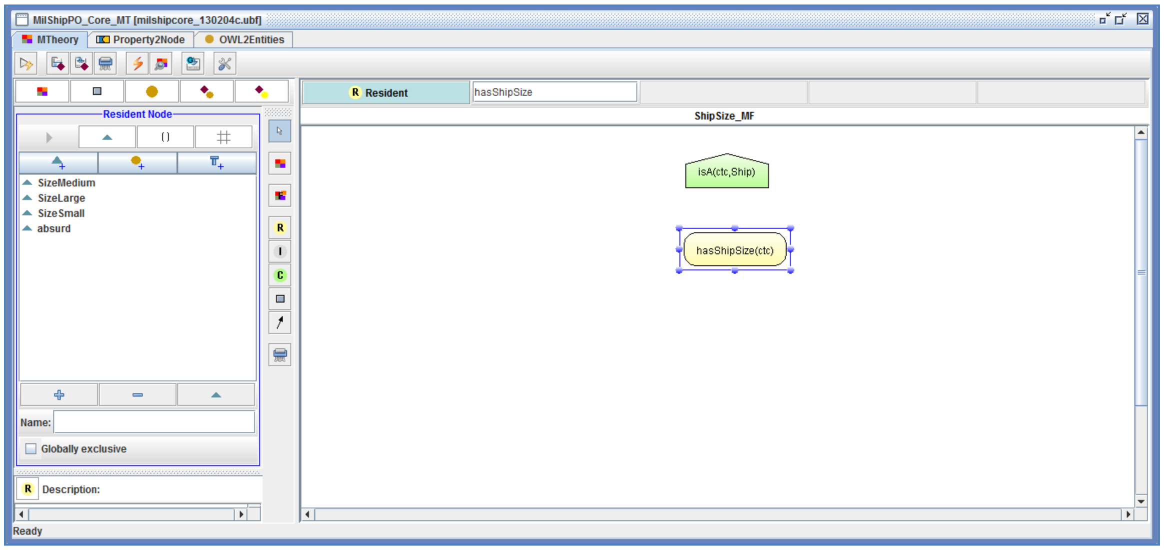


A.4.1.1 Create Model of Primary Query and Tier-1 Attributes

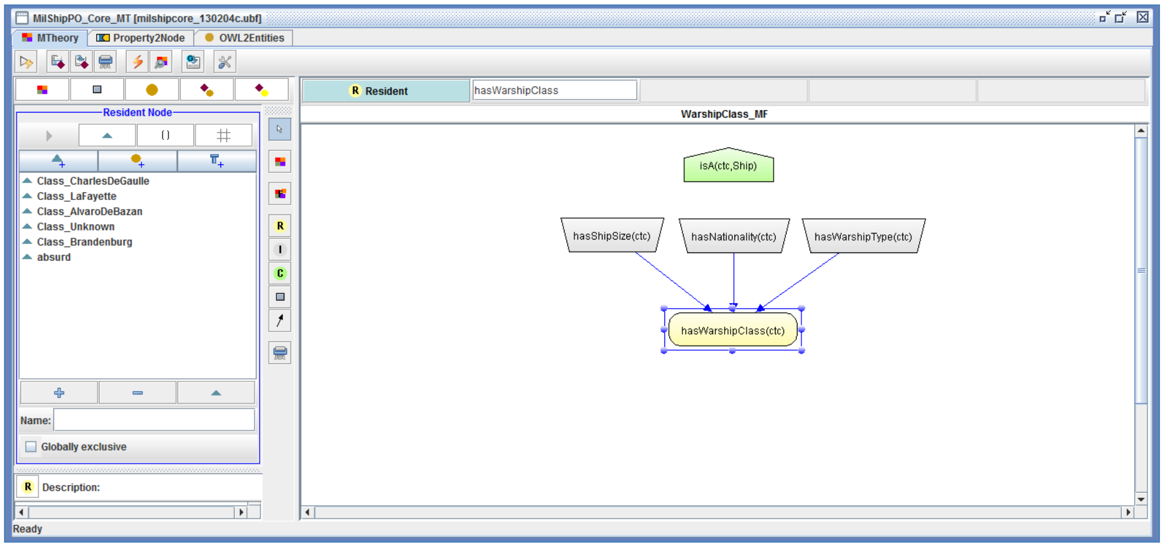
Populate the spiral Prime Query model with conditional probabilities based on the Tier-one attribute relationships.


MilShipPO_Core_MT [milshipcore_130204c.ubf]
MTheory
Property2Node
OWL2Entities
Resident
hasWarshipType
WarshipType_MF
isA(ctc,Ship)
hasWarshipType(ctc)
Resident Node
Type_FFG
Type_FF
Type_CVN
absurd
Type_Other
Name:
Globally exclusive
Description:
Ready

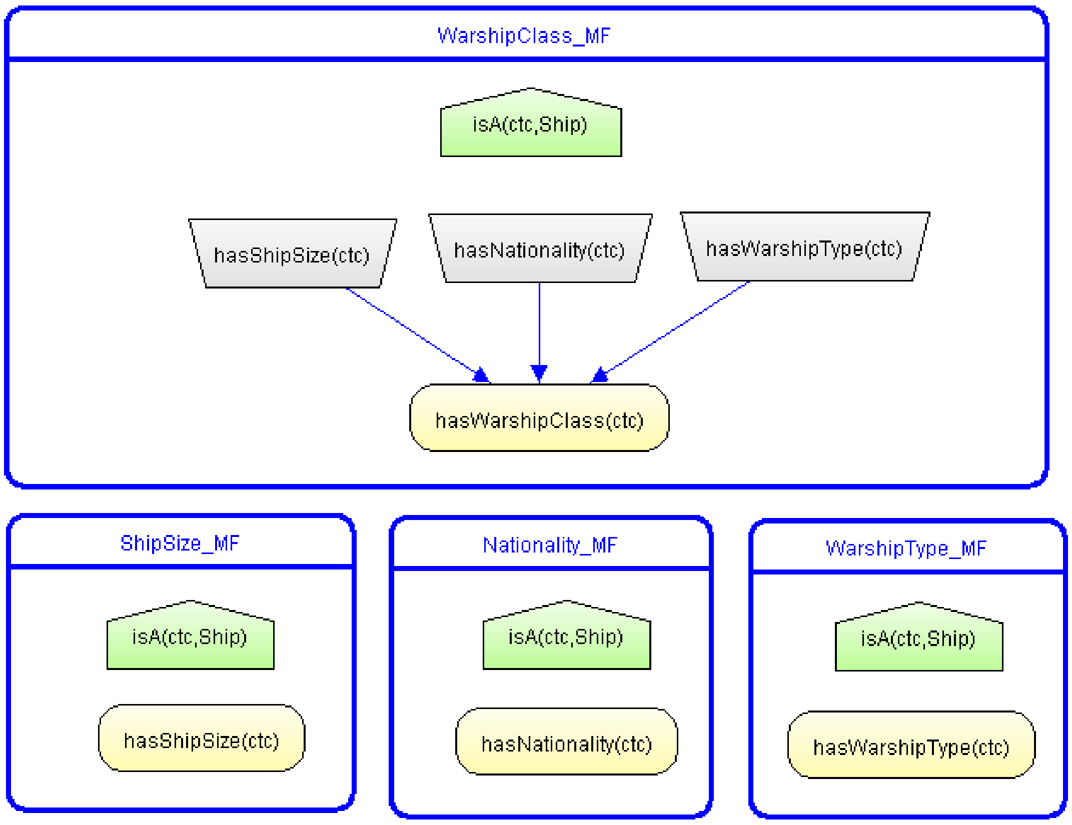

MilShipPO_Core_MT [milshipcore_130204c.ubf]
MTheory
Property2Node
OWL2Entities
Resident
hasNationality
Nationality_MF
isA(ctc,Ship)
hasNationality(ctc)
Resident Node
Nation_ES
Nation_FR
Nation_DE
Nation_Other
absurd
Name:
Globally exclusive
Description:
Ready



SizeMedium
SizeLarge
SizeSmall
absurd



Class_CharlesDeGaulle
Class_LaFayette
Class_AlvaroDeBazan
Class_Unknown
Class_Brandenburg
absurd



A.4.1.2 Populate LPD with Proxy Values

The LPD of the Tier-one attribute nodes are populated with values that allow testing of the core model before all of the relationships are in place.

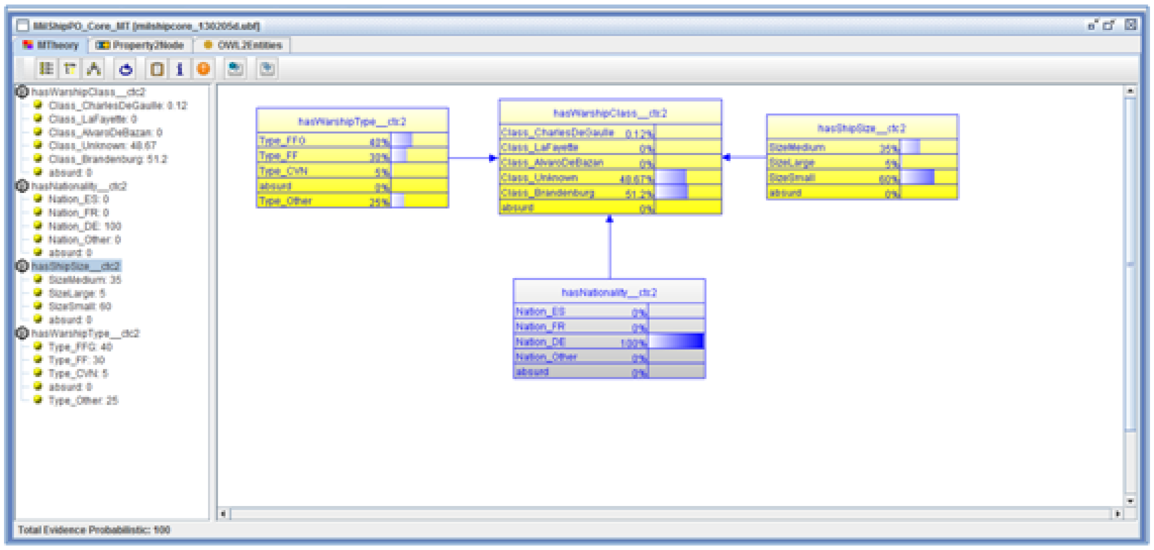
```
Warship Type
[
 Type_FFG = .40,
 Type_FF = .30,
 Type_CVN = .05,
 Type_Other = .25,
 absurd = 0
]
```

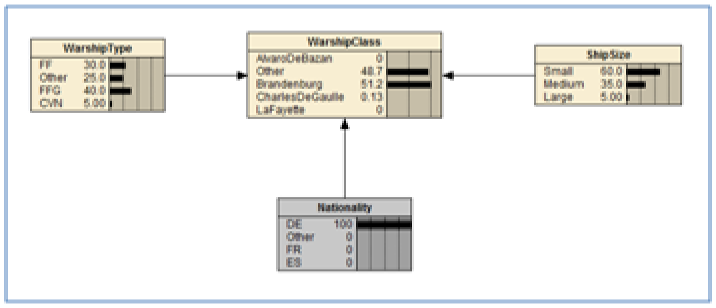
```
Warship Nationality
[
 Nation_ES = .30,
 Nation_FR = .30,
 Nation_DE = .25,
 Nation_Other = .15
]

Warship Size
[
 SizeSmall = .60,
 SizeMedium = .35,
 SizeLarge = .05

]

Warship Class
if any ctc have (hasNationality = Nation_ES ) [
 if any ctc have (hasShipSize=SizeSmall) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=.50,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ]
 ] else if any ctc have (hasShipSize=SizeMedium) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=.80,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.20
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=.50,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else [
 Class_AlvaroDeBazan=.50,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ]
 ] else [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=.50,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ]
 ]
] else if any ctc have (hasNationality=Nation_FR) [
 if any ctc have (hasShipSize=SizeSmall) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.80,
 Class_Unknown=.20
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ]
 ] else if any ctc have (hasShipSize=SizeMedium) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ]
 ] else [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.80,
 Class_LaFayette=0,
 Class_Unknown=.20
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ]
 ]
] else if any ctc have (hasNationality = Nation_DE) [
 if any ctc have (hasShipSize=SizeSmall) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ]
 ] else if any ctc have (hasShipSize=SizeMedium) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.80,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.20
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ]
 ] else [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=.50,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ]
 ]
] else if any ctc have (hasNationality = Nation_Other) [
 if any ctc have (hasShipSize=SizeSmall) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=.33,
 Class_Brandenburg=.33,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.34
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ]
 ] else if any ctc have (hasShipSize=SizeMedium) [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=.33,
 Class_Brandenburg=.33,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.34
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=.50,
 Class_Unknown=.50
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=.33,
 Class_Brandenburg=.33,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=.34
 ]
 ] else [
 if any ctc have (hasWarshipType=Type_FFG) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else if any ctc have (hasWarshipType=Type_FF) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ] else if any ctc have (hasWarshipType=Type_CVN) [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=.50,
 Class_LaFayette=0,
 Class_Unknown=.50
 ] else [
 Class_AlvaroDeBazan=0,
 Class_Brandenburg=0,
 Class_CharlesDeGaulle=0,
 Class_LaFayette=0,
 Class_Unknown=1
 ]
 ]
] else [
 Class_CharlesDeGaulle = .0170,
 Class_LaFayette = .222,
 Class_AlvaroDeBazan = .125,
 Class_Unknown = .485,
 Class_Brandenburg = .151
]
```

A.4.1.3 Create SSBN and Evaluate Logic

Using the software tool, create the SSBN with test evidence. It is useful to create a simple model using a Bayesian network software package to compare testing values.

Variable	Evidence
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE





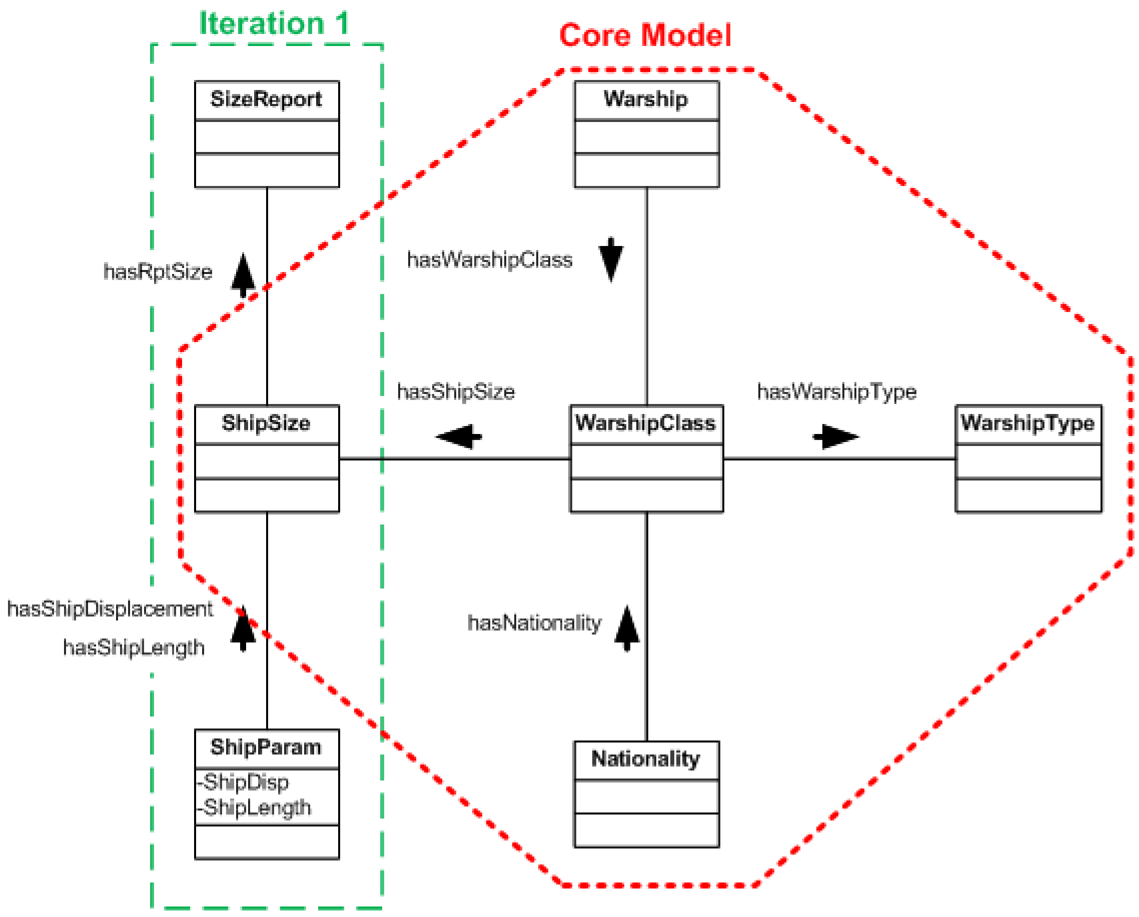
A.4.2 Probabilistic Ontology Construction Process

The iterative steps follow the PO Construction Algorithm, shown above, to systematically expand the initial model while ensuring coherent logic is maintained.

A.4.2.1 Iteration 1

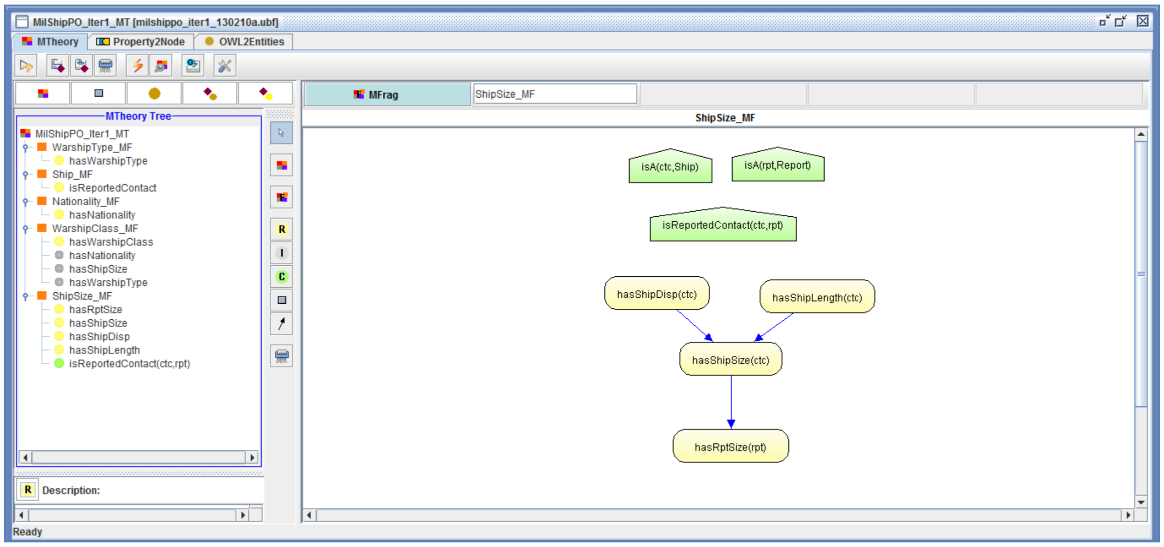
A.4.2.1.1 Select and Decompose a Related Class into its Sub-classes and Attributes

Decompose one of the related classes identified from the class diagram into its subcomponents, classes and attributes.



A.4.2.1.2 Create Representation for the Selected Related Class

Using the information assembled in the decomposition, build the MFrag for the new class.



A.4.2.1.3 Populate LPDs of Related Class Attributes

LPDs for the new nodes may be in the form of a conditional probability table (CPT) or logic statements, depending on the probabilistic ontology software utilized.

```
 Warship Reported Size
if any ctc have ( hasShipSize = SizeSmall ) [
 Rpt_SizeLarge = .10,
 Rpt_SizeMedium = .30,
 Rpt_SizeSmall = .60
] else if any ctc have ( hasShipSize = SizeMedium ) [
 Rpt_SizeLarge = .15,
 Rpt_SizeMedium = .60,
 Rpt_SizeSmall = .25
] else if any ctc have ( hasShipSize = SizeLarge ) [
 Rpt_SizeLarge = .60,
 Rpt_SizeMedium = .30,
 Rpt_SizeSmall = .10
] else [
 Rpt_SizeLarge = .161,
 Rpt_SizeMedium = .475,
 Rpt_SizeSmall = .364
]
```

Ship Displacement

```
[
 Disp5kto10k = .35,
 DispLess5k = .60,
 DispGreater10k = .05
]
```

Ship Length

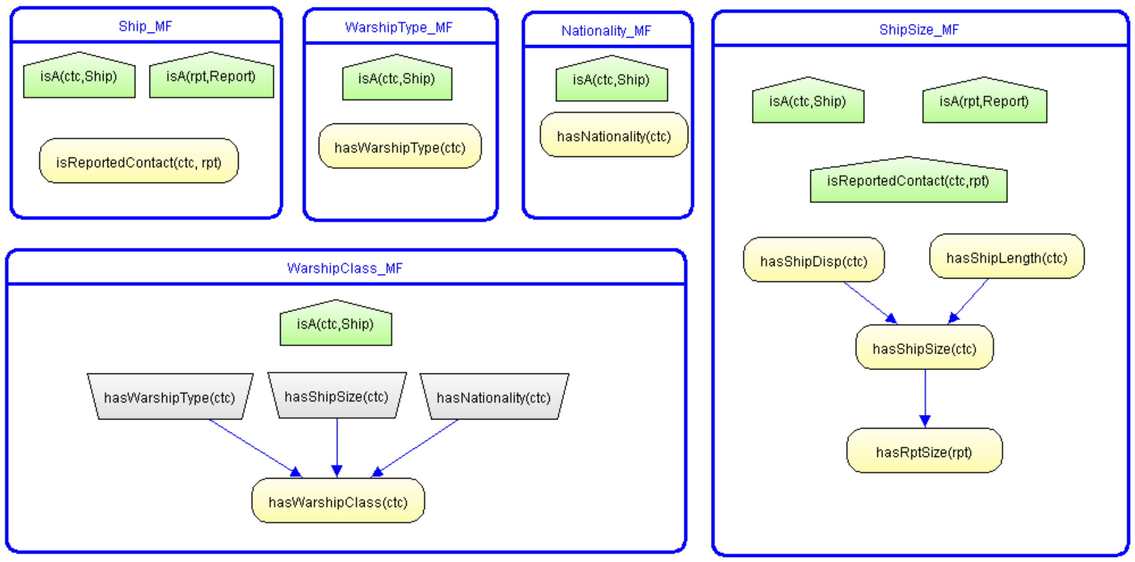
```
[
 LengthGreater150 = .05,
 Length125to150 = .35,
 LengthLess125 = .60
]
```

A.4.2.1.4 Update Existing Model Relationships and LPDs

All legacy nodes affected by addition of the new attribute must be updated to reflect conditional probabilities expressing the relationships associated with the new node.

Ship Size (updated)

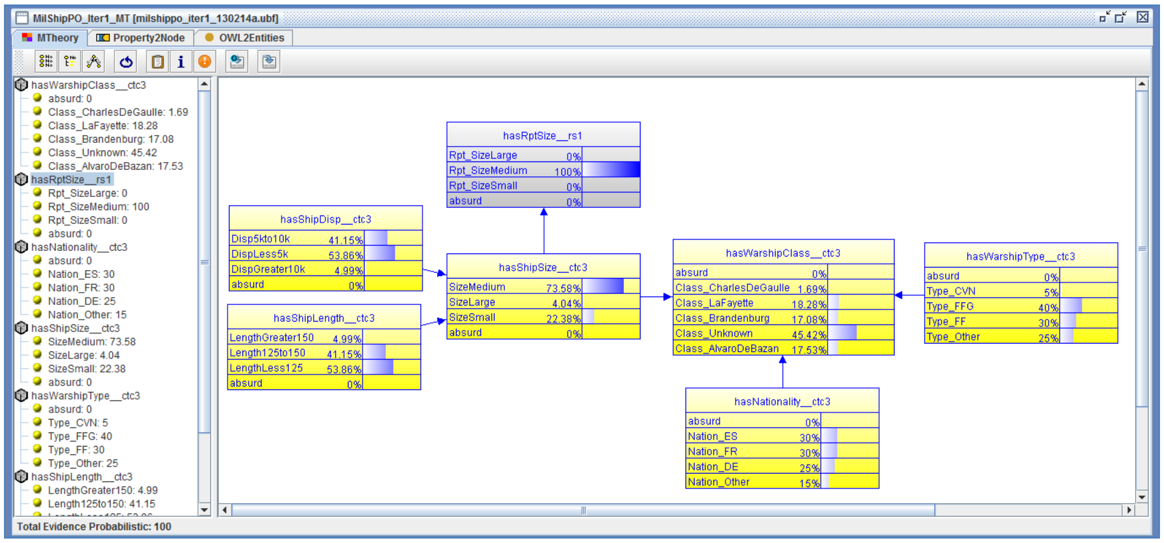
```
if any ctc have ( hasShipDisp = DispLess5k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .10,
 SizeLarge = 0,
 SizeSmall = .90
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ]
] else if any ctc have ( hasShipDisp = Disp5kto10k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ]
] else if any ctc have ( hasShipDisp = DispGreater10k ) [
 if any ctc have ( hasShipLength = LengthLess125 ) [
 SizeMedium = .90,
 SizeLarge = .05,
 SizeSmall = .05
 ] else if any ctc have ( hasShipLength = Length125to150 ) [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ] else [
 SizeMedium = .10,
 SizeLarge = .90,
 SizeSmall = 0
 ]
] else [
 SizeMedium = .582,
 SizeLarge = .0639,
 SizeSmall = .354
]
```



A.4.2.1.5 Create SSBN and Evaluate Logic

After each iteration of model development is complete, query the model to produce an SSBN for the Prime Query using the software.

Variable	Evidence
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true



hasWarshipClass__ctc3
- absurd: 0
- Class_CharlesDeGaulle: 1.69
- Class_LaFayette: 18.28
- Class_Brandenburg: 17.08
- Class_Unknown: 45.42
- Class_AlvaroDeBazan: 17.53

hasRptSize__rs1
- Rpt_SizeLarge: 0
- Rpt_SizeMedium: 100
- Rpt_SizeSmall: 0
- absurd: 0

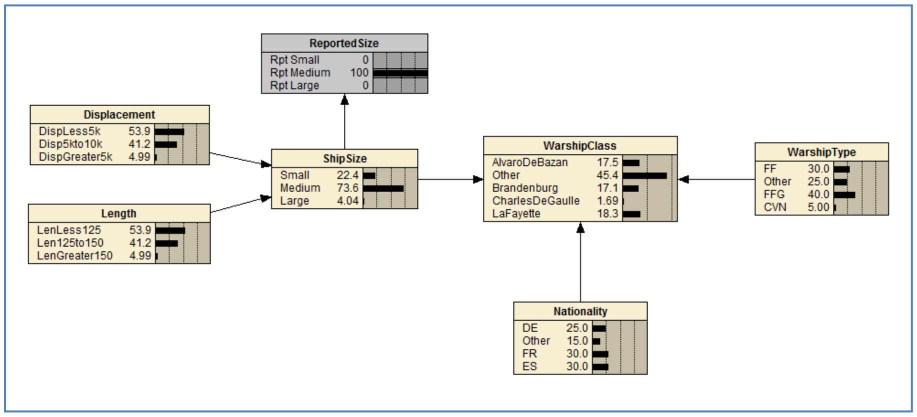
hasNationality__ctc3
- absurd: 0
- Nation_ES: 30
- Nation_FR: 30
- Nation_DE: 25
- Nation_Other: 15

hasShipSize__ctc3
- SizeMedium: 73.58
- SizeLarge: 4.04
- SizeSmall: 22.38
- absurd: 0

hasWarshipType__ctc3
- absurd: 0
- Type_CVN: 5
- Type_FFG: 40
- Type_FF: 30
- Type_Other: 25

hasShipLength__ctc3
- LengthGreater150: 4.99
- Length125to150: 41.15

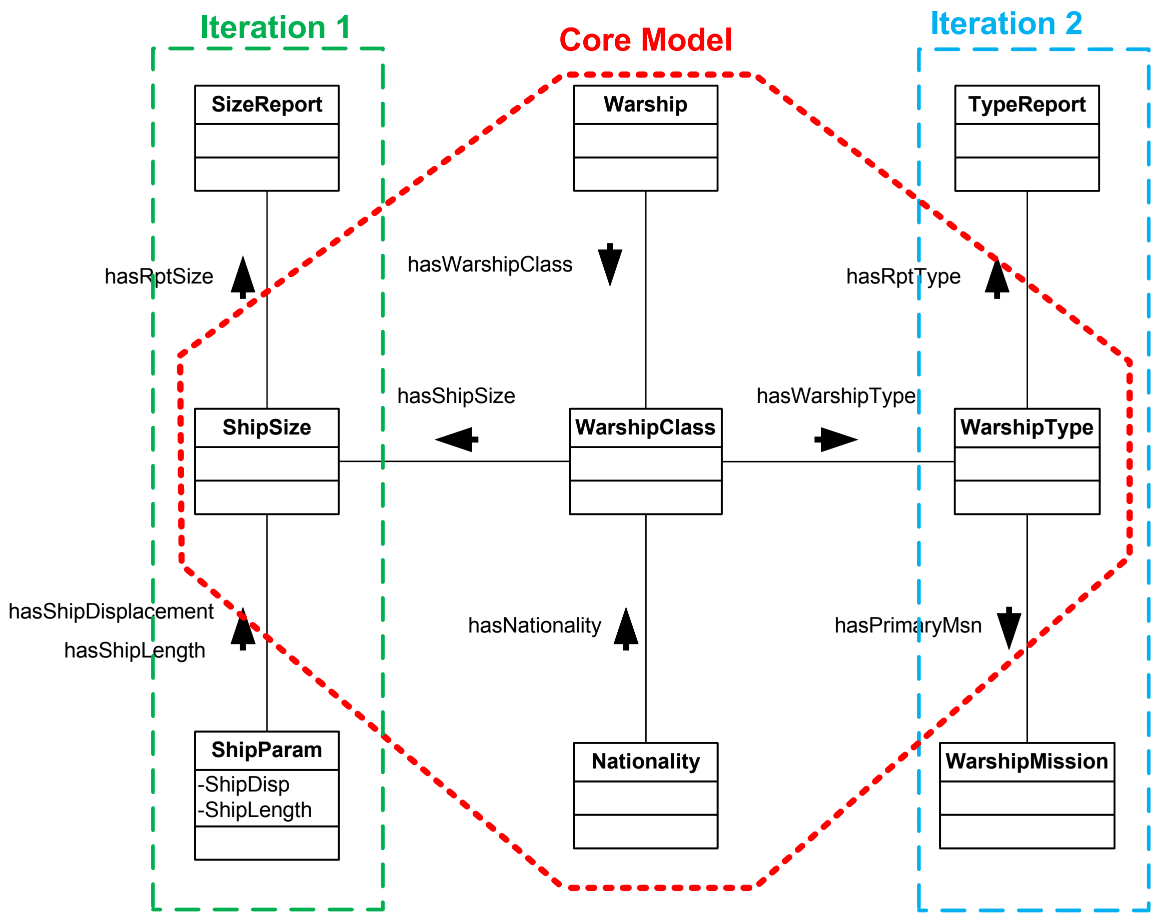
Total Evidence Probabilistic: 100



A.4.2.2 Iteration 2

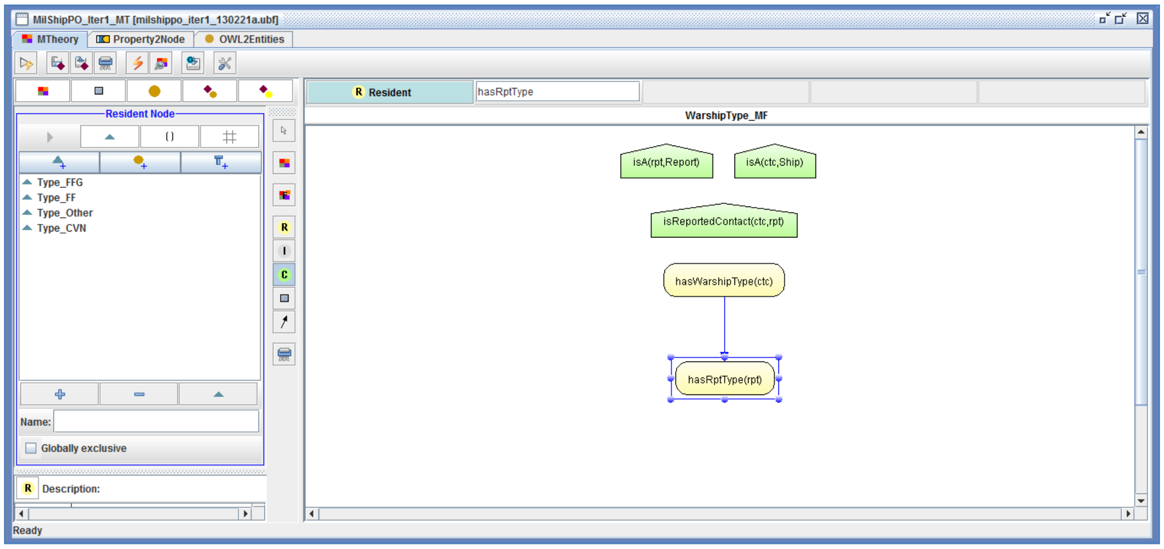
A.4.2.2.1 Select and Decompose a Related Class into its Sub-classes and Attributes

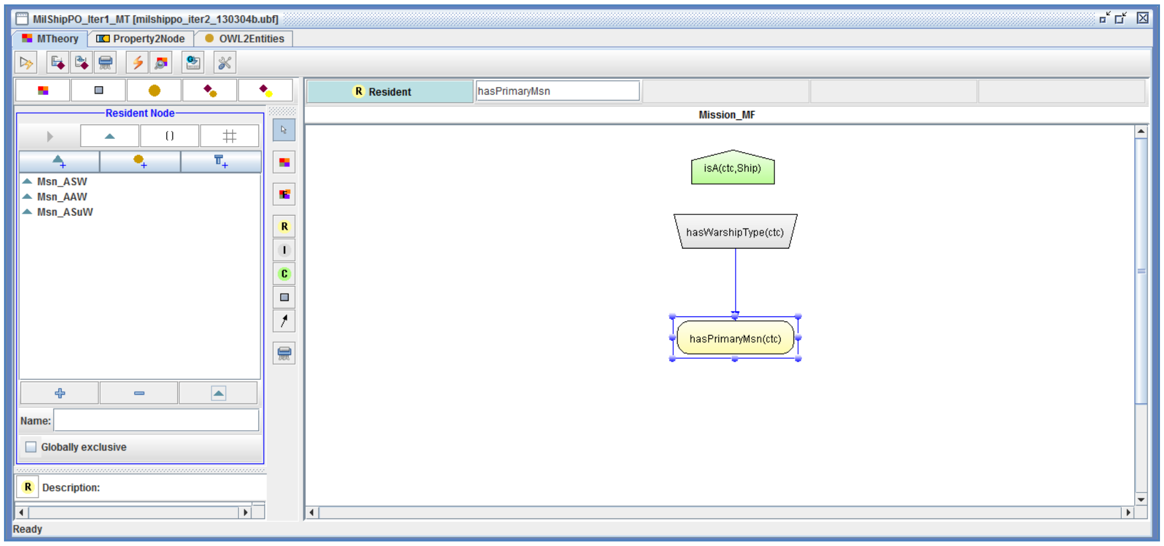
Decompose one of the related classes identified from the class diagram into its subcomponents, classes and attributes.



A.4.2.2.2 Create Representation for the Selected Related Class

Using the information assembled in the decomposition, build the MFrag for the new class.





A.4.2.2.3 Populate LPDs of Related Class Attributes

LPDs for the new nodes may be in the form of a conditional probability table (CPT) or logic statements, depending on the probabilistic ontology software utilized.

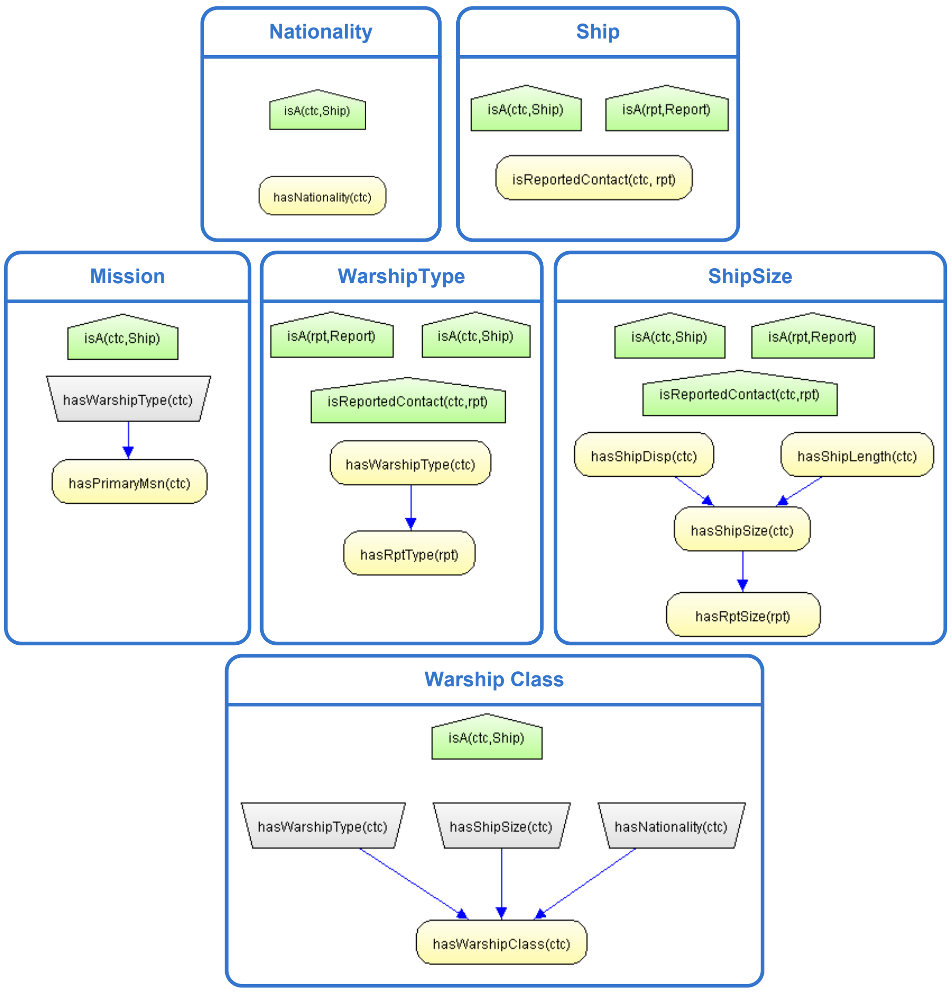
Reported Ship Type
```
if any ctc have ( hasWarshipType = Type_CVN ) [
 Rpt_TypeCVN = .60,
 Rpt_TypeFF = .10,
 Rpt_TypeFFG = .25,
 Rpt_TypeOther = .05
] else if any ctc have ( hasWarshipType = Type_FF ) [
 Rpt_TypeCVN = .05,
 Rpt_TypeFF = .60,
 Rpt_TypeFFG = .30,
 Rpt_TypeOther = .05
] else if any ctc have ( hasWarshipType = Type_FFG ) [
 Rpt_TypeCVN = .05,
 Rpt_TypeFF = .30,
 Rpt_TypeFFG = .60,
 Rpt_TypeOther = .05
] else if any ctc have ( hasWarshipType = Type_Other ) [
 Rpt_TypeCVN = .10,
 Rpt_TypeFF = .30,
 Rpt_TypeFFG = .30,
 Rpt_TypeOther = .30
] else [
 Rpt_TypeCVN = .091,
 Rpt_TypeFF = .38,
 Rpt_TypeFFG = .417,
 Rpt_TypeOther = .112
]
```

Warship Primary Mission
```
if any ctc have ( hasWarshipType = Type_FF ) [
 Msn_ASW = .45,
 Msn_AAW = 0,
 Msn_ASuW = .55
] else if any ctc have ( hasWarshipType = Type_Other ) [
 Msn_ASW = .33,
 Msn_AAW = .33,
 Msn_ASuW = .34
] else if any ctc have ( hasWarshipType = Type_FFG ) [
 Msn_ASW = .25,
 Msn_AAW = .40,
 Msn_ASuW = .35
] else if any ctc have ( hasWarshipType = Type_CVN ) [
 Msn_ASW = .10,
 Msn_AAW = .60,
 Msn_ASuW = .30
] else [
 Msn_ASW = .322,
 Msn_AAW = .273,
 Msn_ASuW = .405
]
```

A.4.2.2.4 Update Existing Model Relationships and LPDs

All legacy nodes affected by addition of the new attribute must be updated to reflect conditional probabilities expressing the relationships associated with the new node.

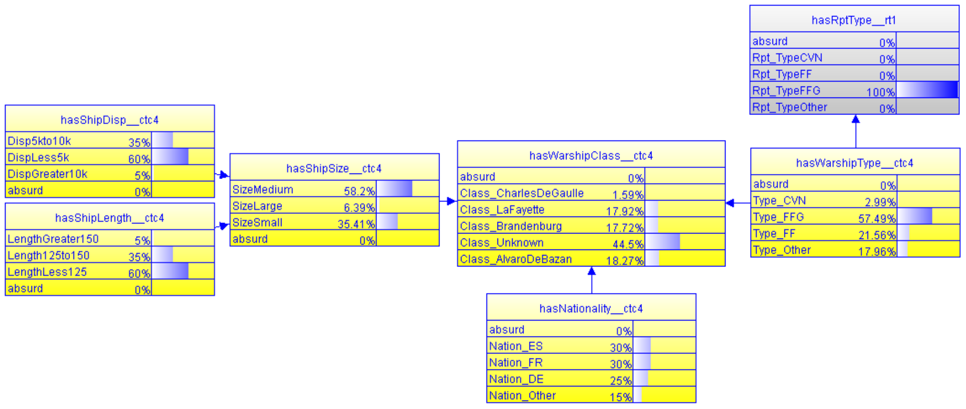
None for this iteration.



A.4.2.2.5 Create SSBN and Evaluate Logic

After each iteration of model development is complete, query the model to produce an SSBN for the Prime Query using the software.

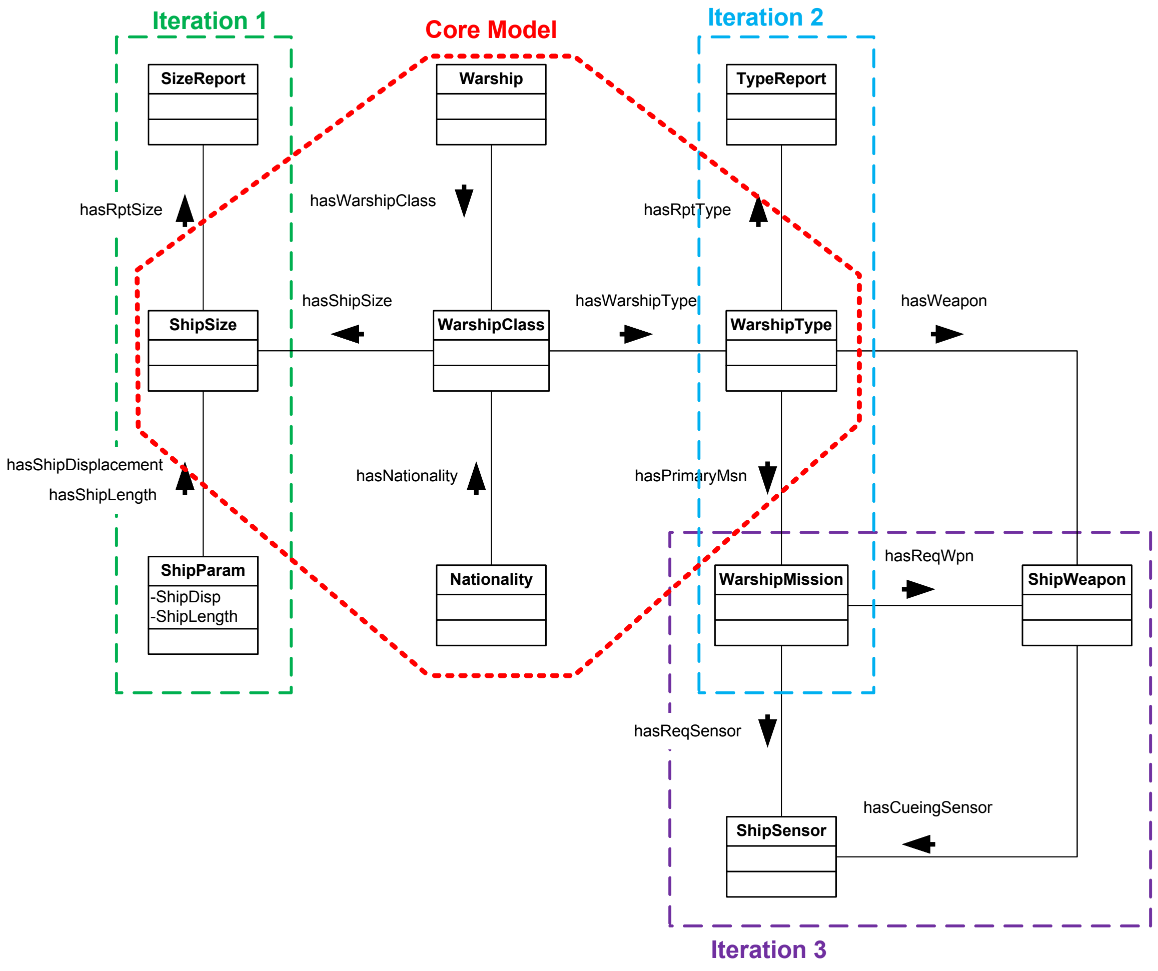
Variable	Evidence
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc4, rt1	isReportedContact(ctc4(Ship), rt1(Report))=true



A.4.2.3 Iteration 3

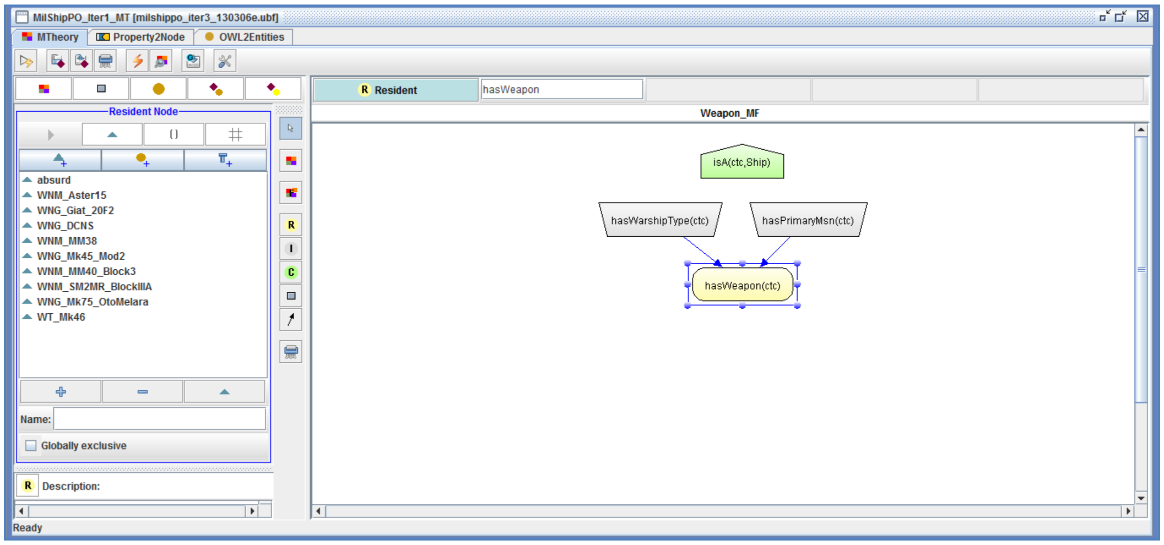
A.4.2.3.1 Select and Decompose a Related Class into its Sub-classes and Attributes

Decompose one of the related classes identified from the class diagram into its subcomponents, classes and attributes.



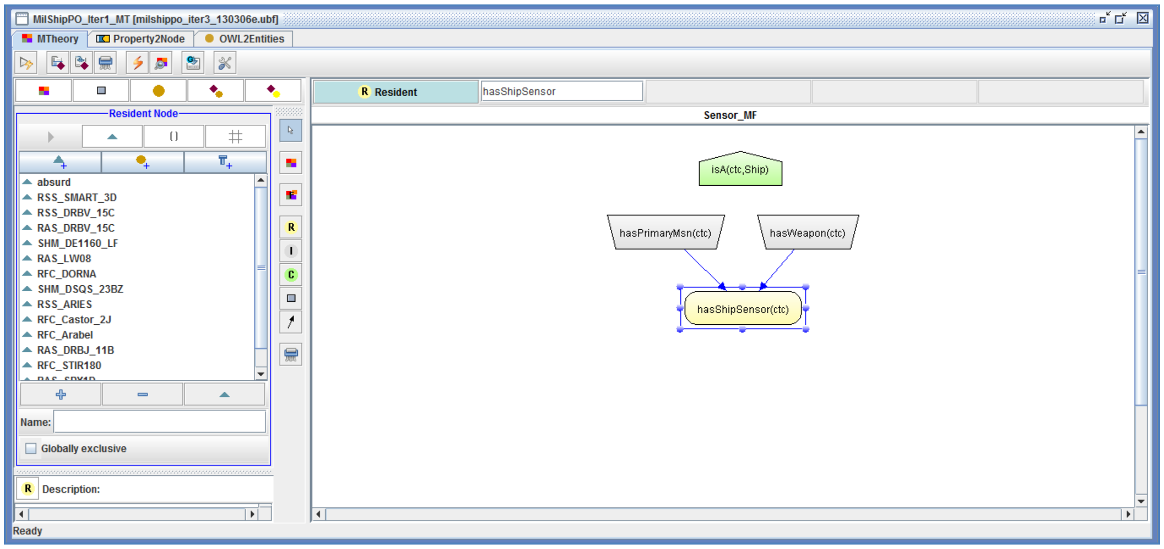
A.4.2.3.2 Create Representation for the Selected Related Class

Using the information assembled in the decomposition, build the MFrag for the new class.



Resident: hasWeapon

absurd
WNM_Aster15
WNG_Giat_20F2
WNG_DCNS
WNM_MM38
WNG_Mk45_Mod2
WNM_MM40_Block3
WNM_SM2MR_BlockIIIA
WNG_Mk75_OtoMelara
WT_Mk46



Resident: hasShipSensor

absurd
RSS_SMART_3D
RSS_DRBV_15C
RAS_DRBV_15C
SHM_DE1160_LF
RAS_LW08
RFC_DORNA
SHM_DSQS_23BZ
RSS_ARIES
RFC_Castor_2J
RFC_Arabel
RAS_DRBJ_11B
RFC_STIR180

A.4.2.3.3 Populate LPDs of Related Class Attributes

LPDs for the new nodes may be in the form of a conditional probability table (CPT) or logic statements, depending on the probabilistic ontology software utilized.

Warship Weapon

```
if any ctc have ( hasWarshipType = Type_FF ) [
 if any ctc have ( hasPrimaryMsn = Msn_AAW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 1,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASuW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = .50,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = .50,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASW) [
 absurd = 1,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else [
 absurd = 0,
 WNG_Mk45_Mod2 = .127,
 WNG_Giat_20F2 = .0668,
 WNM_Aster15 = .0540,
 WNM_MM38 = .0796,
 WNG_DCNS = .127,
 WNM_SM2MR_BlockIIIA = .0859,
 WT_Mk46 = .2007,
 WNG_Mk75_OtoMelara = .145,
 WNM_MM40_Block3 = .114
 ]
]else if any ctc have ( hasWarshipType = Type_Other ) [
 if any ctc have ( hasPrimaryMsn = Msn_AAW) [
 absurd = 0,
 WNM_Aster15 = .170,
 WNG_Giat_20F2 = .160,
 WNG_DCNS = .160,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = .170,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = .170,
 WNG_Mk75_OtoMelara = .170,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASuW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = .160,
 WNG_DCNS = .160,
 WNM_MM38 = .170,
 WNG_Mk45_Mod2 = .170,
 WNM_MM40_Block3 = .170,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = .170,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 1
 ] else [
 absurd = 0,
 WNG_Mk45_Mod2 = .127,
 WNG_Giat_20F2 = .0668,
 WNM_Aster15 = .0540,
 WNM_MM38 = .0796,
 WNG_DCNS = .127,
 WNM_SM2MR_BlockIIIA = .0859,
 WT_Mk46 = .2007,
 WNG_Mk75_OtoMelara = .145,
 WNM_MM40_Block3 = .114
 ]
] else if any ctc have ( hasWarshipType = Type_FFG ) [
 if any ctc have ( hasPrimaryMsn = Msn_AAW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = .33,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = .34,
 WNG_Mk75_OtoMelara = .33,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASuW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = .34,
 WNG_Mk45_Mod2 = .33,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = .33,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASW) [
 absurd = 0,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 1
 ] else [
 absurd = 0,
 WNG_Mk45_Mod2 = .127,
 WNG_Giat_20F2 = .0668,
 WNM_Aster15 = .0540,
 WNM_MM38 = .0796,
 WNG_DCNS = .127,
 WNM_SM2MR_BlockIIIA = .0859,
 WT_Mk46 = .2007,
 WNG_Mk75_OtoMelara = .145,
 WNM_MM40_Block3 = .114
 ]
] else if any ctc have ( hasWarshipType = Type_CVN ) [
 if any ctc have ( hasPrimaryMsn = Msn_AAW) [
 absurd = 0,
 WNM_Aster15 = .50,
 WNG_Giat_20F2 = .50,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASuW) [
 absurd = 0,
 WNM_Aster15 = .50,
 WNG_Giat_20F2 = .50,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else if any ctc have ( hasPrimaryMsn = Msn_ASW) [
 absurd = 1,
 WNM_Aster15 = 0,
 WNG_Giat_20F2 = 0,
 WNG_DCNS = 0,
 WNM_MM38 = 0,
 WNG_Mk45_Mod2 = 0,
 WNM_MM40_Block3 = 0,
 WNM_SM2MR_BlockIIIA = 0,
 WNG_Mk75_OtoMelara = 0,
 WT_Mk46 = 0
 ] else [
 absurd = 0,
 WNG_Mk45_Mod2 = .127,
 WNG_Giat_20F2 = .0668,
 WNM_Aster15 = .0540,
 WNM_MM38 = .0796,
 WNG_DCNS = .127,
 WNM_SM2MR_BlockIIIA = .0859,
 WT_Mk46 = .2007,
 WNG_Mk75_OtoMelara = .145,
 WNM_MM40_Block3 = .114
 ]
] else [
 WNG_Mk45_Mod2 = .127,
 WNG_Giat_20F2 = .0668,
 WNM_Aster15 = .0540,
 WNM_MM38 = .0796,
 WNG_DCNS = .127,
 WNM_SM2MR_BlockIIIA = .0859,
 WT_Mk46 = .2007,
 WNG_Mk75_OtoMelara = .145,
 WNM_MM40_Block3 = .114
]
```

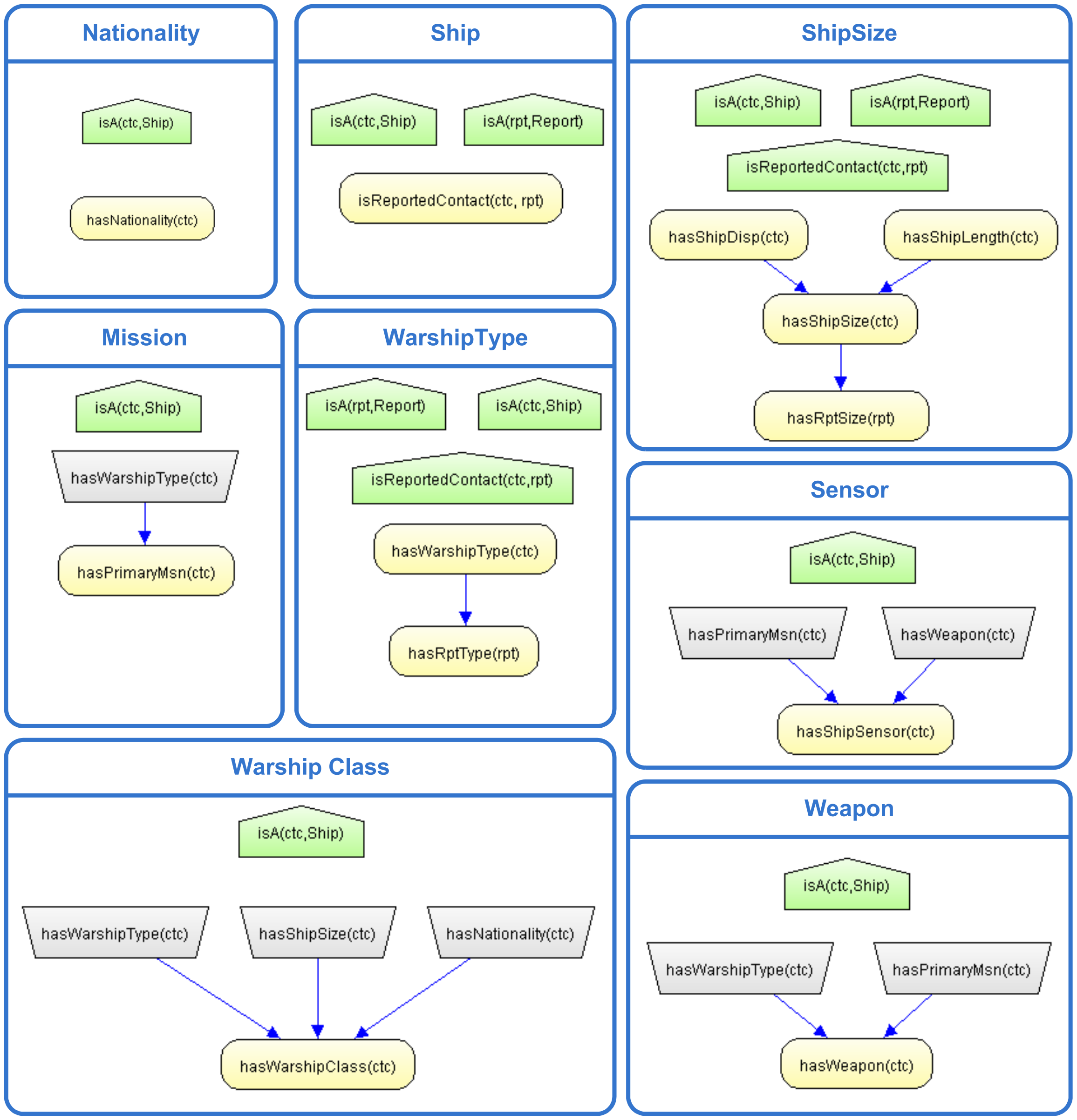
Warship Sensor

```
if any ctc have ( hasPrimaryMsn = Msn_AAW ) [
 if any ctc have ( hasWeapon = WNG_Mk45_Mod2) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = .50,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = .50,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WT_Mk46) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_Mk75_OtoMelara) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = .50,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = .50,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNG_Giat_20F2) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = .50,
 RAS_DRBJ_11B = .50,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_DCNS) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = .50,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = .50,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNM_SM2MR_BlockIIIA) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = .50,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = .50,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNM_MM38) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNM_Aster15) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = .50,
 RAS_DRBJ_11B = .50,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon =WNM_MM40_Block3) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else [
 absurd = 0,
 RSS_DRBV_15C = .0237,
 RFC_Arabel = .0827,
 SHM_DE1160_LF = .109,
 RFC_STIR180 = .104,
 RAS_DRBV_15C = .0160,
 RSS_DRBN34 = .106,
 RAS_SPY1D = .0770,
 RSS_SMART_3D = .0708,
 RFC_Castor_2J = .113,
 RSS_ARIES = .01,
 SHM_DSQS_23BZ = .109,
 RFC_DORNA = .098,
 RAS_LW08 = .0428,
 RAS_DRBJ_11B = .0380
 ]
] else if any ctc have ( hasPrimaryMsn = Msn_ASuW ) [
 if any ctc have ( hasWeapon = WNG_Mk45_Mod2) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = .50,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = .50,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WT_Mk46) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_Mk75_OtoMelara) [
 absurd = 0,
 RSS_SMART_3D = .50,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = .50,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNG_Giat_20F2) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = .50,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = .50,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_DCNS) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = .50,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = .50
 ] else if any ctc have ( hasWeapon = WNM_SM2MR_BlockIIIA) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNM_MM38) [
 absurd = 0,
 RSS_SMART_3D = .50,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = .50,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNM_Aster15) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = .50,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = .50,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon =WNM_MM40_Block3) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = .50,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = .50
 ] else [
 absurd = 0,
 RSS_DRBV_15C = .0237,
 RFC_Arabel = .0827,
 SHM_DE1160_LF = .109,
 RFC_STIR180 = .104,
 RAS_DRBV_15C = .0160,
 RSS_DRBN34 = .106,
 RAS_SPY1D = .0770,
 RSS_SMART_3D = .0708,
 RFC_Castor_2J = .113,
 RSS_ARIES = .01,
 SHM_DSQS_23BZ = .109,
 RFC_DORNA = .098,
 RAS_LW08 = .0428,
 RAS_DRBJ_11B = .0380
 ]
] else if any ctc have ( hasPrimaryMsn = Msn_ASW ) [
 if any ctc have ( hasWeapon = WNG_Mk45_Mod2) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WT_Mk46) [
 absurd = 0,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = .50,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = .50,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_Mk75_OtoMelara) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNG_Giat_20F2) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNG_DCNS) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNM_SM2MR_BlockIIIA) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else if any ctc have ( hasWeapon = WNM_MM38) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon = WNM_Aster15) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
 ] else if any ctc have ( hasWeapon =WNM_MM40_Block3) [
 absurd = 1,
 RSS_SMART_3D = 0,
 RSS_DRBV_15C = 0,
 RAS_DRBV_15C = 0,
 SHM_DE1160_LF = 0,
 RAS_LW08 = 0,
 RFC_DORNA = 0,
 SHM_DSQS_23BZ = 0,
 RSS_ARIES = 0,
 RFC_Castor_2J = 0,
 RFC_Arabel = 0,
 RAS_DRBJ_11B = 0,
 RFC_STIR180 = 0,
 RAS_SPY1D = 0,
 RSS_DRBN34 = 0
] else [
 absurd = 0,
 RSS_DRBV_15C = .0237,
 RFC_Arabel = .0827,
 SHM_DE1160_LF = .109,
 RFC_STIR180 = .104,
 RAS_DRBV_15C = .0160,
 RSS_DRBN34 = .106,
 RAS_SPY1D = .0770,
 RSS_SMART_3D = .0708,
 RFC_Castor_2J = .113,
 RSS_ARIES = .01,
 SHM_DSQS_23BZ = .109,
 RFC_DORNA = .098,
 RAS_LW08 = .0428,
 RAS_DRBJ_11B = .0380
 ]
] else [
 RSS_DRBV_15C = .0237,
 RFC_Arabel = .0827,
 SHM_DE1160_LF = .109,
 RFC_STIR180 = .104,
 RAS_DRBV_15C = .0160,
 RSS_DRBN34 = .106,
 RAS_SPY1D = .0770,
 RSS_SMART_3D = .0708,
 RFC_Castor_2J = .113,
 RSS_ARIES = .01,
 SHM_DSQS_23BZ = .109,
 RFC_DORNA = .098,
 RAS_LW08 = .0428,
 RAS_DRBJ_11B = .0380
]
```

A.4.2.3.4 Update Existing Model Relationships and LPDs

All legacy nodes affected by addition of the new attribute must be updated to reflect conditional probabilities expressing the relationships associated with the new node.

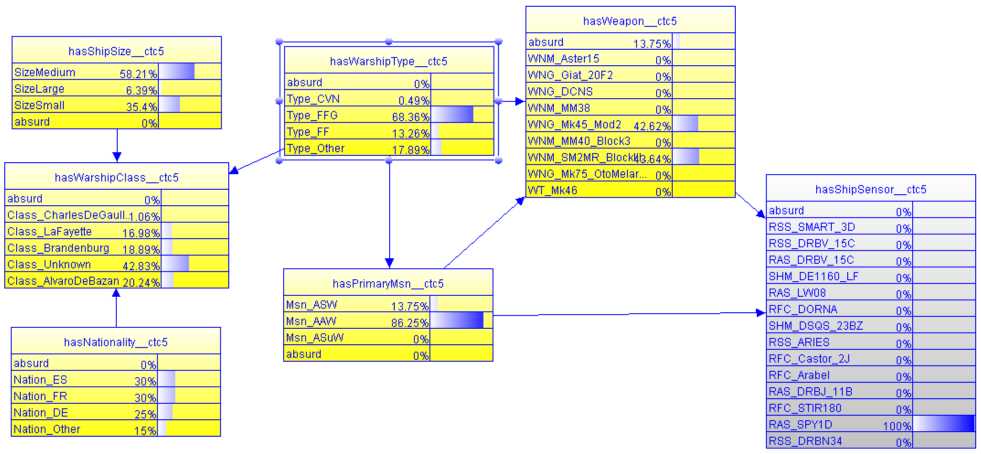
None at this iteration.



A.4.2.3.5 Create SSBN and Evaluate Logic

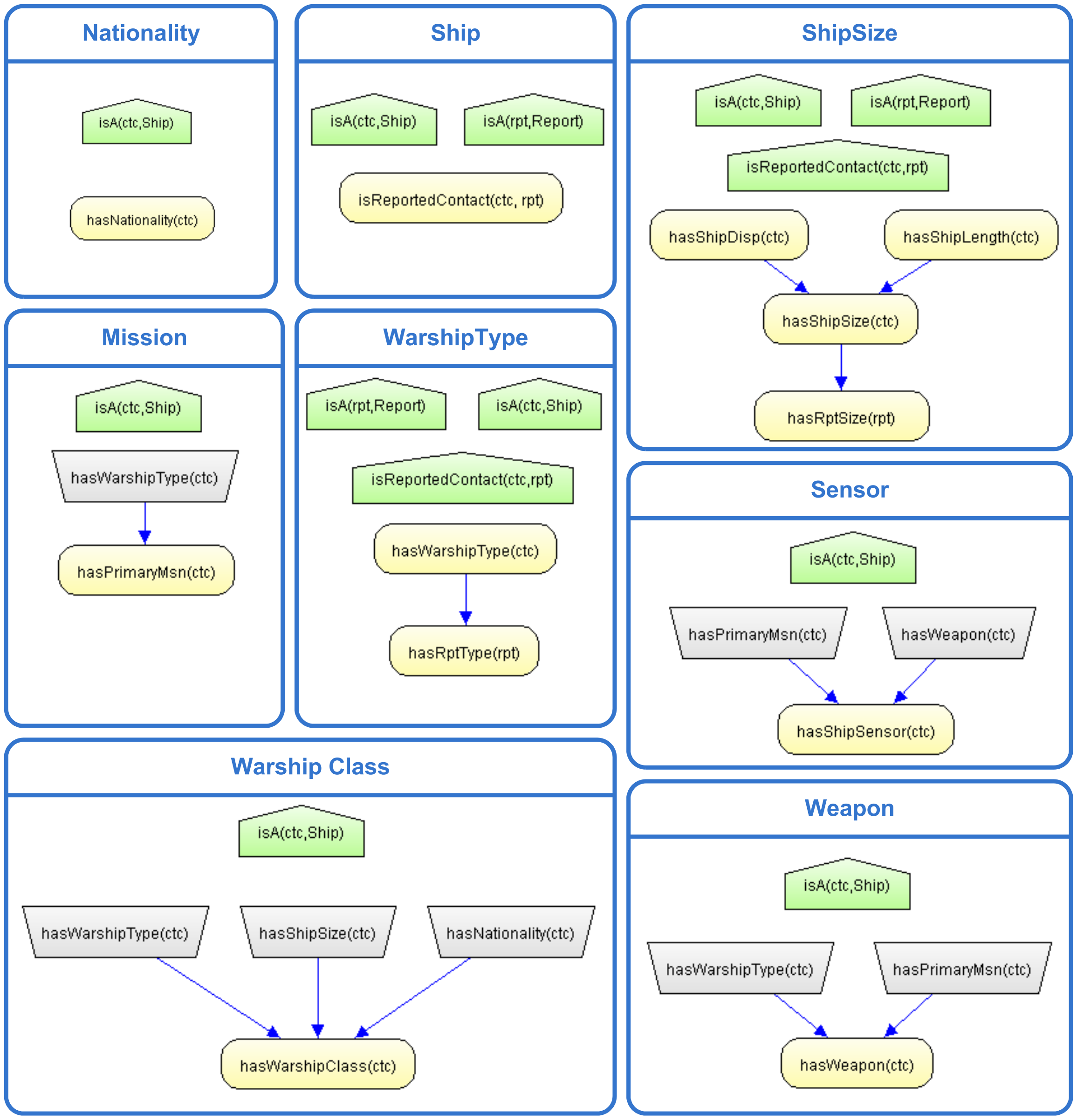
After each iteration of model development is complete, query the model to produce an SSBN for the Prime Query using the software.

Variable	Evidence
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc4, rt1	isReportedContact(ctc4(Ship), rt1(Report))=true
ctc5	hasShipSensor(ctc5(Ship))=RAS_SPY1D

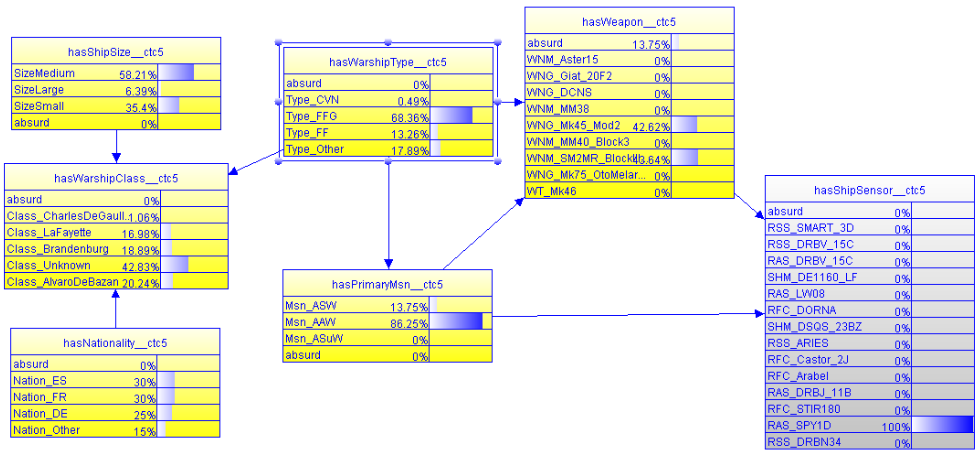


A.4.3 Completed Military Ship Probabilistic Ontology

Three iterations of the model are conducted to complete the first spiral of the Military Ship PO as discussed above. The completed MTheory model is shown.



Variable	Evidence
ctc1	none
ctc2	hasNationality(ctc2(Ship)) = Nation_DE
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc3, rs1	isReportedContact(ctc3(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc4, rt1	isReportedContact(ctc4(Ship), rt1(Report))=true
ctc5	hasShipSensor(ctc5(Ship))=RAS_SPY1D



A.5 Evaluation Activity

The Evaluation Activity completes the PODM by developing a series of test cases to assess the system in realistic scenarios.

A.5 Conduct Elicitation Review

An elicitation review is a holistic review of the probabilistic ontology to ensure it is consistent with the spiral objective and Top-level Objective Statement. Laskey and Mahoney describe the elicitation review as an overall review of node definitions, state definitions, independence assumptions, and probability distributions.

A5.2 Draft Test Cases

A series of increasingly complex test cases is developed to test model logic and coherence in an operational context.

A.5.2.1 Test Case 1

The baseline evidence provides little information to the Commander. A summary paragraph for Test Case 1 reads:

*There is a surface ship contact of interest (*ctc1*) that is reported to be a warship of medium size.*

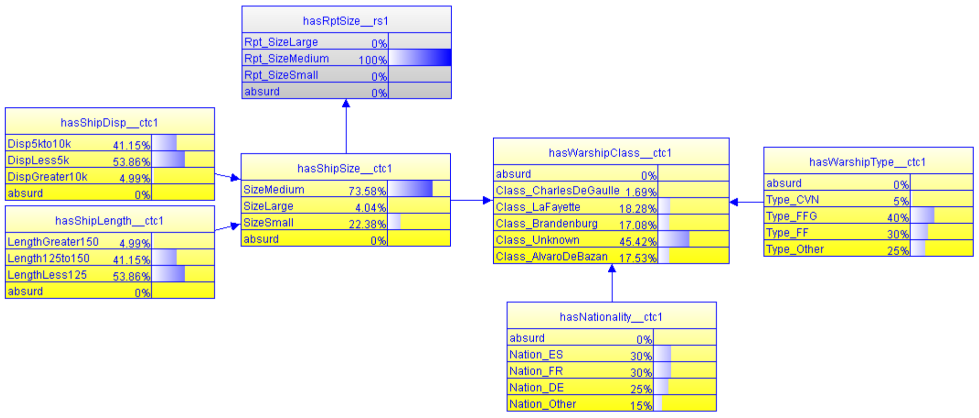
A.5.2.1.1 Populate Evidence Variables

The appropriate evidence is incorporated into the PO model using FOL statements.

Variable	Evidence
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true

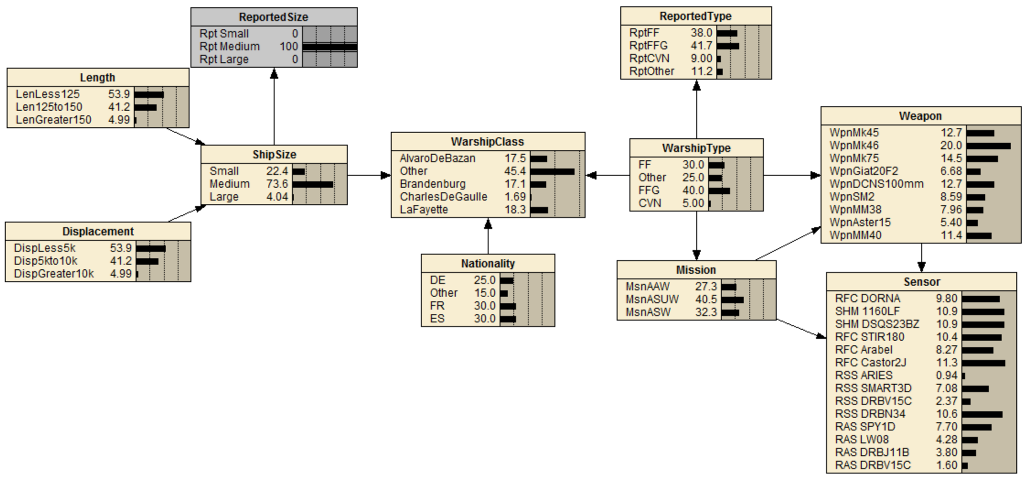
A.5.2.1.2 Run PO Model and Evaluate Results

Once all of the evidence for the case is loaded into the PO, the Prime Query is executed and PO results are evaluated by expert reviewers to identify potential logical or relationship errors.



A.5.2.1.3 Correct Model as Required

The PO should produce the anticipated results shown in the BN test model.



A.5.2.2 Test Case 2

Test Case 2 continues the scenario with the addition of a report that the type of ship is believed to be a guided-missile frigate (FFG). Test Case 2 can be summarized as:

*The warship contact of interest (*ctc1*) is reported to be a medium-sized guided-missile frigate (FFG).*

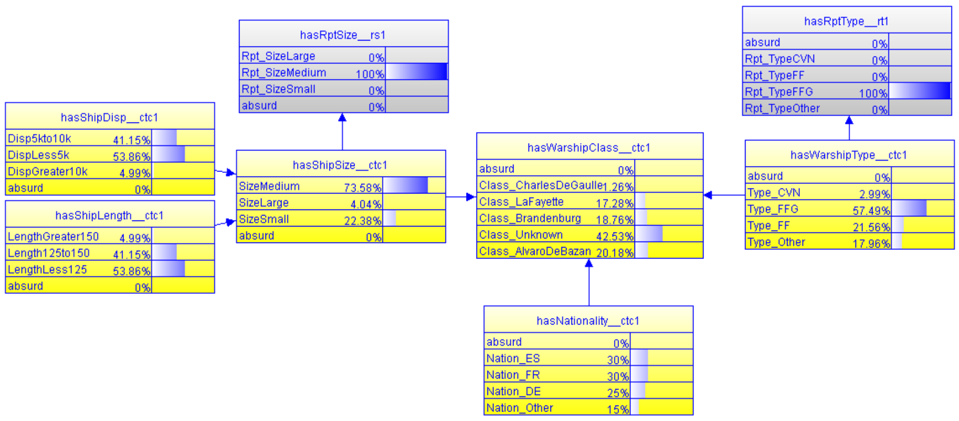
A.5.2.2.1 Populate Evidence Variables

The appropriate evidence is incorporated into the PO model using FOL statements.

Variable	Evidence
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true

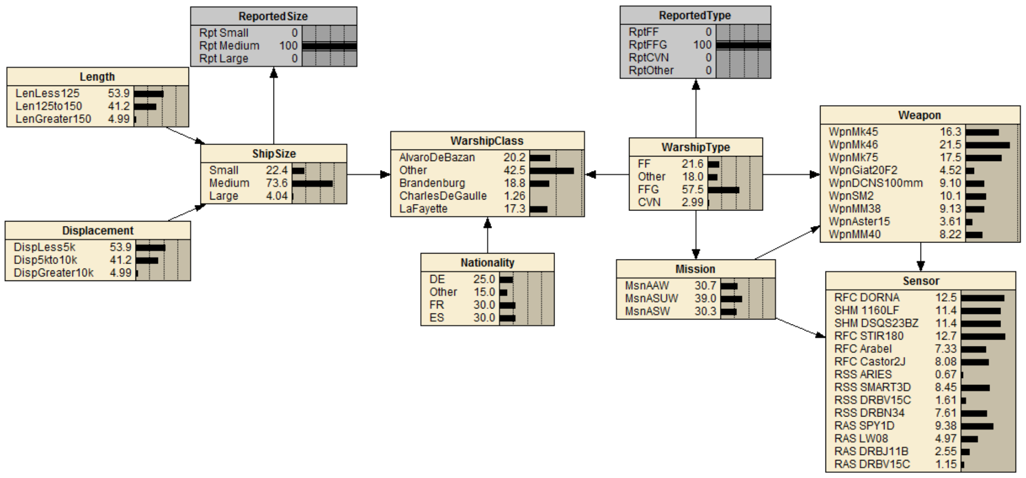
A.5.2.2.2 Run PO Model and Evaluate Results

Once all of the evidence for the case is loaded into the PO, the Prime Query is executed and PO results are evaluated by expert reviewers to identify potential logical or relationship errors.



A.5.2.2.3 Correct Model as Required

The PO should produce the anticipated results shown in the BN test model.



A.5.2.3 Test Case 3

As the scenario continues, electronic sensing equipment has detected the presence of a SPY-1D air search radar emitting from the contact of interest. The summary paragraph for this case reads:

The warship contact of interest (ctc1), reported to be a medium-sized guided-missile frigate, is radiating a SPY-1D air search radar.

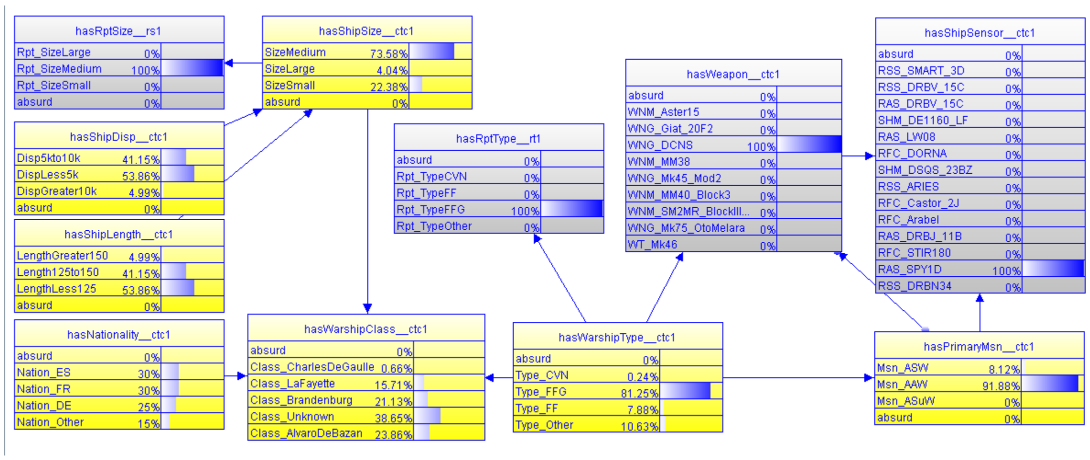
A.5.2.3.1 Populate Evidence Variables

The appropriate evidence is incorporated into the PO model using FOL statements.

Variable	Evidence
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true
ctc1	hasSensor(ctc1(Ship))=RAS_SPY1D

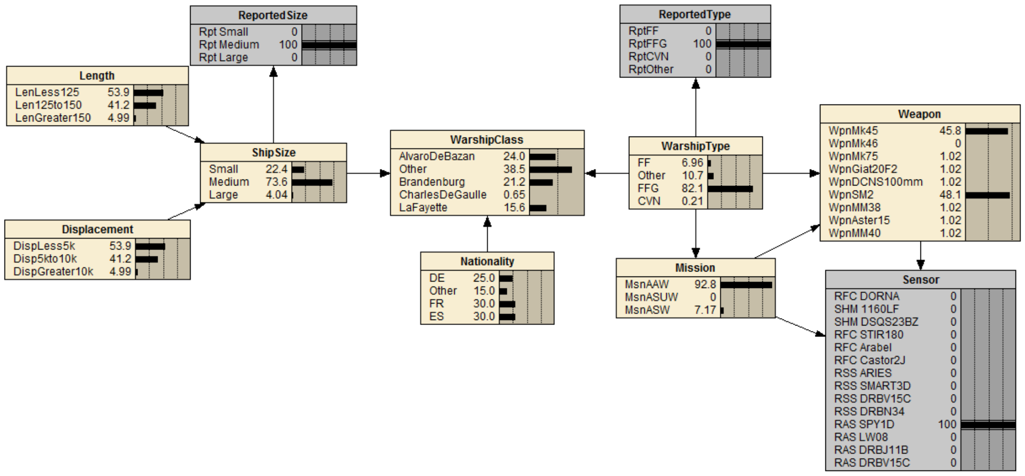
A.5.2.3.2 Run PO Model and Evaluate Results

Once all of the evidence for the case is loaded into the PO, the Prime Query is executed and PO results are evaluated by expert reviewers to identify potential logical or relationship errors.



A.5.2.3.3 Correct Model as Required

The PO should produce the anticipated results shown in the BN test model.



A.5.2.4 Test Case 4

Finally, a helicopter reports a German flag flying on board the COI. The summary for the scenario now reads:

The warship contact of interest (ctc1), reported to be a medium-sized German guided-missile frigate, is radiating a SPY-1D air search radar.

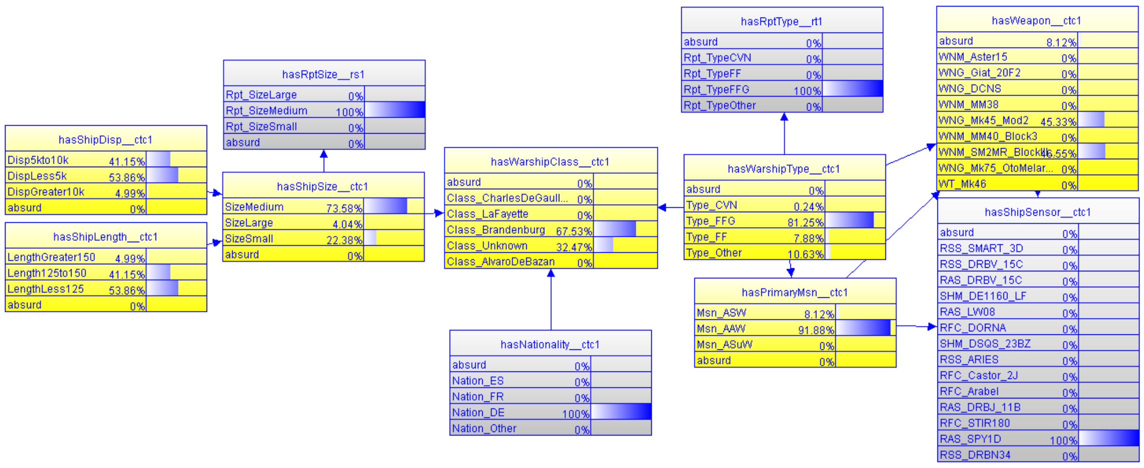
A.5.2.4.1 Populate Evidence Variables

The appropriate evidence is incorporated into the PO model using FOL statements.

Variable	Evidence
rs1	hasRptSize(rs1(Report))=Rpt_SizeMedium
ctc1, rs1	isReportedContact(ctc1(Ship), rs1(Report))=true
rt1	hasReportedType(rt1(Report))=Rpt_TypeFFG
ctc1, rt1	isReportedContact(ctc1(Ship), rt1(Report))=true
ctc1	hasSensor(ctc1(Ship))=RAS_SPY1D
ctc1	hasNationality(ctc1((Ship))=Nation_DE

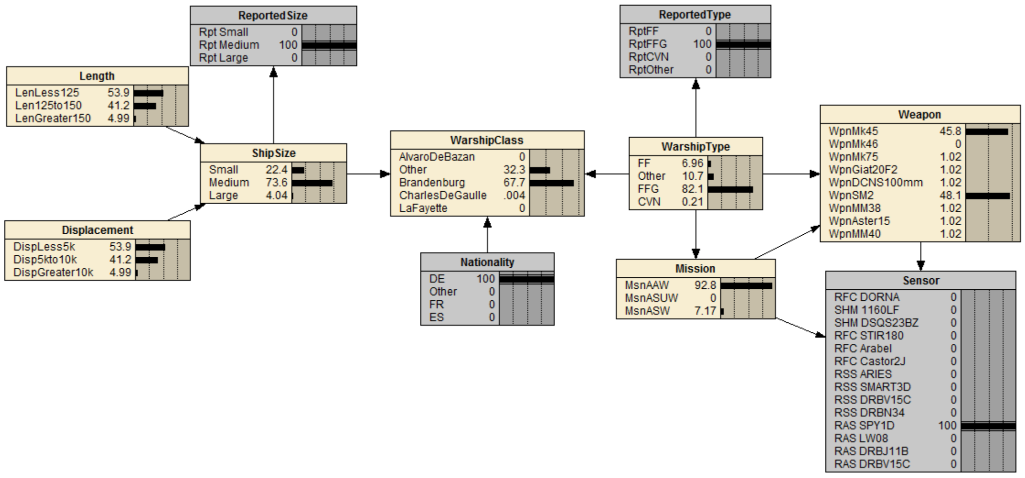
A.5.2.4.2 Run PO Model and Evaluate Results

Once all of the evidence for the case is loaded into the PO, the Prime Query is executed and PO results are evaluated by expert reviewers to identify potential logical or relationship errors.



A.5.2.4.3 Correct Model as Required

The PO should produce the anticipated results shown in the BN test model.



APPENDIX B: TUTORIALS

B.1 Tutorials

Two tutorials are offered to familiarize the reader with useful software tools used for probabilistic ontology development, Protégé and UnBBayes.

Protégé [154] is an ontology development environment that is used for ontology development and organization. The first tutorial initiates the user to the Protégé 4.1 ontology development tool. The tutorial demonstrates use of the software to create a useable ontology to support diagnosis of a patient entering a clinic with a fever. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus.

UnBBayes [159] is a probabilistic ontology development tool used to develop models in the MEBN language. The second tutorial demonstrates use of UnBBayes to create a useable probabilistic ontology to support diagnosis of a patient entering a clinic with a fever. The background ontology is created in the Protégé 4.1 tutorial.

B.2 Tutorial: Patient Diagnosis Ontology Tutorial

This tutorial initiates the user to the Protégé 4.1 ontology development tool. The tutorial will demonstrate use of the software to create a useable ontology to support diagnosis of a patient entering a clinic with a fever. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus.

B.2.1 Create a New Ontology in Protégé

An ontology defines common language to share information about a given domain, allowing a common understanding of the structure and relationships about the information contained within.

a. Start Protégé.

b. In the Welcome to Protégé dialogue, select Create new OWL ontology.

c. In the Create ontology wizard dialogue, enter http://www.semanticweb.org/patientdiagnosis.owl and click Continue.

d. Navigate to the desired physical location of the file and type PatientDiagnosis.owl, then Save and Continue.

e. From the dropdown menu, select OWL/XML and click Finish to save your ontology and open the workspace.

B.2.2 Create Named Classes

Classes are the hierarchically arranged objects of independent existence about which the knowledge is stored in the ontology.

a. Select the Classes tab.

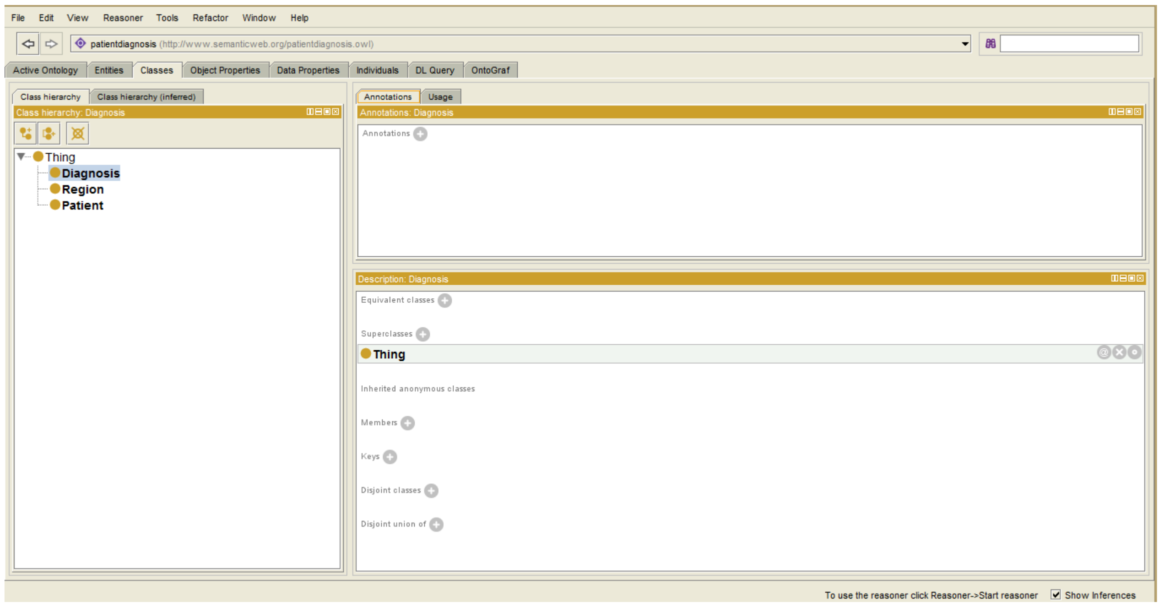
b. Highlight Thing in the Class hierarchy: Thing pane and then select the Add subclass button.

   This creates a new class as a subclass of the selected class (Thing).

c. Enter Patient in the dialogue and then OK.

d. With Patient highlighted, select the Add sibling class button and add the Region, and Diagnosis classes.

e. Save the ontology using File/Save from the Menu bar and then OK.


File Edit View Reasoner Tools Refactor Window Help
patientdiagnosis (http://www.semanticweb.org/patientdiagnosis.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals DL Query OntoGraf
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Diagnosis
Thing
Diagnosis
Region
Patient
Annotations Usage
Annotations: Diagnosis
Annotations
Description: Diagnosis
Equivalent classes
Superclasses
Thing
Inherited anonymous classes
Members
Keys
Disjoint classes
Disjoint union of
To use the reasoner click Reasoner->Start reasoner
Show Inferences

B.2.3 Create Object Properties

Object properties are the attributes common to all members of the class, including their relationships with other classes.

a. Select the Object Properties tab.

b. Highlight topObjectProperty and select the Add sub property button.

c. Enter hasDiagnosis and then OK.

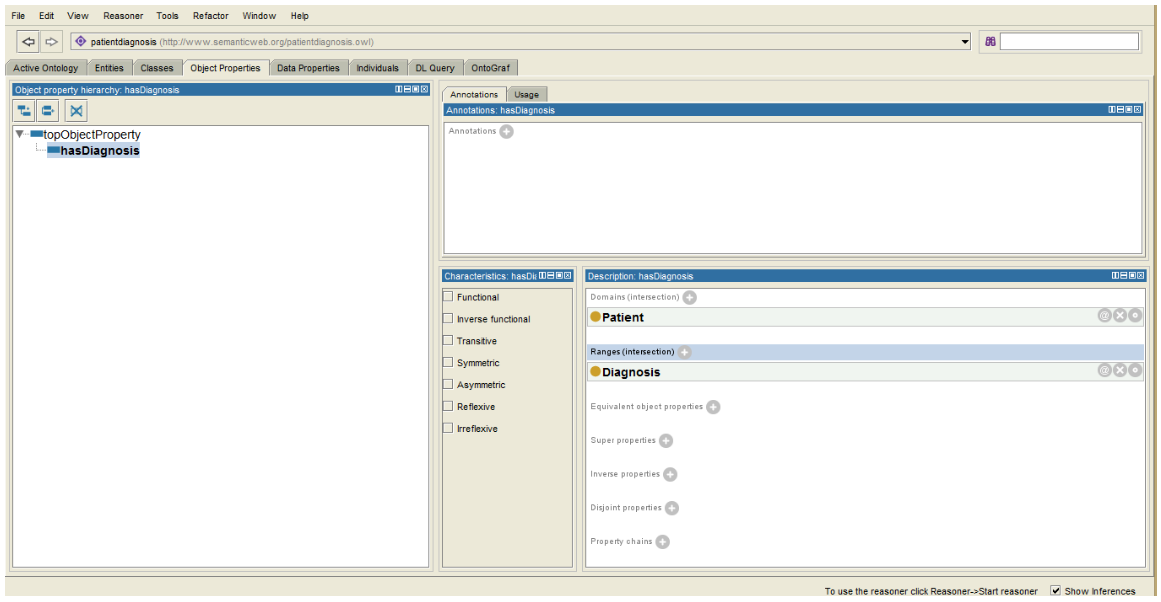
d. With hasDiagnosis highlighted, select the Add icon (+) next to Domains (intersection) on the Description: hasDiagnosis view.

e. In the hasDiagnosis dialogue, select the Class hierarchy tab, expand the Thing class, highlight Patient, then select OK.

f. Select the Add icon (+) next to Ranges (intersection) on the Description: hasDiagnosis view.

g. In the hasDiagnosis dialogue, select the Class hierarchy tab, expand the Thing class, highlight Diagnosis, then select OK.

h. Save the ontology as before.



B.2.4 Create Data Properties

Data properties are enumerated values associated with features of a class, and include detailed lists, numbers, or Boolean information.

a. Select the Data Properties tab.

b. Highlight topDataProperty and select the Add sub property button.

c. Enter hasFluEpidemicPresent and then OK.

d. With hasFluEpidemicPresent still highlighted, select the Add sibling property button and add hasVisitedRegion, and then OK.

e. Highlight hasFluEpidemicPresent and select the Add icon (+) next to Domains (intersection) on the Description: hasFluEpidemicPresent pane.

f. In the hasFluEpidemicPresent dialogue, select the Class hierarchy tab, expand the Thing class, highlight Region, then select OK.

g. Select the Add icon (+) next to Ranges (intersection) on the Description: hasFluEpidemicPresent pane.

h. Select the Built in datatypes tab, highlight Boolean, and click OK.

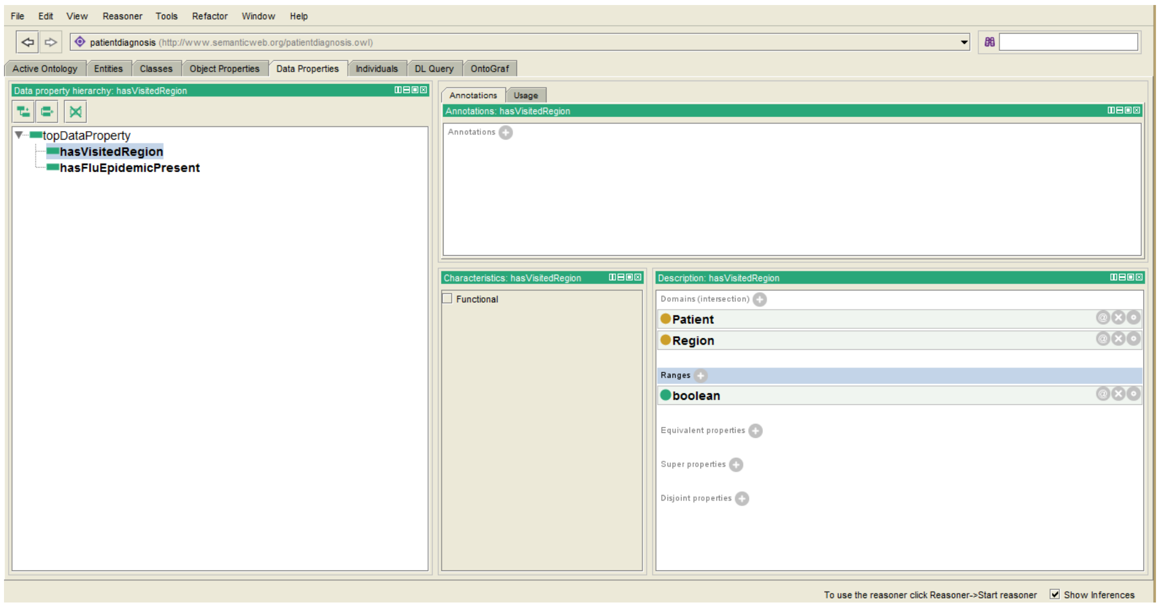
i. Highlight the hasVisitedRegion object property in the Data property hierarchy pane, and then select the Add icon (+) next to Domains (intersection) on the Description: hasVisitedRegion view.

j. In the hasVisitedRegion dialogue, select the Class hierarchy tab, expand the Thing class, highlight Patient, then select OK. Again select the Add icon (+) next to Domains (intersection), highlight Region, then select OK.

k. Select the Add icon (+) next to Ranges (intersection) on the Description: hasVisitedRegion pane.

l. Select the Built in datatypes tab, highlight Boolean, and click OK.

m. Save the ontology.


File
Edit
View
Reasoner
Tools
Refactor
Window
Help
patientdiagnosis (http://www.semanticweb.org/patientdiagnosis.owl)
Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
DL Query
OntoGraf
Data property hierarchy: hasVisitedRegion
Annotations
Usage
Annotations: hasVisitedRegion
Annotations
Characteristics: hasVisitedRegion
Functional
Description: hasVisitedRegion
To use the reasoner click Reasoner->Start reasoner
Show Inferences

topDataProperty
- hasVisitedRegion
- hasFluEpidemicPresent

Domains (intersection)
- Patient
- Region

Ranges
- boolean

Equivalent properties

Super properties

Disjoint properties

B.2.5 Create Instances

Instances represent the evidence available to the ontology in the form of facts of knowledge held to be true.

a. Select the Classes tab.

b. Highlight Diagnosis in the Class hierarchy pane and then click the Add (+) next to Members in the Description: Diagnosis pane.

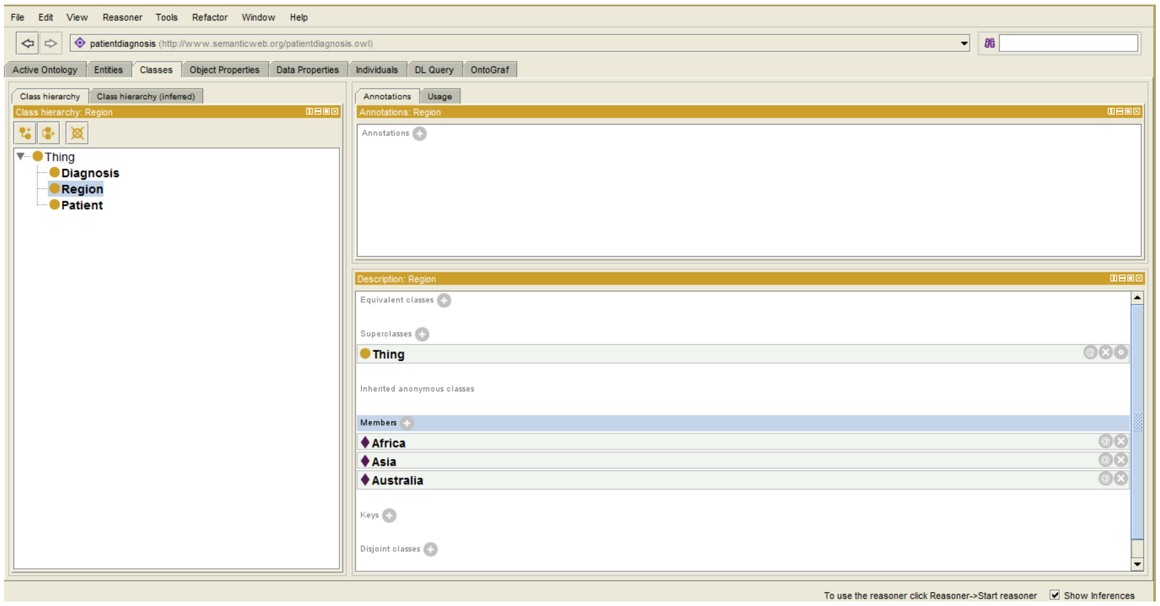
c. Click the Add individual button, type FluEpidemic, then OK.

d. While still in the Diagnosis pane, add the individual OtherVirus in a similar manner.

e. Now highlight Patient in the Class hierarchy pane and add Patient1, Patient2, Patient3, and Patient4.

f. Finally, highlight Region, and add Asia, Australia, and Africa.

g. Save the ontology.



An in-depth tutorial is offered by the University of Manchester at http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

Congratulations!

You have completed the Patient Diagnosis Ontology Tutorial!

B.3 Patient Diagnosis Probabilistic Ontology Tutorial

This tutorial will demonstrate use of UnBBayes to create a useable probabilistic ontology to support diagnosis of a patient entering a clinic with a fever. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus. The background ontology was previously created within Protégé 4.1 and saved as PatientDiagnosis.owl.

B.3.1 Open UnBBayes

a. Double-click on unbbayes-4.11.4.jar.

B.3.2 Create new MTheory from OWL file

A MEBN Theory (MTheory) allows the introduction of uncertainty into an ontology by implicitly expressing a joint probability distribution over groups of hypotheses that are globally consistent within the domain.

a. Click Open net button on toolbar just below main menu bar.

b. Navigate to the PatientDiagnosis.owl file and click Open.

c. In the I/O extension conflict dialogue, select UnbBayes file with PR-OWL 2.0 from the dropdown menu and OK.

d. When the workspace opens, you can check the ontology on the OWL2Entities tab. It should be identical to the ontology created in Protégé during the previous tutorial.

B.3.3 Make three new MFrags

Knowledge in MTheories is captured via MEBN Fragments (MFrags), each of which represents probability information for a group of related random variables.

a. Select the MTheory tab.
b. Click Insert MFrag button at left of MFrag editing pane.
c. In the MFrag name field, highlight DMFrag1 and type Region_MF. Press Enter.
d. Click Insert MFrag button at left of MFrag editing pane.
e. In the MFrag name field, highlight DMFrag2 and type Patient_MF. Press Enter.
f. Click Insert MFrag button at left of MFrag editing pane.
g. In the MFrag name field, highlight DMFrag3 and type RegionVisit_MF. Press Enter.
h. You now have three blank MFrags as shown below.

   ***NOTE**: You can resize windows by dragging the window borders as shown below.*

B.3.4 Rename the MTheory

a. Click Edit the MTheory button at left of MFrag editing pane as shown below.
b. Highlight the text MEBN in the MTheory name field and change to PatientDiagnosis_MT. Press Enter.

B.3.5 Save the MTheory

a. Click on Save net button on the toolbar just below main menu bar.

b. Type PatientDiagnosis_a.ubf in the file name and press the Save button.

   ***NOTE**: Be sure to include the .ubf extension. The file will not save if you have a missing or incorrect file extension.*

c. In the drop-down menu, select UnBBayes File with PR-OWL 2.0 and OK.

d. Observe the File Saved dialogue and click OK.

e. Close and re-open the file, ensuring selection of UnBBayes File with PR-OWL 2.0 from the drop-down menu when prompted.

   ***NOTE**: It is advisable to keep partial models at you construct your PO. Edit the filename to indicate the version number (e.g.* PatientDiagnosis_a.ubf*).*

B.3.6 Define the Region MFrag

The Region MFrag captures knowledge about the presence of a flu epidemic in a specific region.

a. Click Edit the MTheory button.

b. Double-click Region_MF in the MTheory tree. Ensure Region_MF appears above the editing pane.

c. Click the Insert Ordinary Variable button (blue box) at left of editing pane.

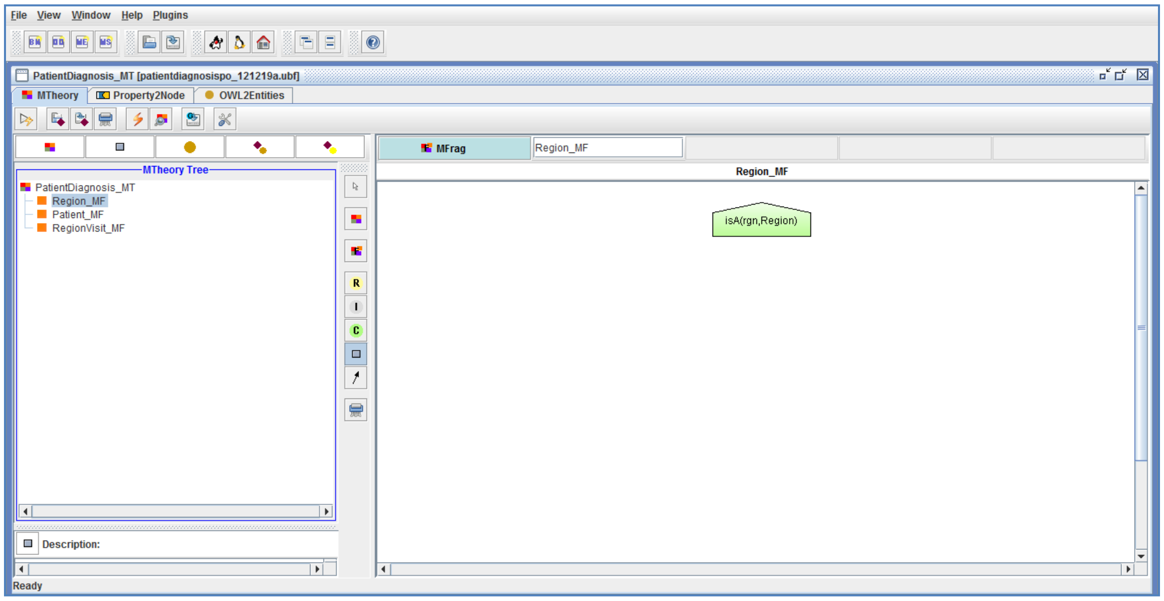
d. Move cursor to top of the MFrag workspace and click to drop an IsA node into the MFrag.

e. Select Region from the drop-down list of types.

f. Change the name OX1 to rgn in the ordinary variable name field. Press Enter.

NOTE: *Lowercase text is used for variable definition.*

g. Right-click the IsA node and select Resize to fit text.



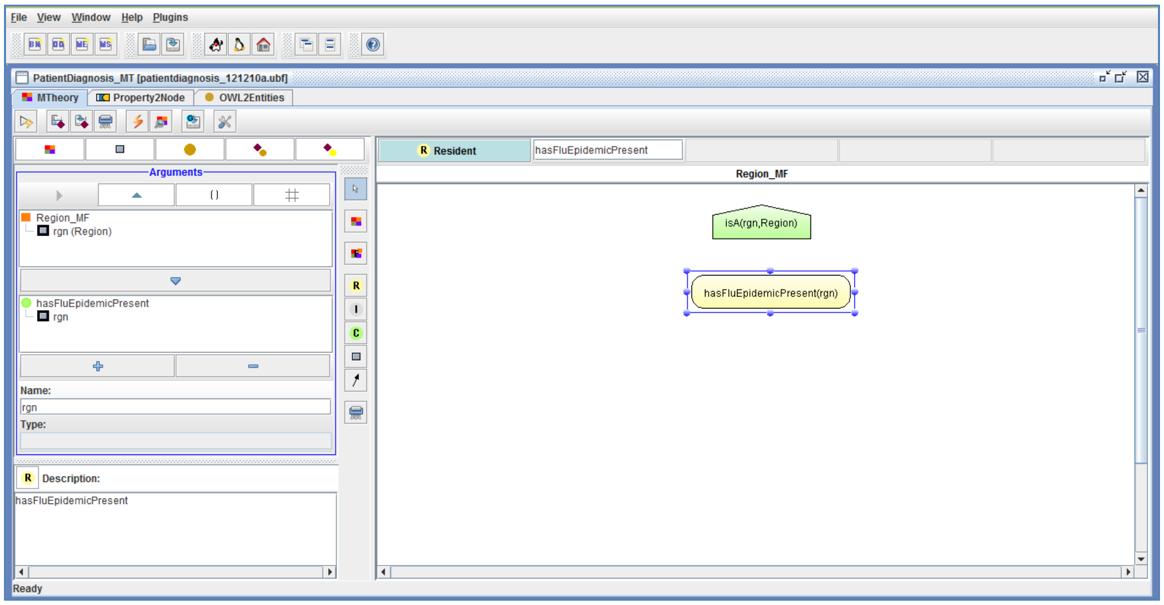
h. Click the Property2Node tab and then the Show OWL Properties button.

i. Highlight hasFluEpidemicPresent and drag the data property variable onto the workspace.

j. Click the MTheory tab, click the Edit the MTheory button, and double click hasFluEpidemicPresent in the Resident Node pane.

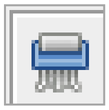
k. In the Resident Node pane, click () to select the ordinary variable for this node.

NOTE: It is possible to create Resident nodes without the drag-and-drop feature, but these will not be linked to the ontology for future saves.

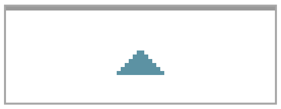


l. Double-click rgn(Region) to insert the ordinary variable. Resize the resident node. The MFrag should look like the figure above.

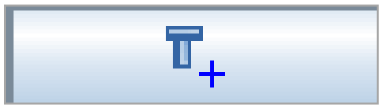
NOTE: The Shredder button can be used to remove unwanted nodes.



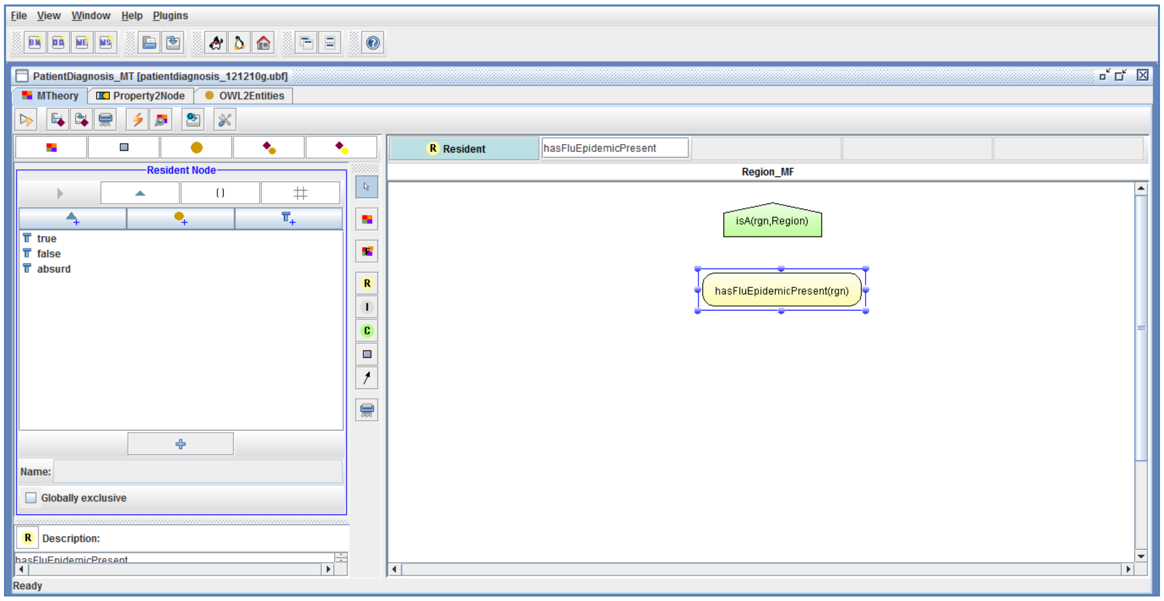
m. Click on the Edit States button in the top-left pane (blue triangle).



n. Click Insert boolean states and then the + button.



o. Save the MTheory with a new version number.



B.3.7 Define the RegionVisit MFrag

The RegionVisit MFrag establishes whether a particular patient has visited a specific region.

a. Click Edit the MTheory button.

b. Double-click RegionVisit_MF in the MTheory tree. Observe RegionVisit_MF above the editing pane.

c. Click the Insert Ordinary Variable button on left side of editing pane.

d. Move cursor to top of the MFrag workspace and click to drop an IsA node into the MFrag.

e. Select Patient from the drop-down list of types.

f. Change the name OX1 to p in the ordinary variable field name

g. Again click the Insert Ordinary Variable button and drop an IsA node onto the MFrag.

h. Select Region from the drop-down list and name it rgn, as before.

i. Click the Property2Node tab and then the Show OWL Properties button.

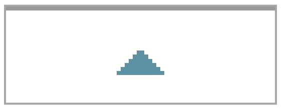
j. Highlight hasVisitedRegion and drag the object property variable onto the workspace.

k. Click the MTheory tab, click the Edit the MTheory button, and double click hasVisitedRegion in the Resident Node pane.

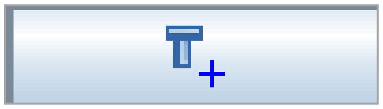
l. In the Resident Node pane, click () to select the ordinary variables for this node.

m. Double-click on p(Patient) and rgn(Region) to insert the ordinary variables.

n. Click on the Edit States button in the Arguments pane.

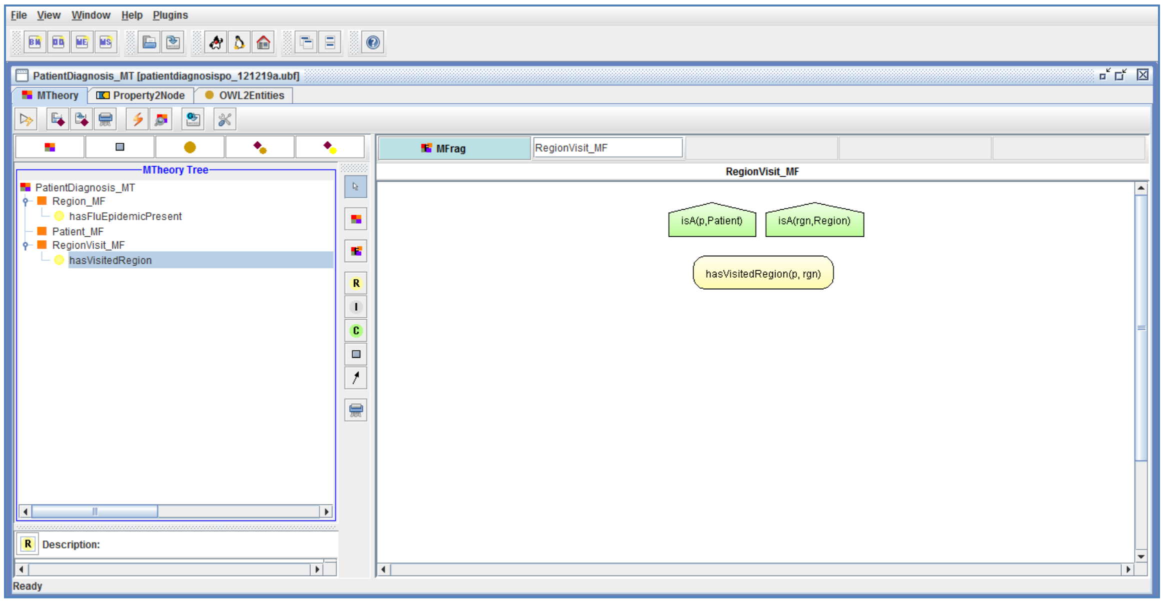


o. Click Insert boolean states button and then the plus (+) to indicate that the hasVisitedRegion(p,rgn) random variable has boolean states.



p. Resize all three nodes in the MFrag.

q. Save the MEBN.



B.3.8 Define the Patient MFrag

The Patient MFrag establishes the relationship between the patient and a region which may or may not have a flu epidemic present to determine a diagnosis.

a. Click Edit the MTheory button and double-click Patient_MF in the MTheory tree.

b. Add two IsA nodes: IsA(p, Patient) and IsA(rgn, Region) as described above.

c. Click the Property2Node tab and then the Show OWL Properties button.

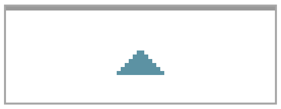
d. Highlight hasDiagnosis and drag the object property variable onto the workspace.

e. Click the MTheory tab, click the Edit the MTheory button, and double click hasDiagnosis in the Resident Node pane.

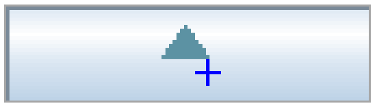
f. In the Resident Node pane, click () to select the ordinary variable for this node.

g. Double-click on p(Patient) to insert the ordinary variable.

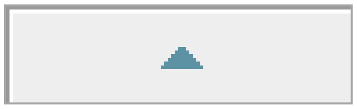
h. Click on the Edit States button



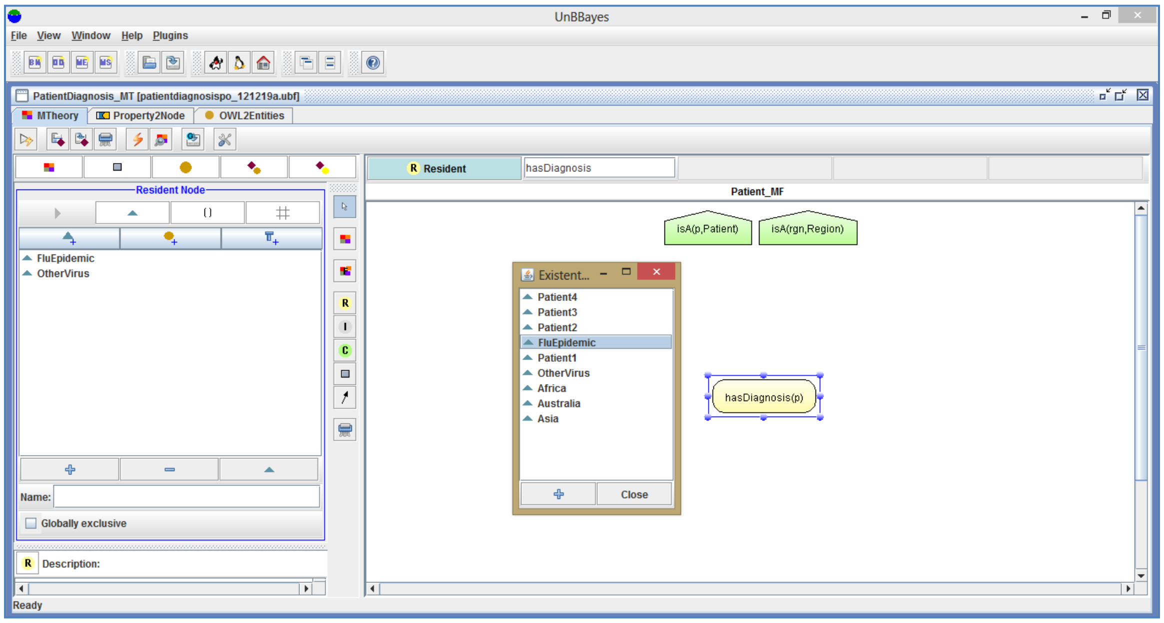
and then the Insert category states button in the Arguments pane.



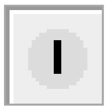
i. Click Add a pre-defined state button (lower blue triangle).



j. Click FluEpidemic, then plus. Click OtherVirus, then plus. Close the dialogue.



k. Click Insert Input Node at left of the editing pane. Click on the editing pane to drop a new blank input node into the MFrag.



l. Double-click hasFluEpidemicPresent in the Resident List to point the new input node to the hasFluEpidemicPresent random variable in the Region MFrag.

m. Click the hasFluEpidemicPresent input node and select rgn from the drop-down list of arguments. Resize the hasFluEpidemicPresent input node.

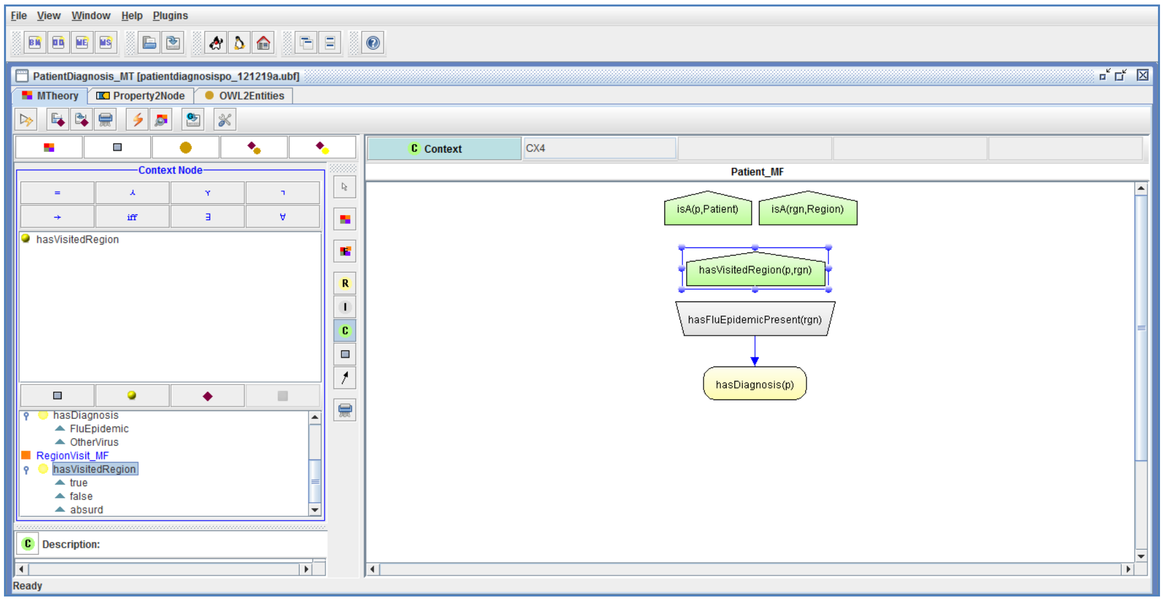
n. Select the arc tool (arrow) from the left of the MFrag pane and draw an arc from hasFluEpidemicPresent to hasDiagnosis.

o. To add the reference context node, click Insert context node to the left of the editing pane and insert a blank context node onto the editing pane. This reference node will represent the knowledge that there is a link between the hasVisitedRegion of a patient and hasFluEpidemicPresent of a region when the patient has visited the region.

p. Click formula in the Context Node pane at the upper left of the window.

q. Double-click the hasVisitedRegion in the tree below the RegionVisit_MF.

r. Double-click hasVisitedRegion in the Context Node pane, then select p from the Patient_label drop-down and rgn from the Region_label drop-down.



s. Resize the nodes.

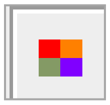
t. Save the MTheory with a new version number.

NOTE: The structure of the PO is complete. The next steps involve populating the probabilities to represent uncertainty.

B.3.9 Define the local distribution for the hasFluEpidemicPresent random variable

This local distribution establishes the prior probability that a region has a flu epidemic present at any moment in time, absent any additional evidence.

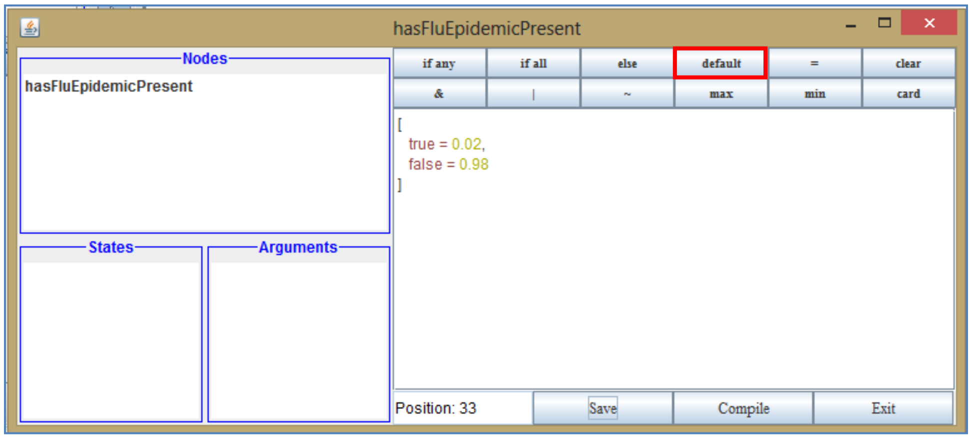
a. Click Edit the MTheory button and double-click the Region_MF MFrag in the MTheory Tree.



b. Select the hasFluEpidemicPresent and click the distribution button # in the Resident Node area. A pop-up box will appear for the distribution.

c. Click the Default button. A formula will appear in the blank node window.

d. Change the word formula after each of the states to 0.02 for True and 0.98 for False as shown below. Delete the absurd state and the comma after False.



```
[
  true = 0.02,
  false = 0.98
]
```

NOTE: *Take care to not erase the comma after the probability in the first state.*

e. Click Compile. Observe a message that the table compiled successfully. Click Save and then Exit.

NOTE: *If you do not save each distribution separately, the input will be lost.*

B.3.10 Define the local distribution for the hasDiagnosis random variable

This local distribution establishes the posterior, conditional relationship that a patient has the flu given the presence (or absence) of the flu in a region that he has visited.

a. Click Edit the MTheory button and double-click the Patient_MF MFrag in the MTheory tree.

b. Select the hasDiagnosis node and click the distribution button # in the Resident Node area. A pop-up box will appear for the distribution.

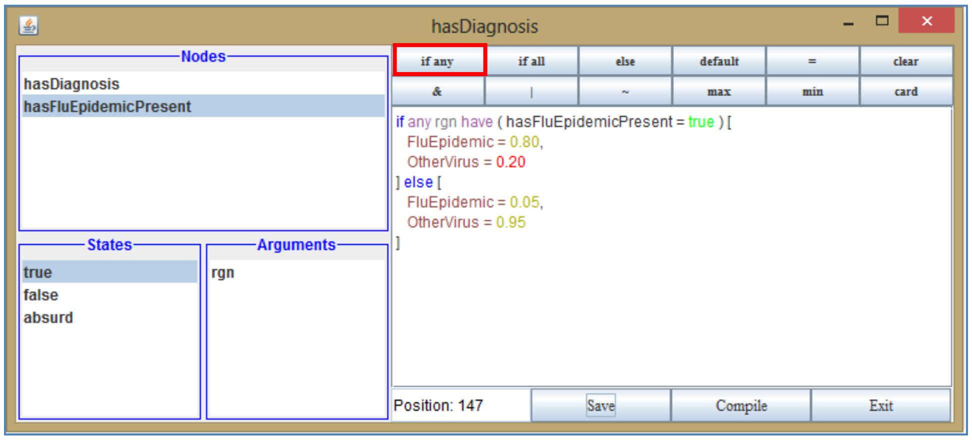
c. Click the if any button. A formula will appear in the blank node window.

d. Change the text paramSubSet to rgn.

e. Change the text booleanFunction to hasFluEpidemicPresent=true.

f. Change the word formula after each of the states to 0.80 for FluEpidemic and 0.20 for OtherVirus. Delete the absurd state and comma after FluEpidemic.

g. Place the cursor after the] and click the else button. Change the word formula after each of the states to 0.05 for FluEpidemic and 0.95 for OtherVirus. Delete the absurd state and comma after FluEpidemic.



```
if any rgn have ( hasFluEpidemicPresent = true ) [
  FluEpidemic = 0.80,
  OtherVirus = 0.20
] else [
  FluEpidemic = 0.05,
  OtherVirus = 0.95
]
```

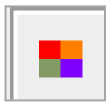
h. Click Compile. Observe the message indicating the table has compiled successfully. Click Save and then click Exit.

i. Save the MTheory with a new version number.

B.3.11 Define the local distribution for the hasVisitedRegion random variable

This local distribution establishes the prior probability that a patient has visited any particular region of interest.

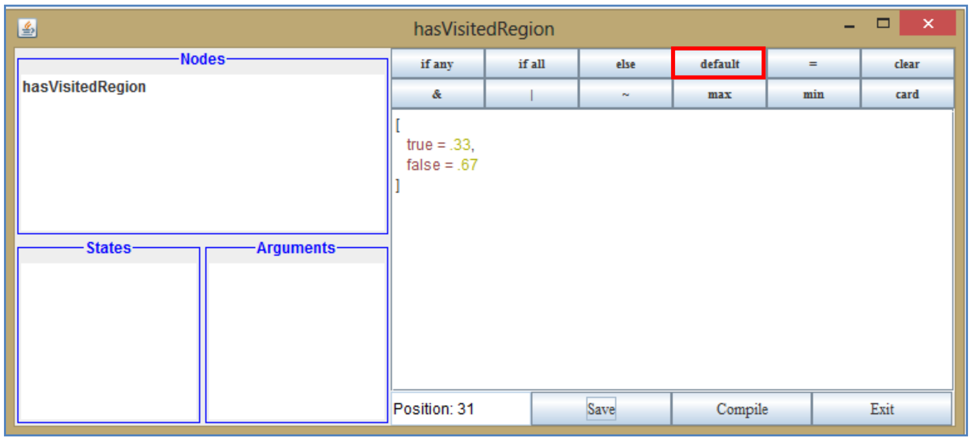
a. Click Edit the MTheory button and double-click the RegionVisit_MF MFrag in the MTheory Tree.



b. Select hasVisitedRegion and click the distribution button # in the Resident Node area. A pop-up box will appear for the distribution.

c. Click the Default button. A formula will appear in the blank node window.

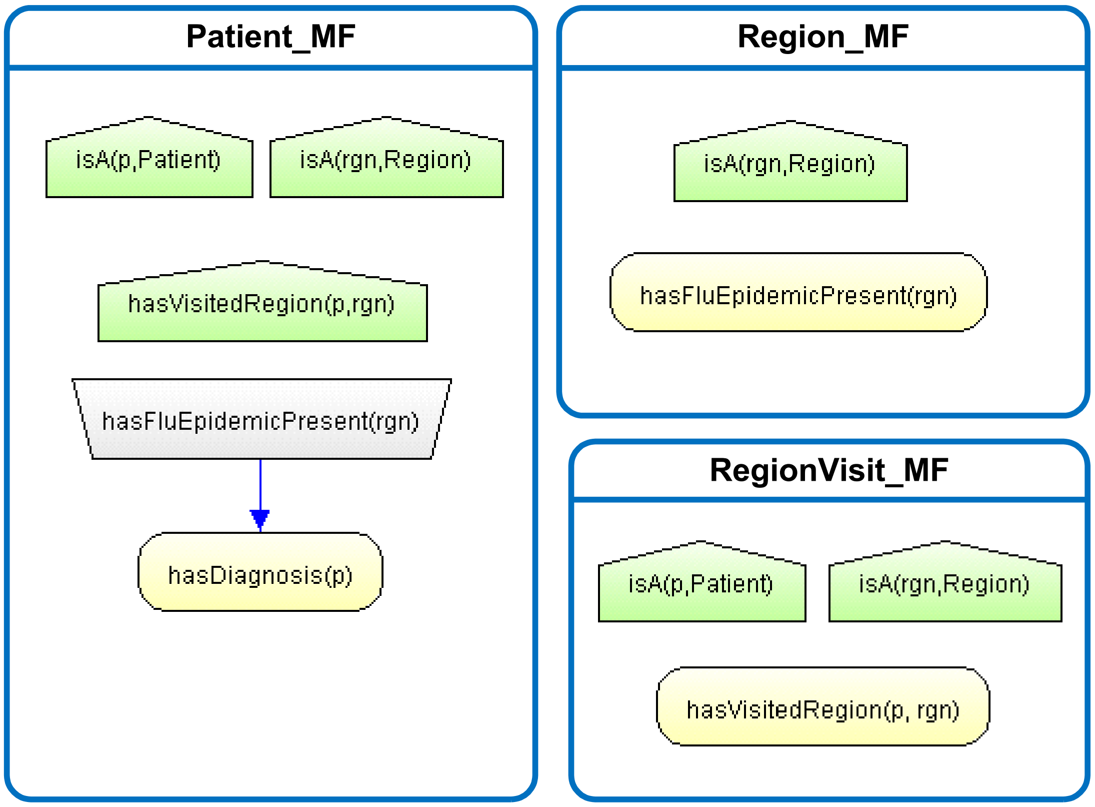
d. Change the word formula after each of the states to 0.33 for True and 0.67 for False as shown below. Delete the absurd state and the comma after False.



```
[
  true = .33,
  false = .67
]
```

e. Click Compile. Observe a message that the table compiled successfully. Click Save and then Exit.

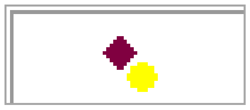
The complete MTheory is shown in the figure below.



B.3.12 Locate Patient 1

This step adds evidence that Patient1 has visited Australia.

a. Click Show findings edition pane.



b. Highlight hasVisitedRegion(0).

c. Click the pencil to bring up the finding: hasVisitedRegion pane.

d. Choose Patient1 from the drop-down menu of arguments, Australia from the second drop-down menu, true from the third drop-down menu, and then click + to add a finding that Patient1 has visited Australia.

e. Save the evidence. Select the Save Knowledge Base button and enter the filename PatientDiagnosis1.plm.

   NOTE*: UnBBayes keeps findings in a separate file from the MTheory and entity instances.*

B.3.13 Run a query

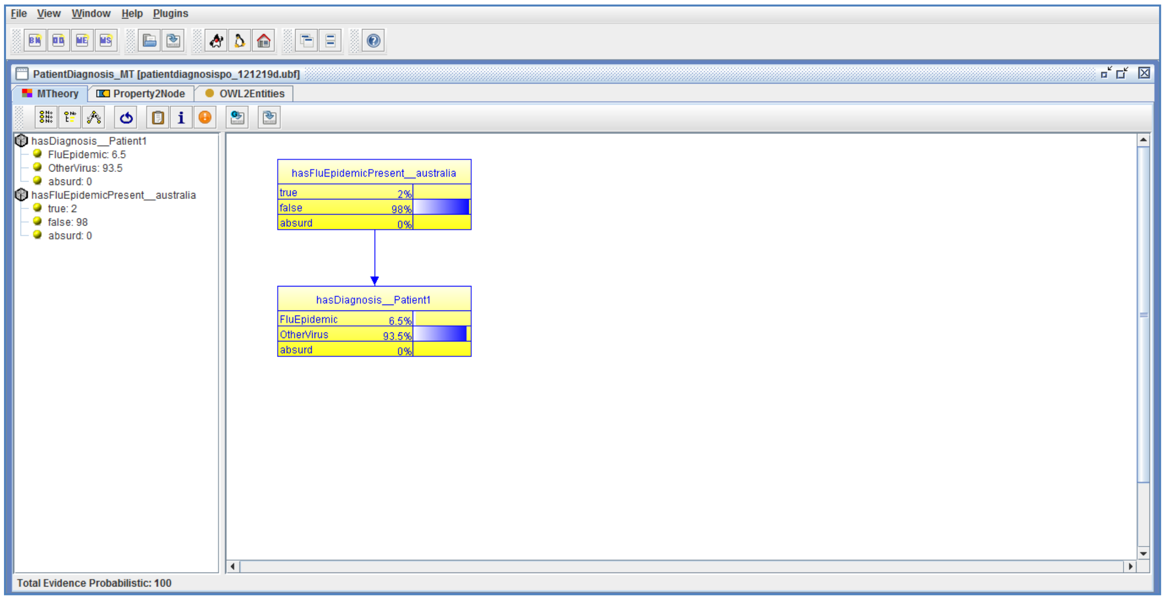
a. Click the Execute Query button.

b. In the pop-up window, highlight hasDiagnosis and press the Select button.

c. Select Patient1 from the drop-down menu of entity instances.

d. Click the Execute button. Observe the Bayesian network.

e. Click the Propagate evidences button to show belief bars.


File View Window Help Plugins
PatientDiagnosis_MT [patientdiagnosispo_121219d.ubf]
MTheory
Property2Node
OWL2Entities

hasDiagnosis__Patient1
FluEpidemic: 6.5
OtherVirus: 93.5
absurd: 0
hasFluEpidemicPresent__australia
true: 2
false: 98
absurd: 0

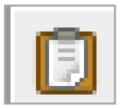
hasFluEpidemicPresent__australia	
true	2%
false	98%
absurd	0%

hasDiagnosis__Patient1	
FluEpidemic	6.5%
OtherVirus	93.5%
absurd	0%

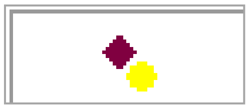
Total Evidence Probabilistic: 100

B.3.14 Add more entities and evidence

a. Click the Return to edit mode button (clipboard).



b. Click Show findings edition pane.



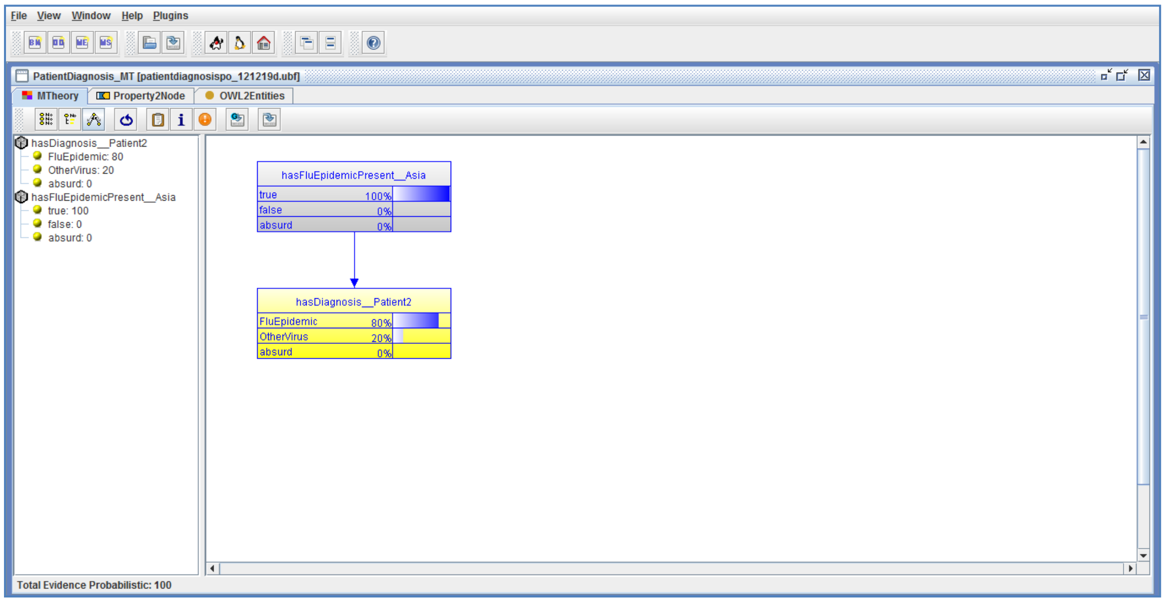
c. Add evidence to the other patients and regions as shown in the table below.

Resident Variable	Variable	Evidence
hasVisitedRegion()	Patient2, Asia	true
hasVisitedRegion()	Patient3, Asia Patient3, Asia	true true
hasVisitedRegion()	Patient4, Asia Patient4, Australia Patient4, Africa	true true true
hasFluEpidemicPresent()	Asia	true
hasFluEpidemicPresent()	Africa	false

d. Save this knowledge base with the name PatientDiagnosis2.plm.

B.3.15 Run additional queries

a. Query PatientDiagnosis(Patient2)


File View Window Help Plugins
PatientDiagnosis_MT [patientdiagnosispo_121219d.ubf]
MTheory
Property2Node
OWL2Entities

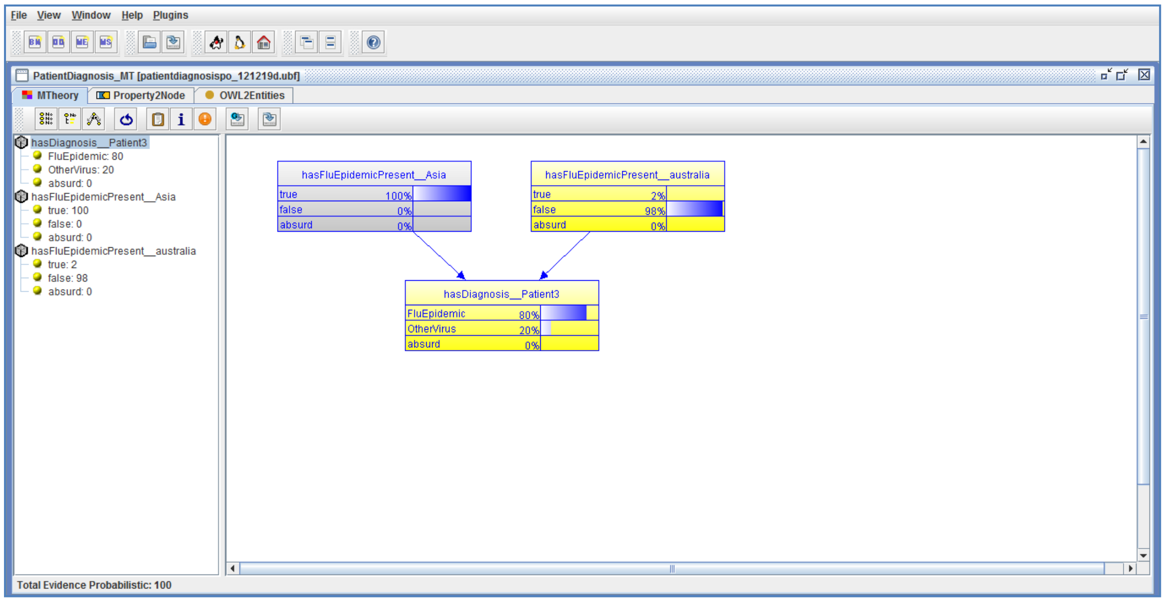
```
hasDiagnosis__Patient2
    FluEpidemic: 80
    OtherVirus: 20
    absurd: 0
hasFluEpidemicPresent__Asia
    true: 100
    false: 0
    absurd: 0
```

hasFluEpidemicPresent__Asia	
true	100%
false	0%
absurd	0%

hasDiagnosis__Patient2	
FluEpidemic	80%
OtherVirus	20%
absurd	0%

Total Evidence Probabilistic: 100

b. Query PatientDiagnosis(Patient3)


File View Window Help Plugins
PatientDiagnosis_MT [patientdiagnosispo_121219d.ubf]
MTheory
Property2Node
OWL2Entities

hasDiagnosis__Patient3
- FluEpidemic: 80
- OtherVirus: 20
- absurd: 0

hasFluEpidemicPresent__Asia
- true: 100
- false: 0
- absurd: 0

hasFluEpidemicPresent__australia
- true: 2
- false: 98
- absurd: 0

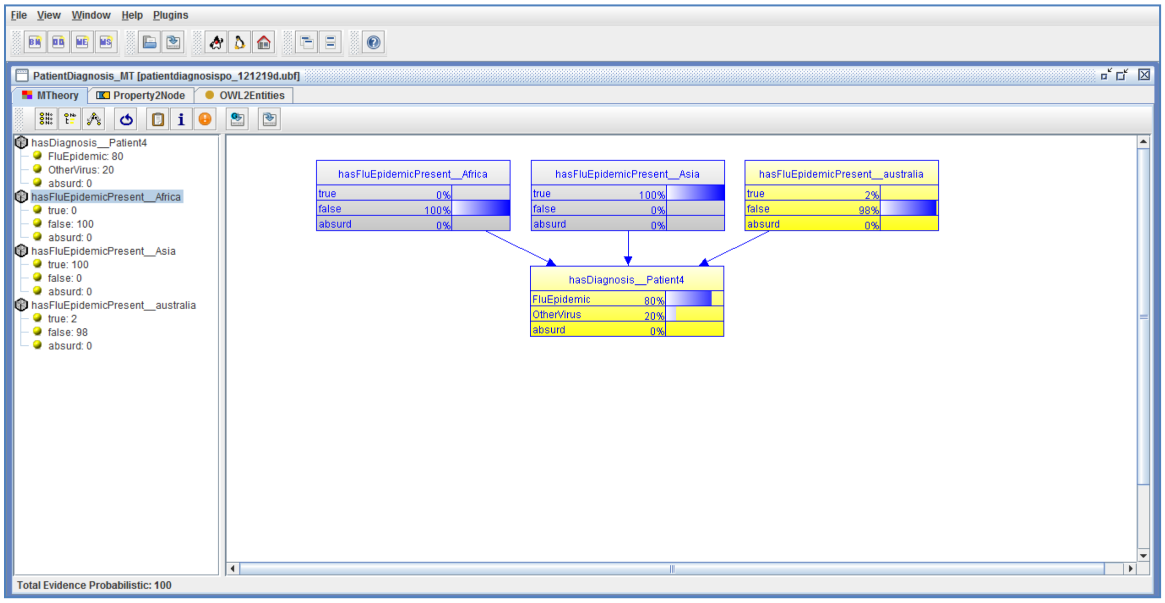
hasFluEpidemicPresent__Asia	
true	100%
false	0%
absurd	0%

hasFluEpidemicPresent__australia	
true	2%
false	98%
absurd	0%

hasDiagnosis__Patient3	
FluEpidemic	80%
OtherVirus	20%
absurd	0%

Total Evidence Probabilistic: 100

c. Query PatientDiagnosis(Patient4)


File View Window Help Plugins
PatientDiagnosis_MT [patientdiagnosispo_121219d.ubf]
MTheory
Property2Node
OWL2Entities

hasDiagnosis__Patient4
- FluEpidemic: 80
- OtherVirus: 20
- absurd: 0

hasFluEpidemicPresent__Africa
- true: 0
- false: 100
- absurd: 0

hasFluEpidemicPresent__Asia
- true: 100
- false: 0
- absurd: 0

hasFluEpidemicPresent__australia
- true: 2
- false: 98
- absurd: 0

hasFluEpidemicPresent__Africa	
true	0%
false	100%
absurd	0%

hasFluEpidemicPresent__Asia	
true	100%
false	0%
absurd	0%

hasFluEpidemicPresent__australia	
true	2%
false	98%
absurd	0%

hasDiagnosis__Patient4	
FluEpidemic	80%
OtherVirus	20%
absurd	0%

Total Evidence Probabilistic: 100

Congratulations!

You have completed the Patient Diagnosis Tutorial.

APPENDIX C: VEHICLE IDENTIFICATION PROBLEM

C.1 Background

Intelligence, surveillance and reconnaissance aircraft employ a suite of sensors to allow the operator to classify detected targets. In this problem, you must develop a decision support system that provides the most likely vehicle type (wheeled or tracked) based on incoming evidence. The model may be used to infer the vehicle type from MTI and imaging sensor reports, weather reports and GIS reports. Vehicles may travel on-road, off-road, or on very-rough terrain. Weather affects imaging sensors and can be generally characterized as clear or cloudy.

C.2 Task

Considering the domain knowledge presented below, develop a probabilistic ontology for military vehicles that will infer vehicle type (wheeled or tracked) from the MTI and imaging sensor reports, weather reports, and GIS reports.

C.3 Research

The following material is provided to assist you in developing and parameterizing your model.

C.3.1 Vehicle

Vehicle speed is restricted by the type of vehicle and the type of terrain traversed. Table C.1 captures the likelihood of a particular vehicle type traveling in a speed regime for each terrain type.

Table C.1 – Vehicle Speed by Terrain

		Stationary	Slow	Medium	Fast	Very Fast
Tracked	Road	.01	.30	.69	0	0
	Off Road	.05	.94	.01	0	0
	Very Rough	.20	.80	0	0	0
Wheeled	Road	.01	.05	.25	.65	.04
	Off Road	.40	.60	0	0	0
	Very Rough	.20	.20	.20	.20	.20
Non Vehicle	Road	.99	.01	0	0	0
	Off Road	.99	.01	0	0	0
	Very Rough	.99	.01	0	0	0

Identification of vehicles is often aided by the background knowledge of vehicles common to the area of interest. Table C.2 captures the assumed proportion of vehicle-like objects in the area of interest.

Table C.2 –Vehicle Population

Tracked	Wheeled	Non Vehicle
.45	.50	.05

C.3.2 Sensors

One of the aircraft sensors is a Moving Target Indicator (MTI) which provides an approximate position and velocity for moving targets, but cannot see stationary objects. Table C.3 characterizes performance for the MTI sensor.

Table C.3 – MTI Performance

	Slow	Medium	Fast	No Report
Stationary	.01	.01	.01	.97
Slow	.70	.19	.01	.10
Medium	.10	.70	.12	.08
Fast	.05	.20	.70	.05
Very Fast	.01	.05	.89	.05

The aircraft also carries an imaging sensor that distinguishes vehicles from other objects, and reliably distinguishes wheeled from tracked vehicles. Unfortunately, cloud cover can interfere with the ability of the imaging sensor to operate correctly. Table C.4 provides performance data for the imaging sensor against expected objects in varying weather conditions.

Table C.4 – Imaging Sensor Performance

		Tracked	Wheeled	No-vehicle
Tracked	Clear	.80	.15	.05
	Cloudy	.60	.30	.10
Wheeled	Clear	.10	.80	.10
	Cloudy	.20	.60	.20
Non-vehicle	Clear	.05	.05	.90
	Cloudy	.15	.15	.70

The aircrew also has access to a Geographic Information System (GIS) that compares estimated target position to a database that provides terrain characterization of road, off-road, or very-rough terrain. Occasionally incorrect reports are generated due to system position error or corrupted data in the terrain database. Likelihoods representing GIS performance are collected in Table C.5.

Table C.5 – Likelihood of GIS Report of Terrain

	Rpt: Road	Rpt: Off-road	Rpt: Very Rough
Road	.85	.10	.05
Off-road	.05	.85	.10
Very Rough	.05	.10	.85

C.3.3 Environment

The region of interest is dominated by road networks, off-road terrain, and very rough terrain. Table C.6 captures the environment for the region of interest.

Table C.6 – Prior Distribution of Terrain in Region of Interest

Road	Off-road	Very Rough
.41	.31	.28

Finally, weather in the region of interest is generally good. The ratio of clear to cloudy weather is summarized in Table C.7.

Table C.7 – Weather in the Area of Interest

Clear	Cloudy
.75	.25

C.4 Bayesian Network

The model shown in Figure C.1 illustrates a Bayesian network representing the Vehicle Identification model and may be used to aid in construction or testing of the probabilistic ontology. This may be used to assist you in creating and testing the probabilistic ontology.

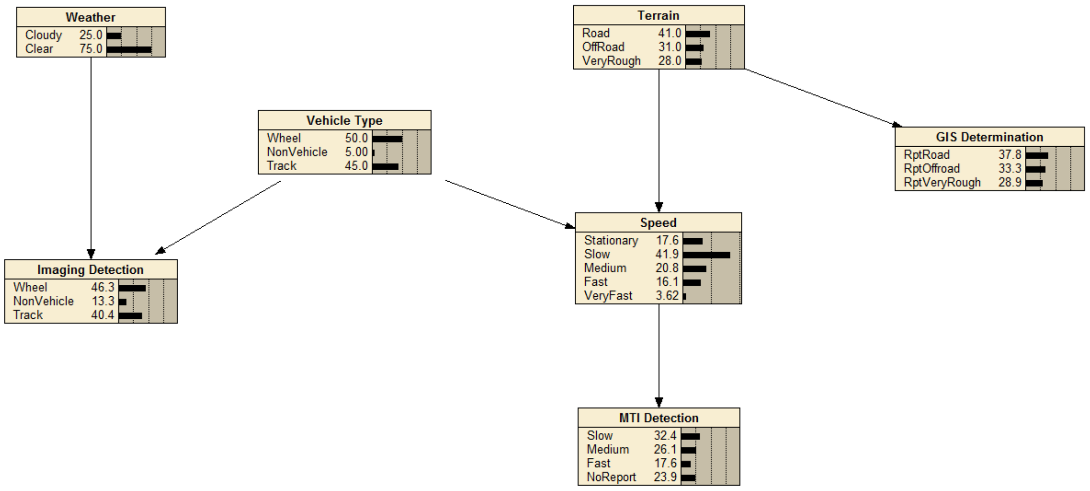


Figure C.1 – Vehicle ID Bayesian Network

C.5 Probabilistic Ontology Development Methodology Solution

C.5.1 Frame

C.5.1.1 Define the Spiral

C.5.1.1.1 Objective Statement:

To develop a probabilistic ontology for military vehicles that will infer vehicle type (wheeled or tracked) from MTI and imaging sensor reports, weather reports, and GIS reports.

C.5.1.1.2 Prime Query:

Which of the following vehicle types is the subject of incoming reports (wheeled, tracked, non-vehicle)?

C.5.1.2 Define Requirements

C.5.1.2.1 Requirements Table

Requirements		
ID	Name	Notes
1	Identify Vehicle Type	PQ: ID vehicle type (Wheeled, Tracked, Non-vehicle)
1.1	Account for terrain	Provide identification on various terrain types
1.2	Account for target speed	Provide identification at varying target speeds
2	Accept Reports	
2.1	Accept MTI Reports	Moving Target Imagery reports are based on target speed
2.2	Accept Imaging Reports	Imaging reports of un-obscured target are included
2.3	Accept Weather Reports	Weather affects imaging reports
2.4	Accept GIS Reports	GIS provides terrain-type reports (road, rough, very rough)
3	Recognize System Limitations	
3.1	Weather Limitation	Weather (clouds) negatively affects Imaging Reports
3.2	Speed Limitation	MTI does not report a stationary (speed=0) target
3.3	GIS Limitation	GIS includes system error

C.5.1.2.2 Individuals Table

Not specified in initial problem

Individuals	
Class	Instances
Contact	ctc1 / ctc2
Region	reg1 / reg2
GISReport	rptGIS1
ImageryReport	rptImage1
MTIReport	rptMTI1
Terrain	offroadTerrain / roadTerrain / veryRoughTerrain
VehicleType	nonVehicle / tracked / wheeled
NonVehicle	barricade
TrackedVehicle	tank
WheeledVehicle	truck / volvo
Weather	clearWeather / cloudyWeather

C.5.1.3 Define Metrics

C.5.1.3..1 Metrics Table

Metrics	
ID	Goal Metric
1	Correctly identify the vehicle type 85% of time

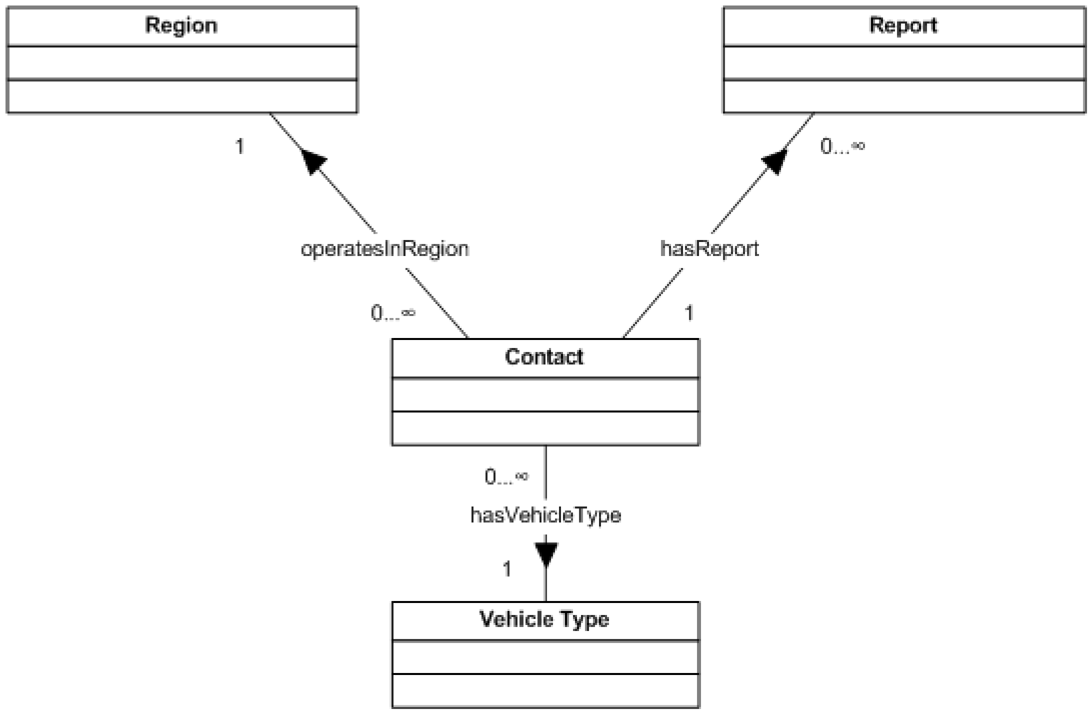
C.5.1.4 Identify Tier-one Attributes

C.5.1.4.1 Tier-one Attributes Table

Tier-1 Attributes	
ID	Tier-one Attribute
1	Imaging Report
2	Vehicle Speed

C.5.1.5 Draft Initial Class Diagram

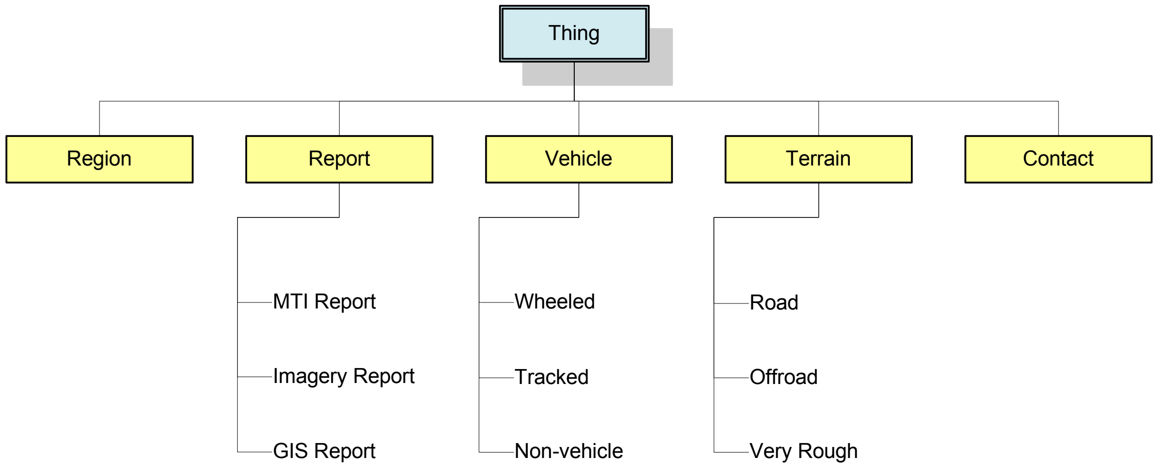
C.5.1.5.1 Initial Class Diagram

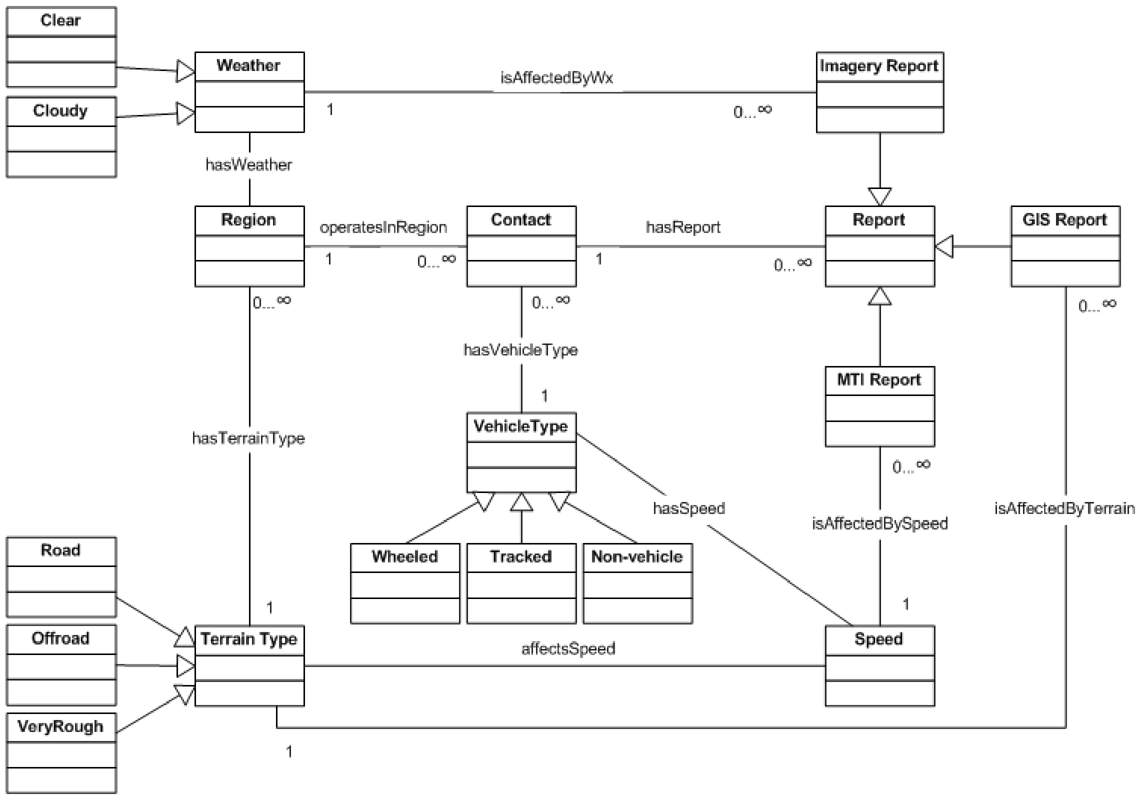


C.5.2 Ontology Development

C.5.2.1 Conduct Ontological Engineering

C.5.2.1.1 Taxonomy and Relationships



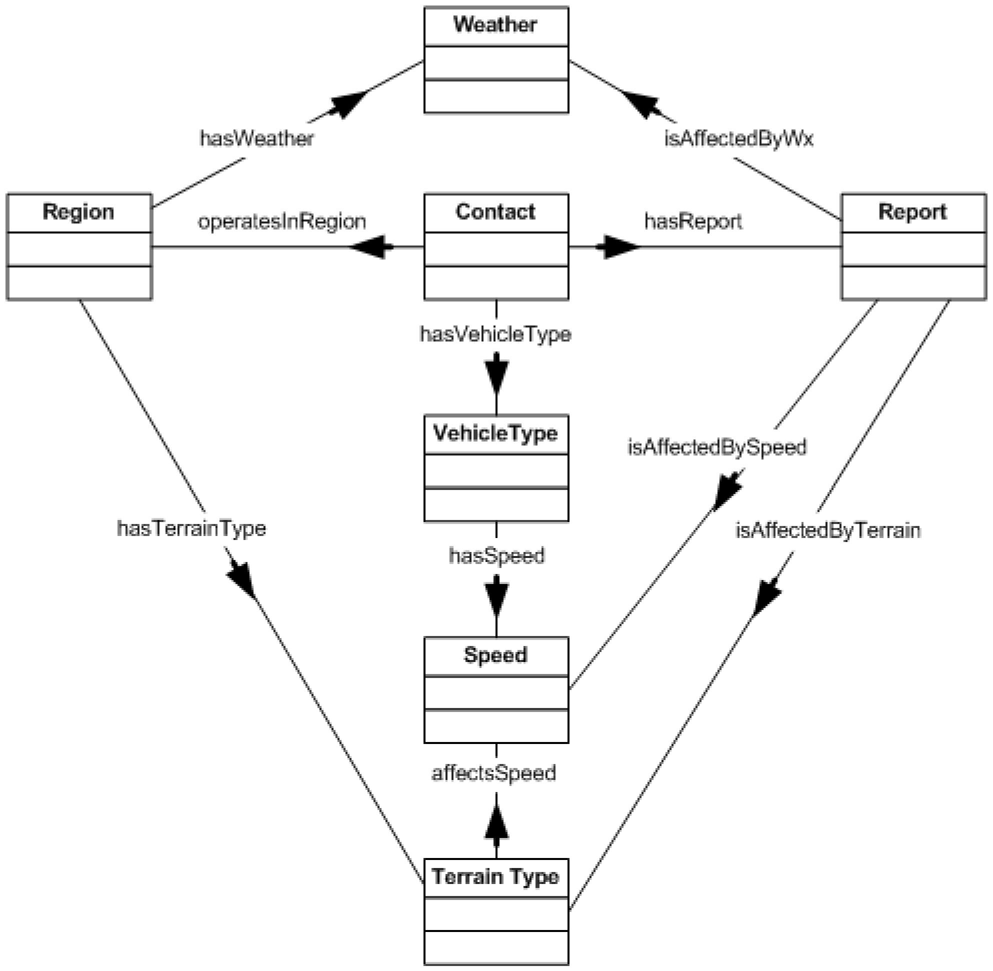


C.5.2.2 Research Usable Ontologies

C.5.2.2.1 Class Table

Classes					
Class	Object Property	Data Property	Relation	Domain	Range
Contact	vehicleType operatingRegion receivedReport	contactID contactSpeed	hasVehicleType hasOperatingRegion hasReceivedReport hasContactSpeed	Contact Contact Contact Contact	Vehicle Region Report <double>
Region	terrainType weatherType	regionID	hasTerrainType hasWeatherType	Region Region	Terrain Weather
GISReport	reportedTerrain	reportIDGIS	hasReportedTerrain	GISReport	Terrain
ImageryReport	reportedImage	reportIDImagery	hasReportedImage	ImageryReport	Vehicle
MTIReport		reportIDMTI reportedSpeed	hasReportedSpeed	MTIReport	<string>
NonVehicle		nonVehicleID			
TrackedVehicle		trackedVehicleID			
WheeledVehicle		wheeledVehicleID			

C.5.2.2.2 Complete Class Diagram



C.5.2.3 Research Heuristics and Algorithms

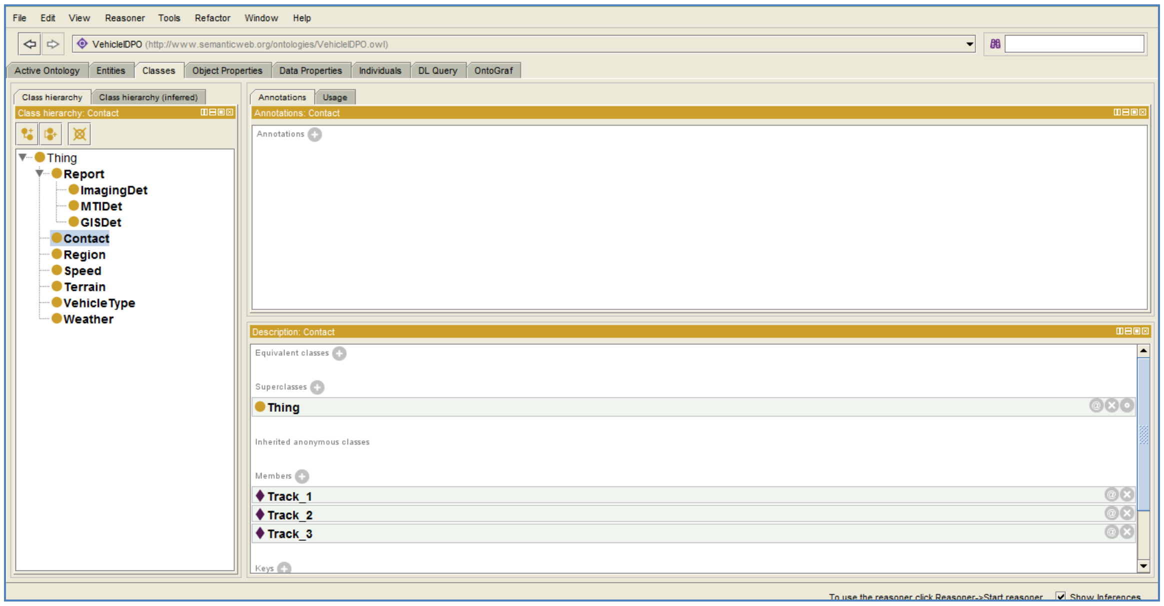
C.5.2.3.1 Formal Axiom and Rules Table

Axioms & Rules				
Axiom	Weather	MTI	Terrain	Region
Description	Weather affects imagery sensors	MTI produces "No-report" for stationary targets	A single Terrain Type is assigned to each Region	A Contact will operate in only one Region
Expression	NA	NA	NA	NA
Classes	Imaging Report Region	MTI Report Contact	Terrain Region	Contact Region
Relations	hasReportedImage	hasReportedSpeed	hasTerrainType	hasOperatingRegion
Variables	isClearWeather	contactSpeed	NA	NA

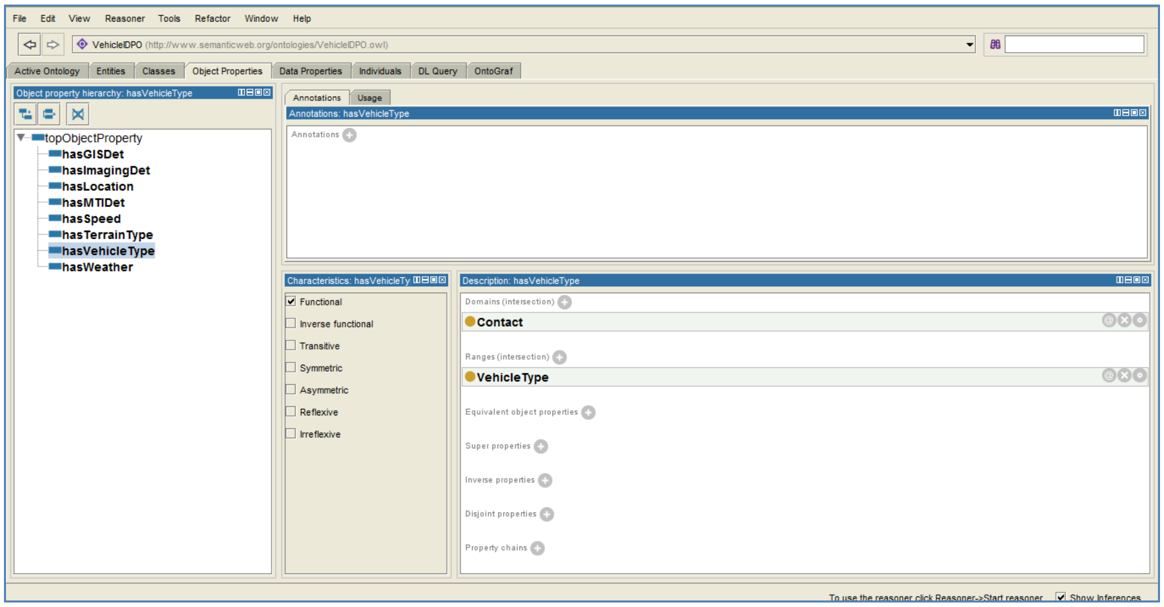
C.5.2.4 Implement Ontology Model

C.5.2.4.1 Operational Ontology

Create classes from taxonomy and provide annotation of each class


File Edit View Reasoner Tools Refactor Window Help
VehicleDPO (http://www.semanticweb.org/ontologies/VehicleDPO.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals DL Query OntoGraf
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Contact
Thing
Report
ImagingDet
MTIDet
GISDet
Contact
Region
Speed
Terrain
VehicleType
Weather
Annotations Usage
Annotations: Contact
Annotations
Description: Contact
Equivalent classes
Superclasses
Thing
Inherited anonymous classes
Members
Track_1
Track_2
Track_3
Keys
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Create Object Properties using relations from Class Table and provide annotation. Include domain and range.


File Edit View Reasoner Tools Refactor Window Help
VehicleIDPO (http://www.semanticweb.org/ontologies/VehicleIDPO.owl)
Active Ontology | Entities | Classes | Object Properties | Data Properties | Individuals | DL Query | OntoGraf
Annotations | Usage
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Object property hierarchy: hasVehicleType

- topObjectProperty
  - hasGISDet
  - hasImagingDet
  - hasLocation
  - hasMTIDet
  - hasSpeed
  - hasTerrainType
  - hasVehicleType
  - hasWeather

Annotations: hasVehicleType

Annotations

Characteristics: hasVehicleType

- [✓] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Description: hasVehicleType

Domains (intersection)
● Contact

Ranges (intersection)
● VehicleType

Equivalent object properties

Super properties

Inverse properties

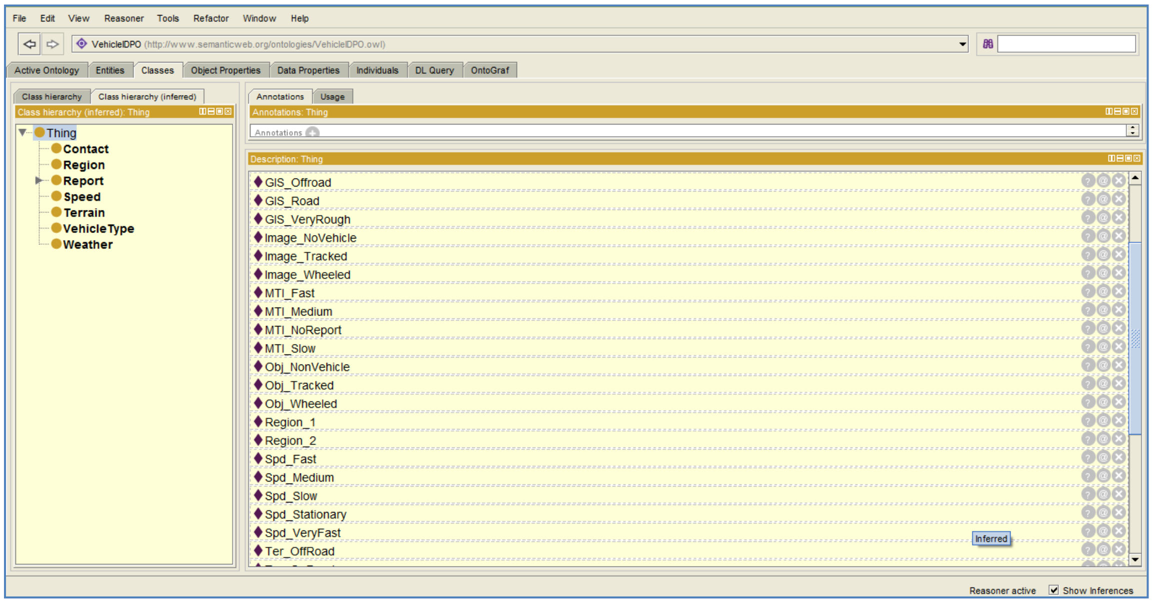
Disjoint properties

Property chains

Create Data Properties from information in the Class Table and provide annotation. Include domain and range information.

Not Required

Create Instances specified in the Evidence Table



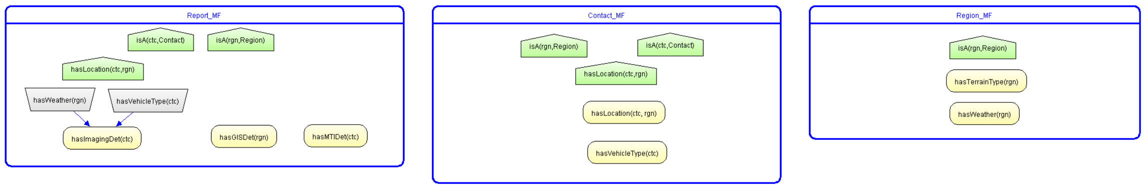
Thing
- Contact
- Region
- Report
- Speed
- Terrain
- VehicleType
- Weather

♦GIS_Offroad
♦GIS_Road
♦GIS_VeryRough
♦Image_NoVehicle
♦Image_Tracked
♦Image_Wheeled
♦MTI_Fast
♦MTI_Medium
♦MTI_NoReport
♦MTI_Slow
♦Obj_NonVehicle
♦Obj_Tracked
♦Obj_Wheeled
♦Region_1
♦Region_2
♦Spd_Fast
♦Spd_Medium
♦Spd_Slow
♦Spd_Stationary
♦Spd_VeryFast
♦Ter_OffRoad

C.5.3 Probability Incorporation

C.5.3.1 Core Model Generation

C.5.3.1.1 Create Model of Primary Query and Tier-1 Attributes



C.5.3.1.2 Populate LPD with Proxy Values

```
hasVehicleType
[
   Obj_Tracked = .45,
   Obj_Wheeled = .50,
   Obj_NonVehicle = .05
]
```

```
hasLocation
[
   Region_1 = .50,
   Region_2 = .50
]
hasWeather
[
   Wx_Clear = .75,
   Wx_Cloudy = .25
]

hasTerrainType
[
   Ter_OnRoad = 0.41,
   Ter_OffRoad = 0.31,
   Ter_VeryRough = 0.28
]

hasReportedImage
if any rgn have ( hasWeather = Wx_Cloudy ) [
   if any ctc have (hasVehicleType = Obj_Wheeled) [
      Image_Tracked = .20,
      Image_Wheeled = .60,
      Image_NoVehicle = .20
   ] else if any ctc have (hasVehicleType = Obj_NonVehicle) [
      Image_Tracked = .15,
      Image_Wheeled = .15,
      Image_NoVehicle = .70
   ] else if any ctc have (hasVehicleType = Obj_Tracked) [
      Image_Tracked = .60,
      Image_Wheeled = .30,
      Image_NoVehicle = .10
   ] else [
      Image_Tracked = .404,
      Image_Wheeled = .463,
      Image_NoVehicle = .133
   ]
] else if any rgn have ( hasWeather = Wx_Clear ) [
   if any ctc have (hasVehicleType = Obj_Wheeled) [
      Image_Tracked = .10,
      Image_Wheeled = .80,
      Image_NoVehicle = .10
   ] else if any ctc have (hasVehicleType = Obj_NonVehicle) [
      Image_Tracked = .05,
      Image_Wheeled = .05,
      Image_NoVehicle = .90
   ] else if any ctc have (hasVehicleType = Obj_Tracked) [
      Image_Tracked = .80,
      Image_Wheeled = .15,
      Image_NoVehicle = .05
   ] else [
      Image_Tracked = .404,
      Image_Wheeled = .463,
      Image_NoVehicle = .133
```

```
    ]
] else [
    Image_Tracked = .404,
    Image_Wheeled = .463,
    Image_NoVehicle = .133
]
```

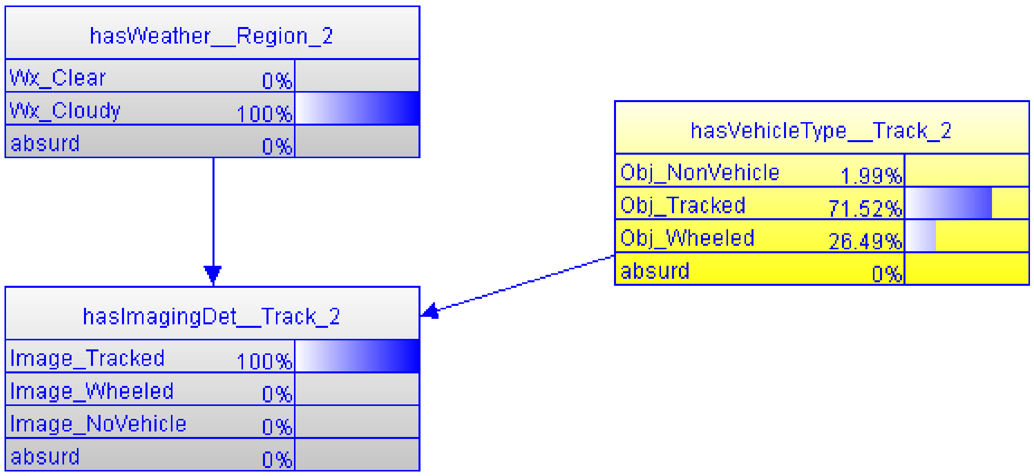
hasReportedGIS

```
[
    GIS_VeryRough = .60,
    GIS_Road = .30,
    GIS_Offroad = .10
]
```

hasReportedMTI

```
[
    MTI_Fast = .25,
    MTI_Slow = .25,
    MTI_NoReport = .25,
    MTI_Medium = .25
]
```

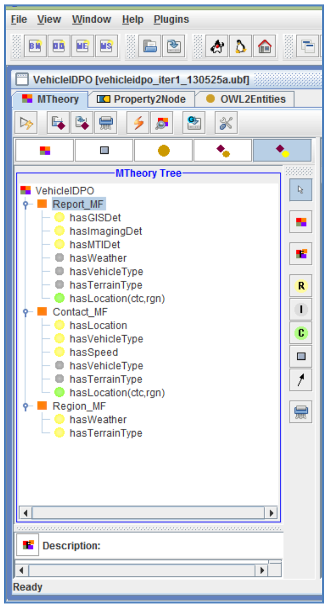
C.5.3.1.3 Create SSBN and Evaluate Logic



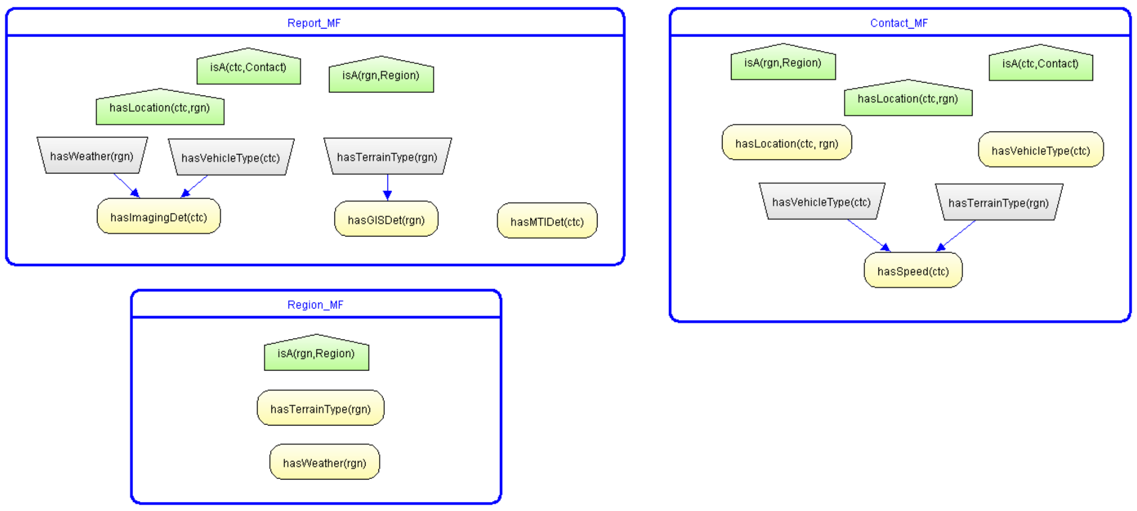
C.5.3.4 Probabilistic Ontology Construction Process

C.5.3.4.1 Iteration 1

Select and decompose a Tier-one Attribute into sub-attributes



Expand model to include new sub-attributes



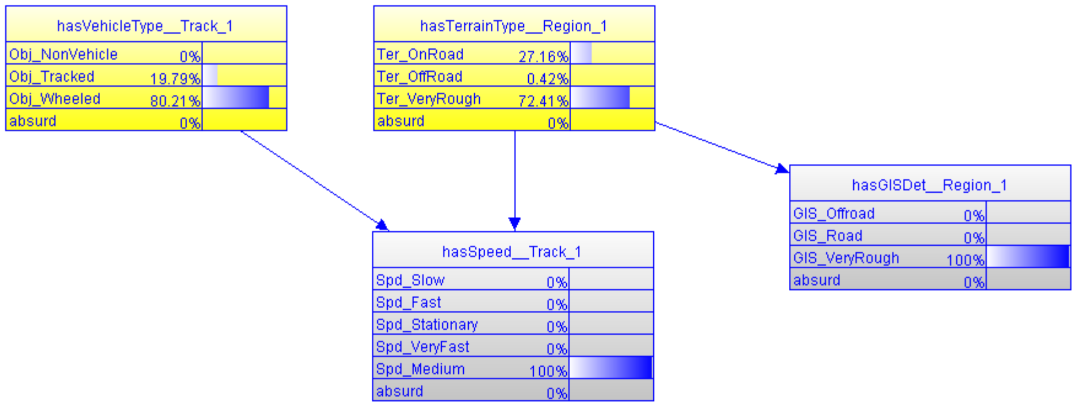
Populate LPD for new sub-attributes

```
hasGISDet
if any rgn have ( hasTerrainType = Ter_OnRoad ) [
   GIS_Offroad = .10,
   GIS_Road = .85,
   GIS_VeryRough = .05
] else if any rgn have ( hasTerrainType = Ter_OffRoad  ) [
   GIS_Offroad = .85,
   GIS_Road = .05,
   GIS_VeryRough = .10
] else if any rgn have ( hasTerrainType = Ter_VeryRough  ) [
   GIS_Offroad = .10,
   GIS_Road = .05,
   GIS_VeryRough = .85
] else [
   GIS_Offroad = .333,
   GIS_Road = .378,
   GIS_VeryRough = .289
]

hasSpeed
if any rgn have ( hasTerrainType = Ter_OnRoad ) [
   if any ctc have (hasVehicleType = Obj_Wheeled) [
      Spd_Slow = .05,
      Spd_Fast = .65,
      Spd_Stationary = .01,
      Spd_VeryFast = .04,
      Spd_Medium = .25
   ] else if any ctc have (hasVehicleType = Obj_NonVehicle) [
      Spd_Slow = .01,
      Spd_Fast = 0,
      Spd_Stationary = .99,
      Spd_VeryFast = 0,
      Spd_Medium = 0
   ] else if any ctc have (hasVehicleType = Obj_Tracked) [
      Spd_Slow = .30,
      Spd_Fast = 0,
      Spd_Stationary = .01,
      Spd_VeryFast = 0,
      Spd_Medium = .69
   ] else [
      Spd_Slow = .161,
      Spd_Fast = .325,
      Spd_Stationary = .059,
      Spd_VeryFast = .02,
      Spd_Medium = .435
   ]
] else if any rgn have ( hasTerrainType = Ter_OffRoad ) [
   if any ctc have (hasVehicleType = Obj_Wheeled) [
      Spd_Slow = .60,
      Spd_Fast = 0,
      Spd_Stationary = .40,
      Spd_VeryFast = 0,
      Spd_Medium = 0
   ] else if any ctc have (hasVehicleType = Obj_NonVehicle) [
      Spd_Slow = .01,
      Spd_Fast = 0,
      Spd_Stationary = .99,
      Spd_VeryFast = 0,
      Spd_Medium = 0
   ] else if any ctc have (hasVehicleType = Obj_Tracked) [
      Spd_Slow = .94,
      Spd_Fast = 0,
      Spd_Stationary = .05,
      Spd_VeryFast = 0,
      Spd_Medium = .01
   ] else [
      Spd_Slow = .724,
      Spd_Fast = 0,
      Spd_Stationary = .272,
      Spd_VeryFast = 0,
      Spd_Medium = .004
   ]
] else if any rgn have ( hasTerrainType = Ter_VeryRough ) [
   if any ctc have (hasVehicleType = Obj_Wheeled) [
      Spd_Slow = .20,
      Spd_Fast = .20,
      Spd_Stationary = .20,
      Spd_VeryFast = .20,
      Spd_Medium = .20
   ] else if any ctc have (hasVehicleType = Obj_NonVehicle) [
      Spd_Slow = .01,
      Spd_Fast = 0,
      Spd_Stationary = .99,
      Spd_VeryFast = 0,
      Spd_Medium = 0
   ] else if any ctc have (hasVehicleType = Obj_Tracked) [
      Spd_Slow = .80,
      Spd_Fast = 0,
      Spd_Stationary = .20,
      Spd_VeryFast = 0,
      Spd_Medium = 0
   ] else [
      Spd_Slow = .46,
      Spd_Fast = .10,
      Spd_Stationary = .24,
      Spd_VeryFast = .10,
      Spd_Medium = .10
   ]
] else [
   Spd_Slow = .419,
   Spd_Fast = .161,
   Spd_Stationary = .176,
   Spd_VeryFast = .036,
   Spd_Medium = .208
]
```

Update legacy LPD with new relationships

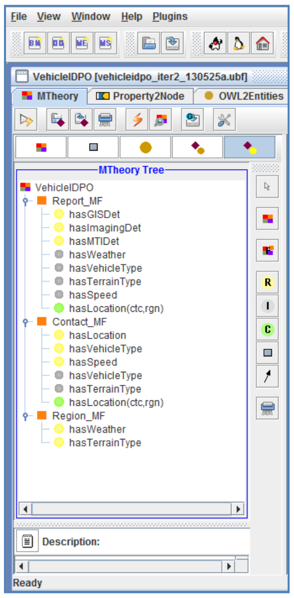
None Required

Create SSBN and evaluate logic



C.5.3.4.2 Iteration 2

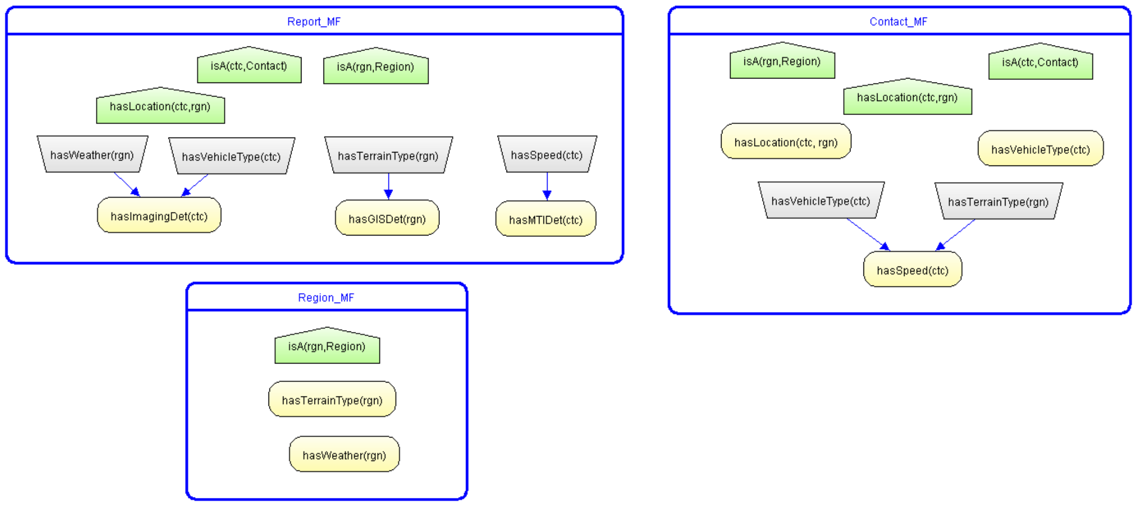
Select and decompose a Tier-one Attribute into sub-attributes



VehicleIDPO [vehicleidpo_iter2_130525a.ubf]

- VehicleIDPO
  - Report_MF
    - hasGISDet
    - hasImagingDet
    - hasMTIDet
    - hasWeather
    - hasVehicleType
    - hasTerrainType
    - hasSpeed
    - hasLocation(ctc,rgn)
  - Contact_MF
    - hasLocation
    - hasVehicleType
    - hasSpeed
    - hasVehicleType
    - hasTerrainType
    - hasLocation(ctc,rgn)
  - Region_MF
    - hasWeather
    - hasTerrainType

Expand model to include new sub-attributes



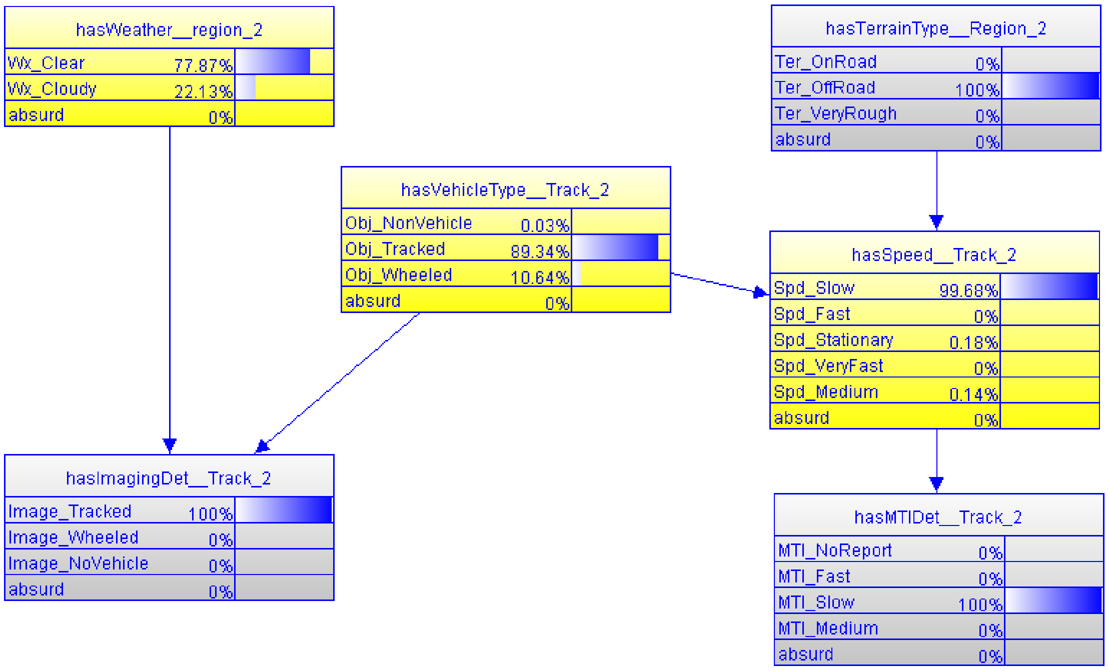
Populate LPD for new sub-attributes

```
hasMTIDet
if any ctc have ( hasSpeed = Spd_Stationary ) [
   MTI_NoReport = .97,
   MTI_Fast = .01,
   MTI_Slow = .01,
   MTI_Medium = .01
] else if any ctc have ( hasSpeed = Spd_Slow ) [
   MTI_NoReport = .10,
   MTI_Fast = .01,
   MTI_Slow = .70,
   MTI_Medium = .19
] else if any ctc have ( hasSpeed = Spd_Medium ) [
   MTI_NoReport = .08,
   MTI_Fast = .12,
   MTI_Slow = .10,
   MTI_Medium = .70
] else if any ctc have ( hasSpeed = Spd_Fast ) [
   MTI_NoReport = .05,
   MTI_Fast = .70,
   MTI_Slow = .05,
   MTI_Medium = .20
] else if any ctc have ( hasSpeed = Spd_VeryFast ) [
   MTI_NoReport = .05,
   MTI_Fast = .89,
   MTI_Slow = .01,
   MTI_Medium = .05
] else [
   MTI_NoReport = .237,
   MTI_Fast = .174,
   MTI_Slow = .33,
   MTI_Medium = .259
]
```

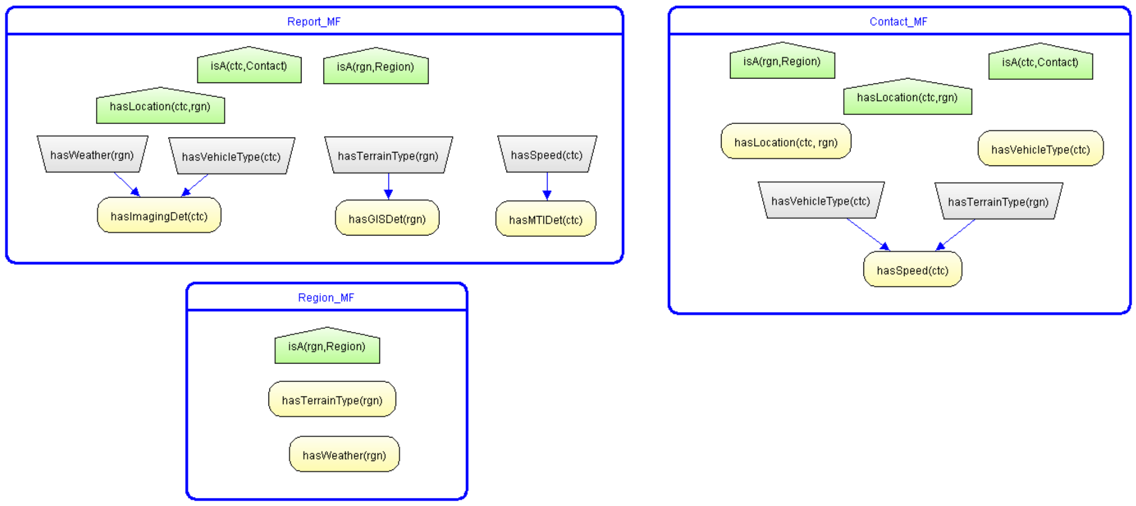
Update legacy LPD with new relationships

None required.

Create SSBN and evaluate logic



C.5.3.5 Completed Probabilistic Ontology



APPENDIX D: TERRORIST CREWMEMBER PROBLEM

D.1 Background

Crewmembers of merchant vessels are regularly multinational and transient. This is one possible way that terrorists or terror organizations can smuggle personnel or material into target countries. Using information about an individual crewmember's relations, influences and group associations may provide insight into the likelihood of that sailor being involved in terrorism. While some affiliations may increase the likelihood that an individual may join a terrorist group and attempt access to a target country via merchant ship, there is always the uncertainty that comes from the human condition. This uncertainty associated with the multitude of factors affecting the character's context must be captured conditionally.

D.2 Task

Considering the domain knowledge presented below, develop a probabilistic ontology that may be used to assess the likelihood that that an individual merchant ship crewmember is involved in terrorism given his relationships, influences and group associations.

D.3 Research

The following material is provided to assist you in developing and parameterizing your model.

D.3.1 Dependence and Independence Assumptions

Some appropriate assumptions are necessary to accommodate available data without compromising the utility of the model. It is assumed that there is a one percent (0.01) chance that any random person in the target demographic is a terrorist, as captured in Table D.1.

Table D.1 – Crewmember is a Terrorist (Prior)

Terrorist	
True	False
0.01	0.99

D.3.2 Relationships

Some of the greatest effects on terrorist recruits are their relationships with those that are close in their lives. The opinions, beliefs, and actions of these core people play a large part in recruiting future terrorists. Research shows that terrorists have a 75% chance of having a relationship with another terrorist. This relationship data is captured in Table D.2.

Table D.2 – Relationship Partition

Relationship		
Terrorist	Related	Not Related
True	.75	.25
False	.01	.99

Research shows that if a crewmember has a relationship with terrorists, there is a 68% chance that he has a friend who is a terrorist. The model should use 1% likelihood that an individual chosen at random is involved in terrorism. This friendship data is captured in Table D.3.

Table D.3 – Friendship with Terrorist

Friendship		
Relationship	Friendship	No Friendship
Related	.68	.32
Unrelated	.01	.99

As is the case in the *Friendship with Terrorist* node, there is always the possibility that a coincidental relationship brings an innocent individual into contact with a terrorist. However, given that a crewmember has a relationship with terrorists, there is a 14% chance that he is related to a terrorist. Table D.4 illustrates the associated likelihoods.

Table D.4 – Kinship with Terrorist

Kinship		
Relationship	Yes	No
Related	.14	.86
Unrelated	.01	.99

D.3.3 Influences

This node summarizes the likelihood of terrorism involvement due to the crewmember's life influences including the possibility he has witnessed personal tragedy associated with operations Enduring Freedom and Iraqi Freedom. The underlying assumption behind this partition is that an individual who chooses to become involved in terrorism has been negatively influenced in some way by his personal history. For a terrorist crewmember it is assumed with certainty that his history affects his decision to become involved. For non-terrorists from the demographic of interest, 20% likelihood is assigned to past history affecting this decision as shown in Table D.5.

Table D.5 – Influence Partition

Influence		
Terrorist	Influenced	Not Influenced
True	1	0
False	.20	.80

Operations Iraqi Freedom and Enduring Freedom have been controversial in some parts of the world. Knowing someone killed as a result of these operations could affect the likelihood of becoming involved in terrorism. The likelihood of a crewmember knowing someone detained or killed in these operations is captured in Table D.6.

Table D.6 Knows Imprisoned or Killed in US Operations

US Operations		
OEF/OIF Influence	True	False
Influenced	.75	.25
Not Influenced	.02	.98

D.3.4 Associations

Individuals may be involved in one or more organizations, some of which may be involved in terrorist activities. The association nodes summarize these likelihoods as shown below.

The crewmember has a prior probability of 1%, indicating that one in one-hundred individuals in the target demographic are members of a particular organization. This is shown in Table D.7.

Table D.7 – Member of Organization

Organization	
True	False
0.01	0.99

Very few organizations are actually involved in terrorism. If a crewmember is a member of an organization and a terrorist, it is likely that the organization is a terrorist organization. Otherwise, there is a very low likelihood of this case. Table D.8 illustrates the appropriate likelihoods.

Table D.8 – Terrorist Organization

Terrorist Organization			
Terrorist Crewmember	Member of Org	True	False
True	True	.90	.10
True	False	.1	.99
False	True	.1	.99
False	False	.1	.99

D.4 Bayesian Network

The model shown in Figure D.1 illustrates a Bayesian network representing the Terrorist Crewmember model and may be used to aid in construction or testing of the probabilistic ontology. This may be used to assist you in creating and testing the probabilistic ontology.

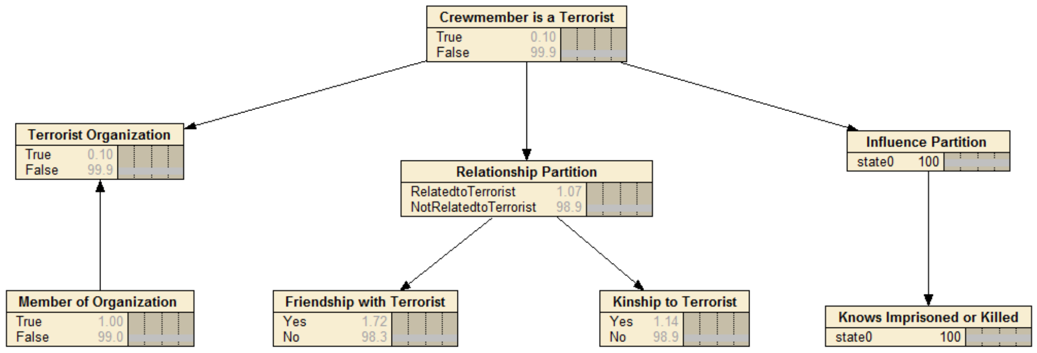


Figure D.1 – Terrorist Crewmember Bayesian Network

D.5 Probabilistic Ontology Development Methodology Solution

D.5.1 Frame

D.5.1.1 Define the Spiral

D.5.1.1.1 Objective Statement:

To develop a probabilistic ontology for a merchant vessel crewmember that will infer the likelihood of involvement in terrorism given his relationships and group associations.

D.5.1.1.2 Prime Query:

What is the likelihood that this individual crewmember is involved in terrorism?

D.5.1.2 Define Requirements

D.5.1.2.1 Requirements Table

Requirements		
ID	Name	Notes
1	Determine Terrorist Affiliation	PQ: Is Crewmember a terrorist?
1.1	Account for background demographics	Include the possibility that a crewmember could be a terrorist without the identifiers discussed
2	Account for Associations	Terrorist may be connected through associations
2.1	Account for Organizations	Crewmember may be member of organization, terrorist or otherwise
3	Account for Relationships	Relationships frequently define causality
3.1	Account for Friendships	Terrorist friends may indicate terrorist tendency
3.2	Account for Kinship	Terrorist kin may indicate terrorist tendency
4	Account for Influences	Crewmembers may be influenced by background
4.1	Account for OEF/OIF	Knowledge of US operations abroad may affect likelihood of terrorism

D.5.1.2.2 Individuals Table

Individuals	
Class	Instances
Crewmember	Crew_1 / Crew_2/ Crew_3
Organization	Org_1/ Org_2T/ Org_3/ Org_4T
Friend	Friend_1/ Friend_2T/ Friend_3
Family Member	Kin_1/ Kin_2T/ Kin_3

D.5.1.3 Define Metrics

D.5.1.3.1 Metrics Table

Metrics	
ID	Metric
M1	Correctly identify the vehicle type 85% of time

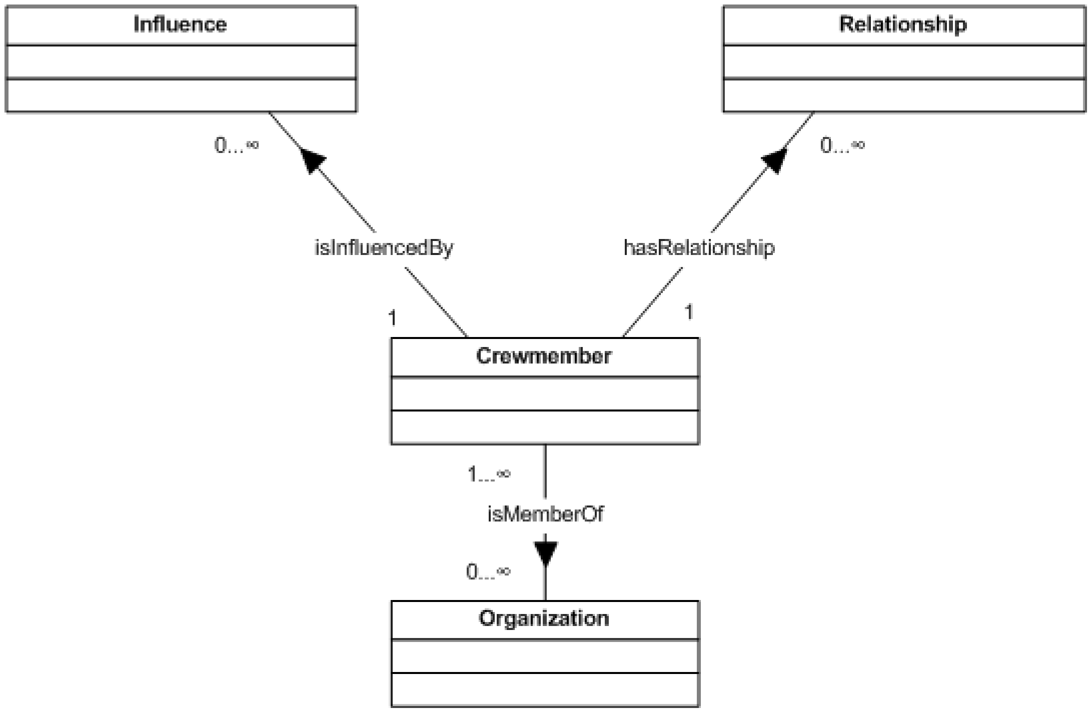
D.5.1.4 Identify Tier-one Attributes

D.5.1.4.1 Tier-one Attributes Table

Tier-1 Attributes	
ID	Tier-one Attribute
T1	Organization
T2	Relationship
T3	Influence

D.5.1.5 Draft Initial Class Diagram

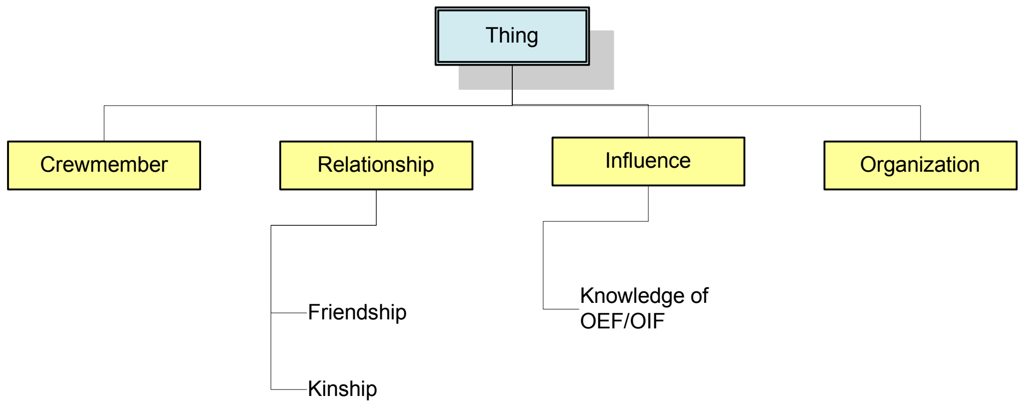
D.5.1.5.1 Initial Class Diagram

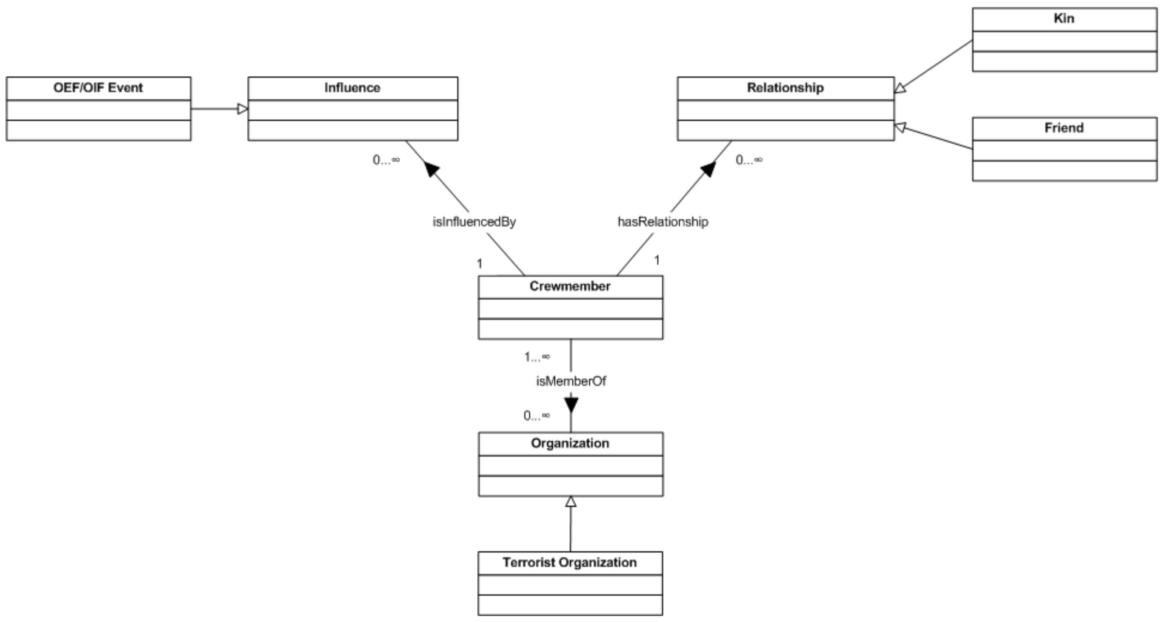


D.5.2 Ontology Development

D.5.2.1 Conduct Ontological Engineering

D.5.2.1.1 Taxonomy and Relationships



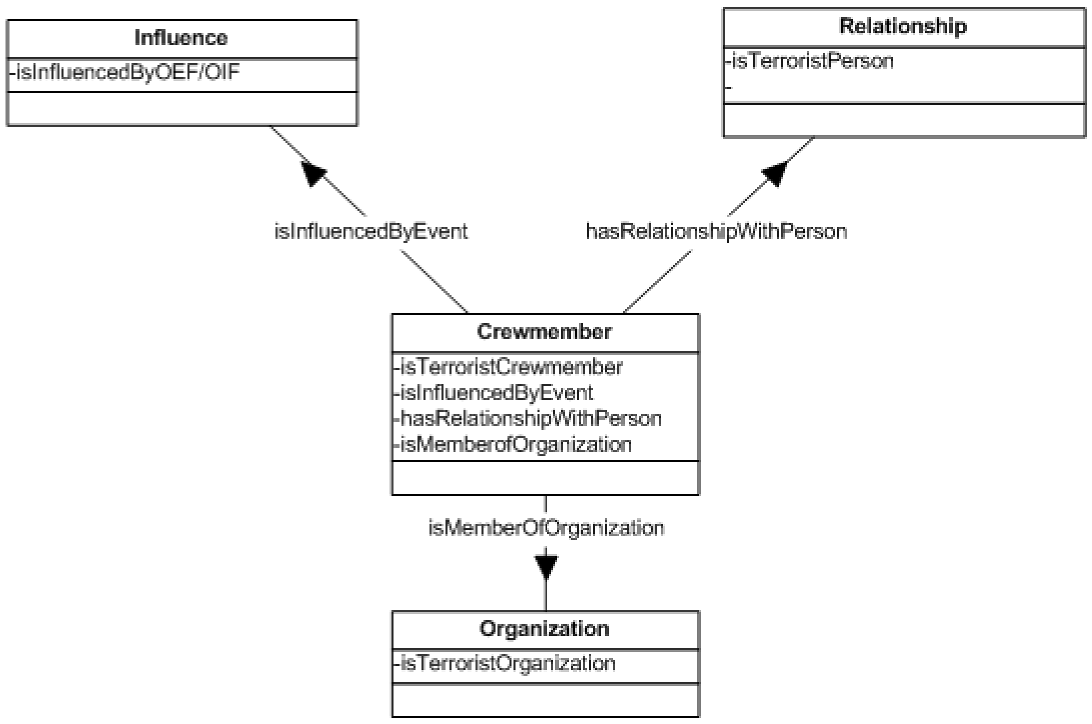


D.5.2.2Research Usable Ontologies

D.5.2.2.1 Class Table

Classes				
Class	Object Property	Data Property	Domain	Range
Crewmember		isTerroristCrewmember isInfluencedByOEF/OIF isInfluencedByEvent hasRelationshipWithPerson isMemberofOrganization hasTerroristFriend hasTerroristKin	Crewmember Crewmember Crewmember/Event Crewmember/Relationship Crewmember/Organization Crewmember Crewmember	Boolean Boolean Boolean Boolean Boolean
Relationship		isTerroristPerson	Relationship	Boolean
Organization		isTerroristOrganziation	Organization	Boolean

D.5.2.2.2 Complete Class Diagram



D.5.2.3 Research Heuristics and Algorithms

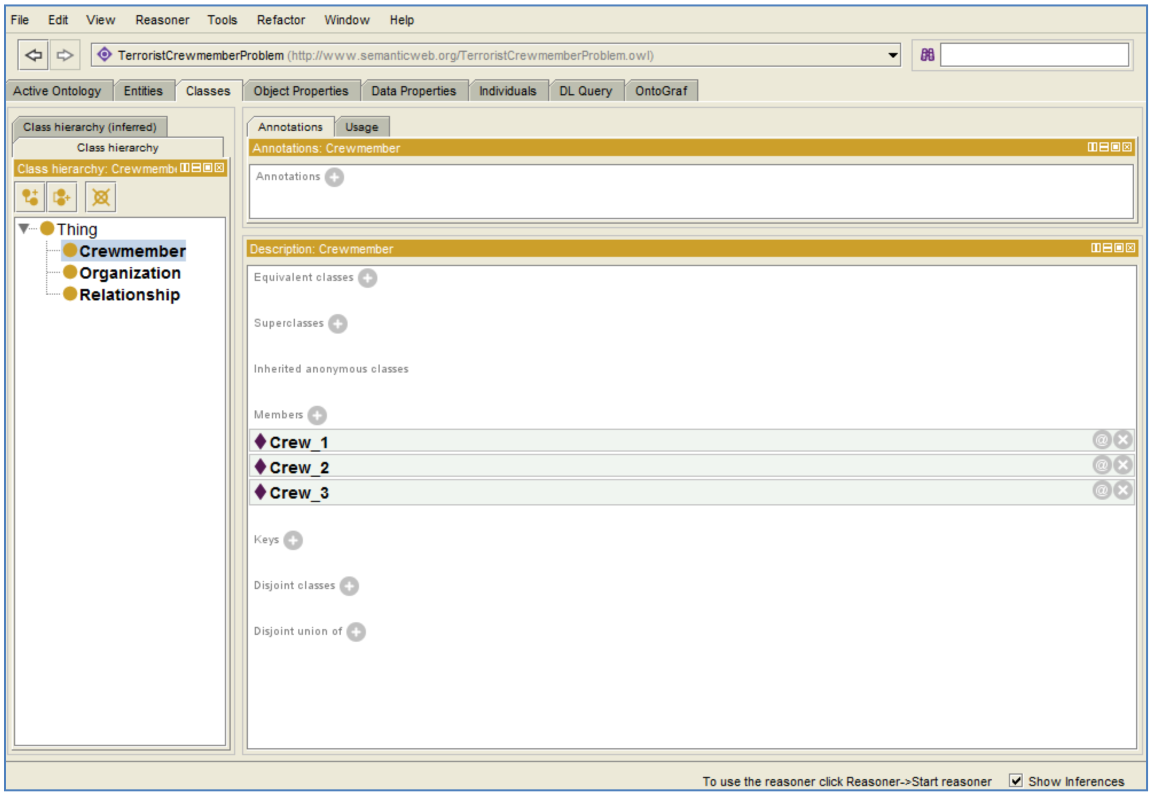
D.5.2.3.1 Formal Axiom and Rules Table

Axioms & Rules		
Axiom	Relationship	Influence
Description	Relationship with terrorist could be coincidental	It is assumed with certainty that history affects decision
Expression	NA	NA
Classes	Crewmember Relationship	Crewmember
Relations	hasRelationshipWithPerson	isInfluencedByEvent
Variables	boolean	boolean

D.5.2.4 Implement Ontology Model

D.5.2.4.1 Operational Ontology

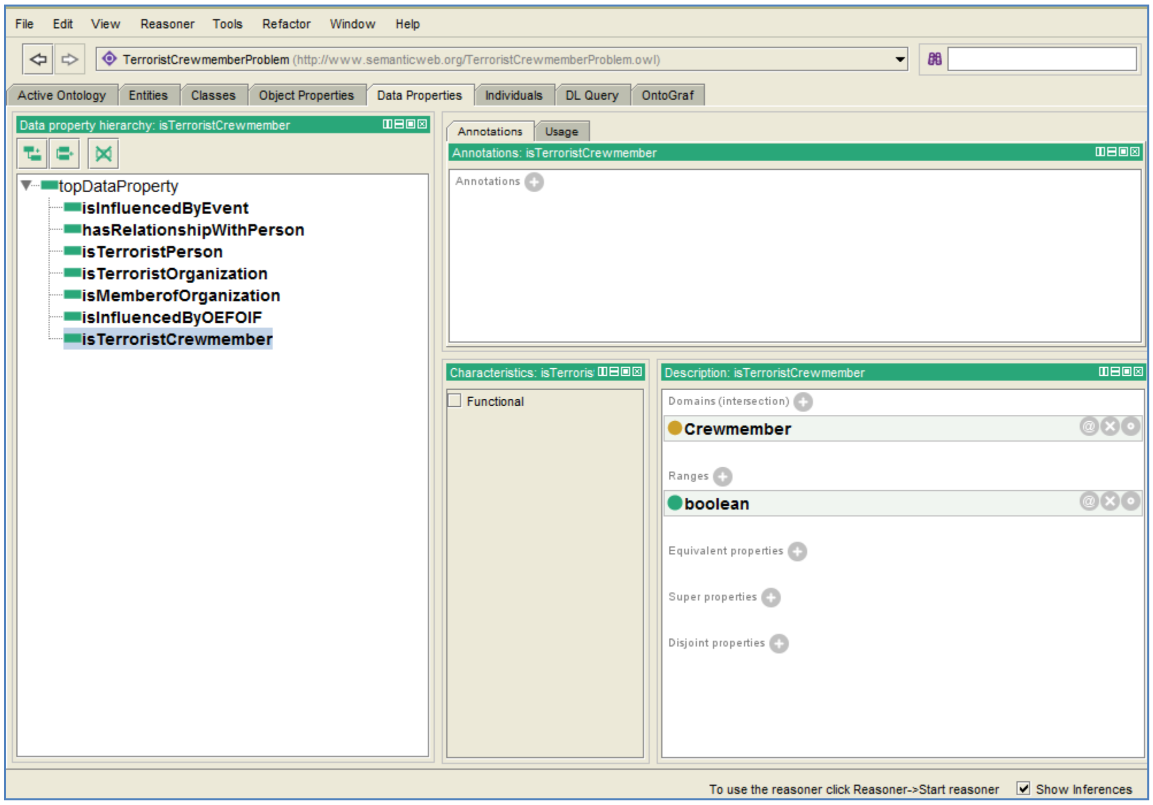
Create classes from taxonomy and provide annotation of each class.



Create Object Properties using relations from the Class Table and provide annotation. Include domain and range.

No object properties were used for this model.

Create Data Properties from information in the Class Table and provide annotation. Include domain and range information.


File Edit View Reasoner Tools Refactor Window Help
TerroristCrewmemberProblem (http://www.semanticweb.org/TerroristCrewmemberProblem.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals DL Query OntoGraf
Annotations Usage
Characteristics: isTerroris
Show Inferences

Data property hierarchy: isTerroristCrewmember

- topDataProperty
  - isInfluencedByEvent
  - hasRelationshipWithPerson
  - isTerroristPerson
  - isTerroristOrganization
  - isMemberofOrganization
  - isInfluencedByOEFOIF
  - isTerroristCrewmember

Annotations: isTerroristCrewmember

Annotations

Functional

Description: isTerroristCrewmember

Domains (intersection)

Crewmember

Ranges

boolean

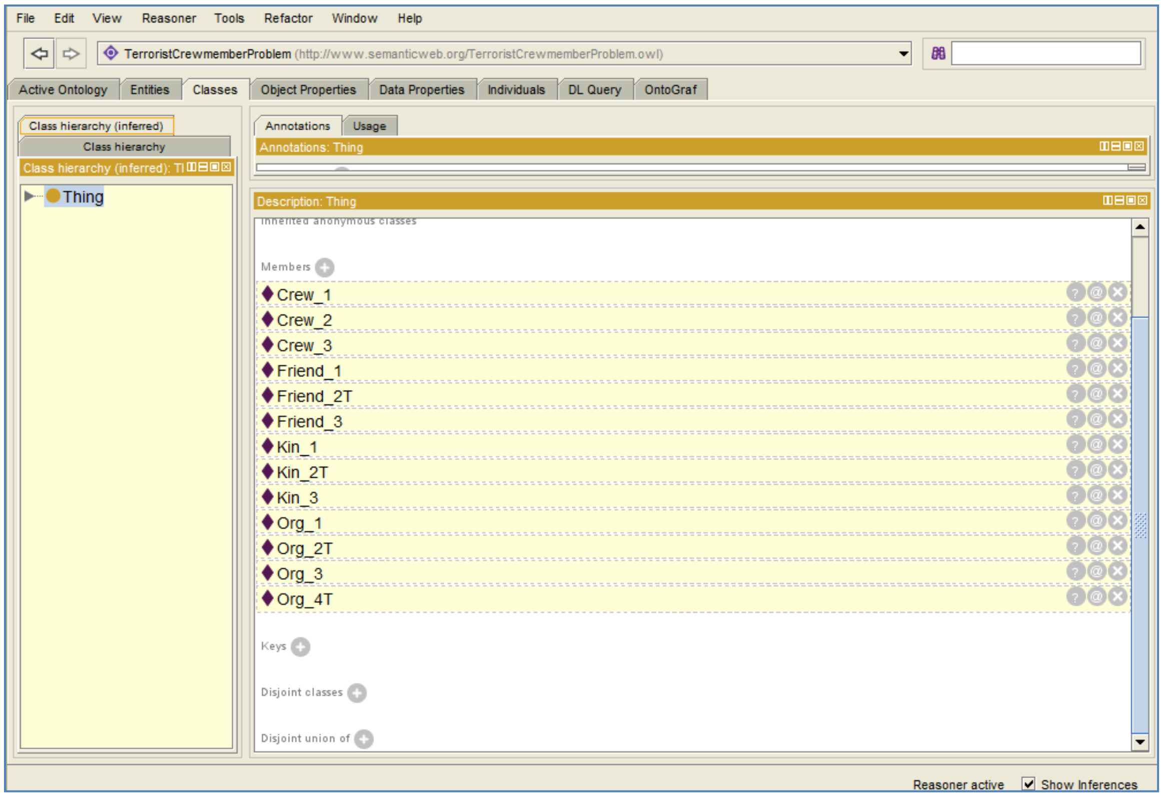
Equivalent properties

Super properties

Disjoint properties

To use the reasoner click Reasoner->Start reasoner

Create Instances specified in the Individuals Table.

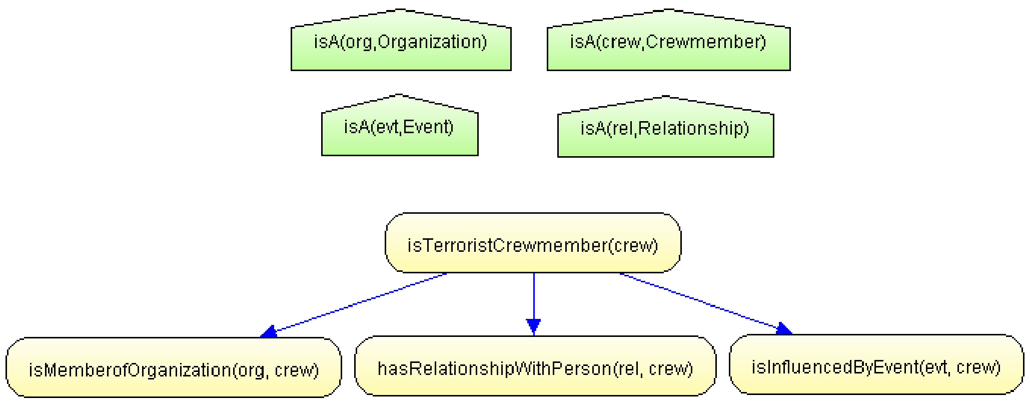

File Edit View Reasoner Tools Refactor Window Help
TerroristCrewmemberProblem (http://www.semanticweb.org/TerroristCrewmemberProblem.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals DL Query OntoGraf
Class hierarchy (inferred)
Class hierarchy
Class hierarchy (inferred): Thing
Thing
Annotations Usage
Annotations: Thing
Description: Thing
Inherited anonymous classes
Members
Keys
Disjoint classes
Disjoint union of
Reasoner active
Show Inferences

Crew_1
Crew_2
Crew_3
Friend_1
Friend_2T
Friend_3
Kin_1
Kin_2T
Kin_3
Org_1
Org_2T
Org_3
Org_4T

D.5.3 Probability Incorporation

D.5.3.1 Core Model Generation

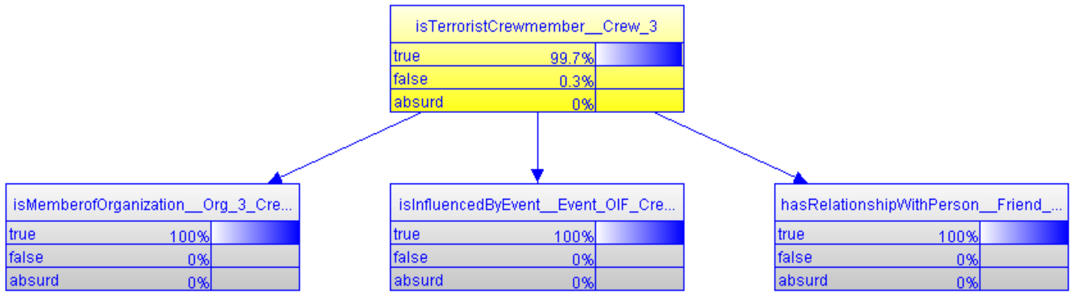
D.5.3.1.1 Create Model of Primary Query and Tier-1 Attributes


isA(org,Organization)
isA(crew,Crewmember)
isA(evt,Event)
isA(rel,Relationship)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)
hasRelationshipWithPerson(rel, crew)
isInfluencedByEvent(evt, crew)

D.5.3.1.2 Populate LPD with Proxy Values

```
isTerroristCrewmember
[
   true = .01,
   false = .99,
   absurd = 0
]

isMemberOfOrganization
if any crew have ( isTerroristCrewmember = true ) [
   true = .90,
   false = .10,
   absurd = 0
] else if any crew have ( isTerroristCrewmember = false ) [
   true = .01,
   false = .99,
   absurd = 0
] else [
   true = .002,
   false = .998,
   absurd = 0
]

hasRelationshipWithPerson
if any crew have ( isTerroristCrewmember = true ) [
   true = .75,
   false = .25,
   absurd = 0
] else if any crew have ( isTerroristCrewmember = false ) [
   true = .01,
   false = .99,
   absurd = 0
] else [
   true = .011,
   false = .989,
   absurd = 0
]

isInfluencedByEvent
if any crew have ( isTerroristCrewmember = true ) [
   true = 1,
   false = 0,
   absurd = 0
] else if any crew have ( isTerroristCrewmember = false ) [
   true = .20,
   false = .80,
   absurd = 0
] else [
   true = .201,
   false = .799,
   absurd = 0
]
```

D.5.3.1.3 Create SSBN and Evaluate Logic



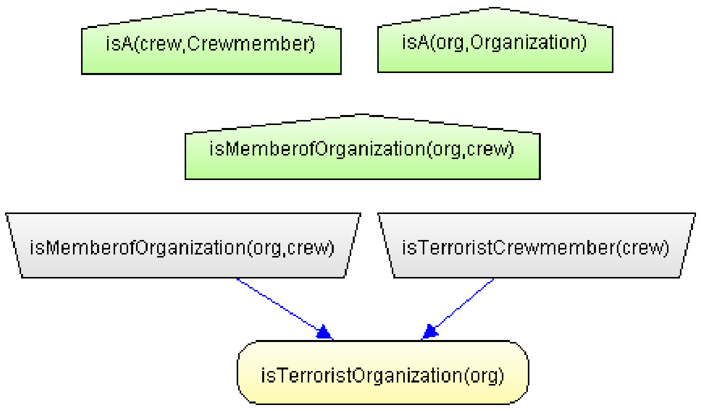
isTerroristCrewmember__Crew_3
true 99.7%
false 0.3%
absurd 0%
isMemberofOrganization__Org_3_Cre...
true 100%
false 0%
absurd 0%
isInfluencedByEvent__Event_OIF_Cre...
true 100%
false 0%
absurd 0%
hasRelationshipWithPerson__Friend_...
true 100%
false 0%
absurd 0%

D.5.3.2 Probabilistic Ontology Construction Process

D.5.3.2.1 Iteration 1

Select and decompose a Tier-one Attribute into sub-attributes

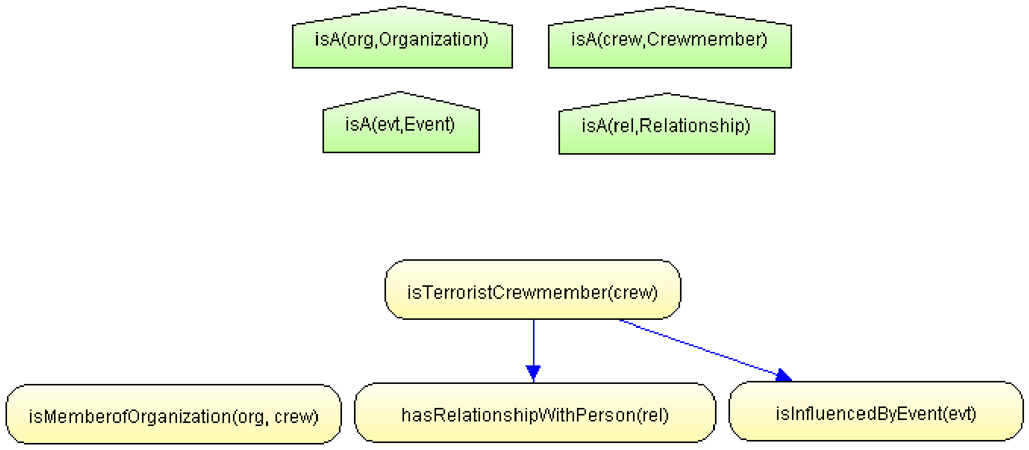
Organization_MFrag



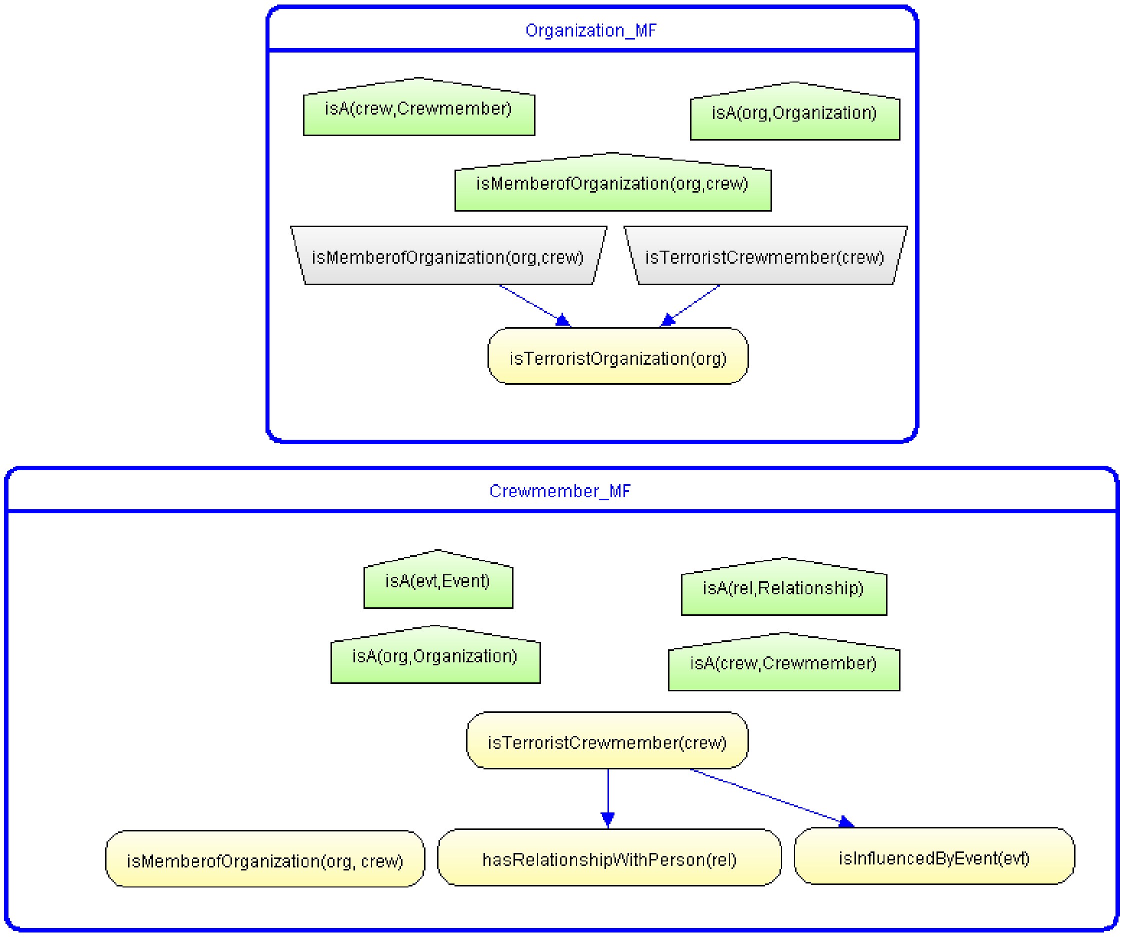
isA(crew,Crewmember)
isA(org,Organization)
isMemberofOrganization(org,crew)
isMemberofOrganization(org,crew)
isTerroristCrewmember(crew)
isTerroristOrganization(org)

Update model to include new sub-attributes

Crewmember_MFrag



Updated MTheory


Organization_MF
isA(crew,Crewmember)
isA(org,Organization)
isMemberofOrganization(org,crew)
isMemberofOrganization(org,crew)
isTerroristCrewmember(crew)
isTerroristOrganization(org)
Crewmember_MF
isA(evt,Event)
isA(rel,Relationship)
isA(org,Organization)
isA(crew,Crewmember)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)
hasRelationshipWithPerson(rel)
isInfluencedByEvent(evt)

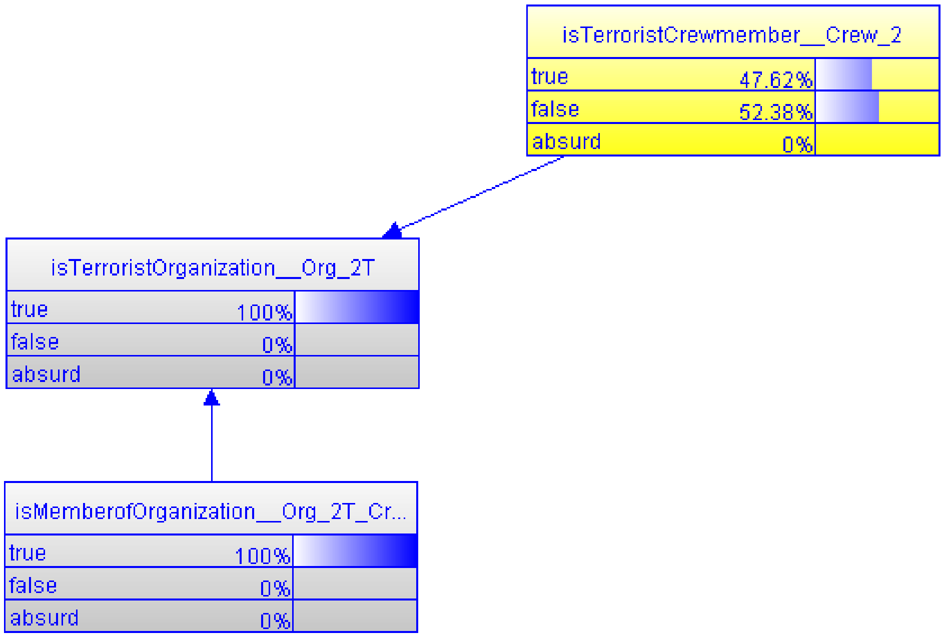
Populate LPD for new sub-attributes

```
isTerroristOrganization
if any crew have ( isTerroristCrewmember = true ) [
   if any org.crew have (isMemberofOrganization = true) [
      true = 0.90,
      false = 0.10
   ] else [
      true = 0.01,
      false = 0.99
   ]
] else if any crew have ( isTerroristCrewmember = false ) [
   if any org.crew have (isMemberofOrganization = true) [
      true = 0.01,
      false = 0.99
   ] else [
      true = 0.01,
      false = 0.99
   ]
] else [
   true = 0.01,
   false = 0.99
]
```

Update legacy LPD with new relationships

```
isMemberOfOrganization
[
   true = .01,
   false = .99,
   absurd = 0
]
```

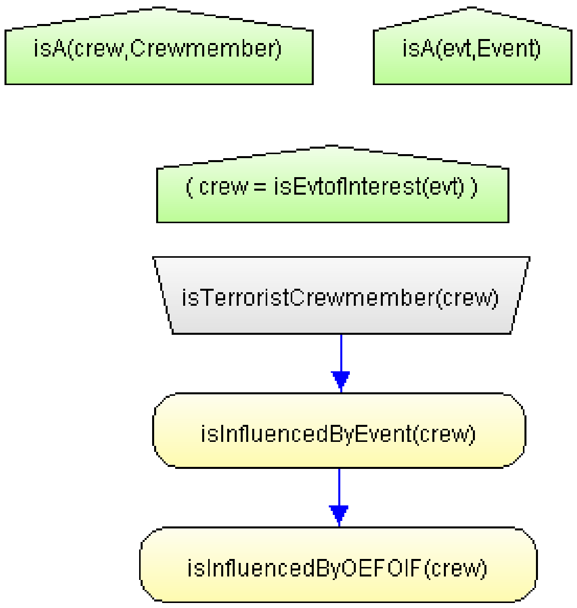
Create SSBN and evaluate logic


isTerroristCrewmember__Crew_2
true 47.62%
false 52.38%
absurd 0%
isTerroristOrganization__Org_2T
true 100%
false 0%
absurd 0%
isMemberofOrganization__Org_2T_Cr...
true 100%
false 0%
absurd 0%

D.5.3.2.2 Iteration 2

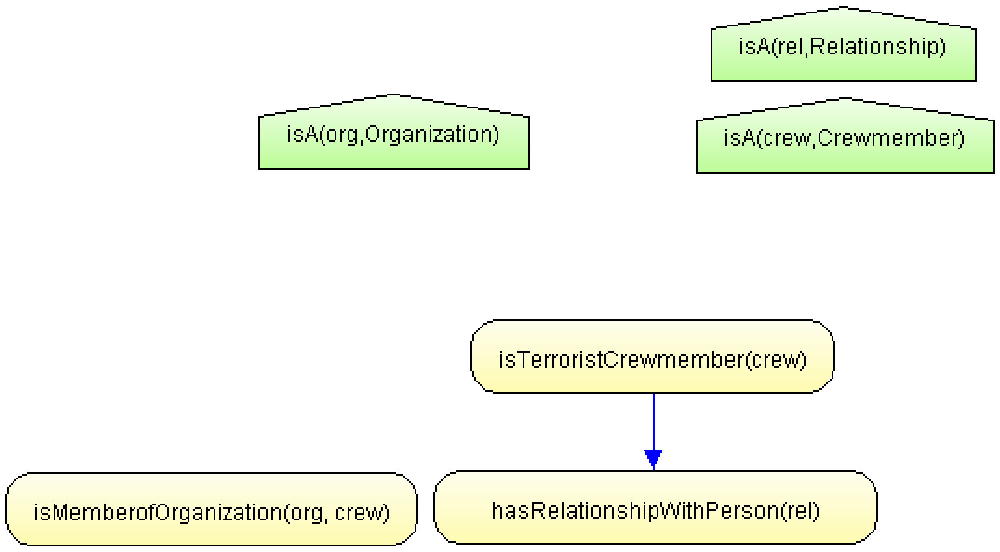
Select and decompose a Tier-one Attribute into sub-attributes

Influence_MFrag

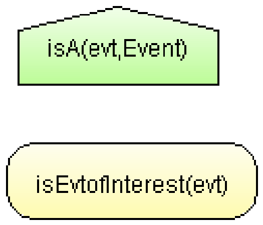

isA(crew,Crewmember)
isA(evt,Event)
(crew = isEvtofInterest(evt))
isTerroristCrewmember(crew)
isInfluencedByEvent(crew)
isInfluencedByOEFOIF(crew)

Update model to include new sub-attributes

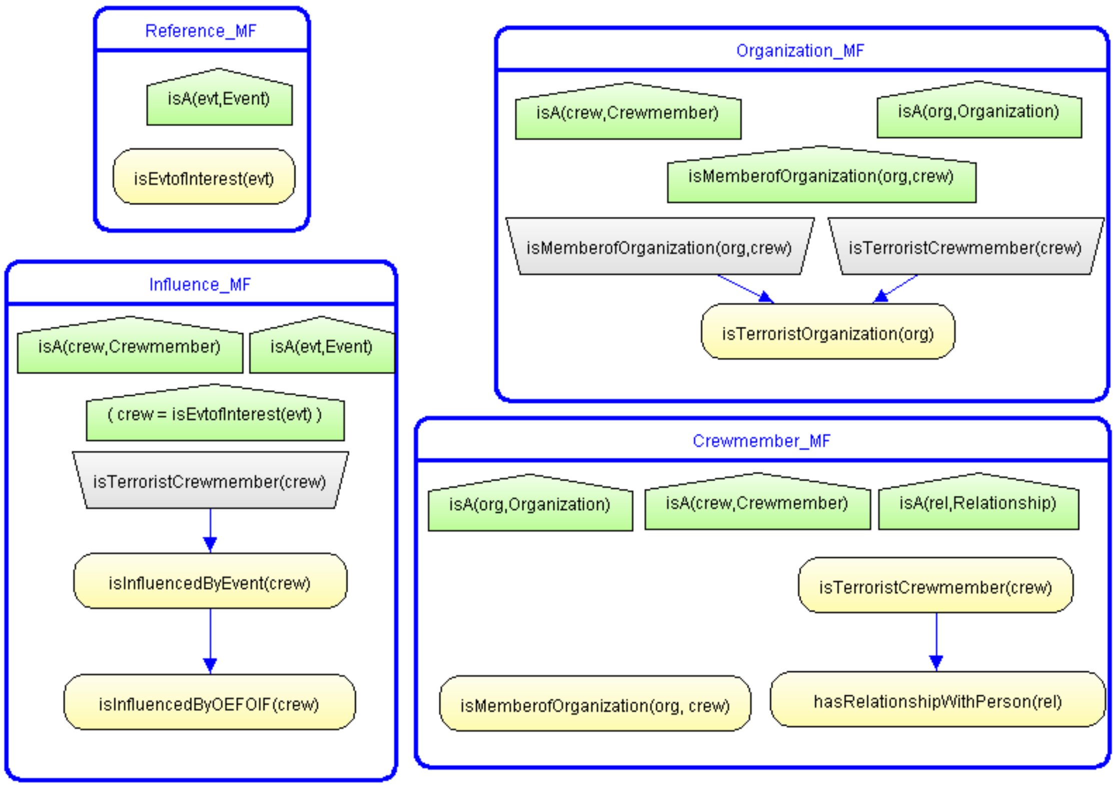
Crewmember_MFrag


isA(rel,Relationship)
isA(org,Organization)
isA(crew,Crewmember)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)
hasRelationshipWithPerson(rel)

Reference_MFrag



Updated MTheory


Reference_MF
isA(evt,Event)
isEvtofInterest(evt)
Organization_MF
isA(crew,Crewmember)
isA(org,Organization)
isMemberofOrganization(org,crew)
isMemberofOrganization(org,crew)
isTerroristCrewmember(crew)
isTerroristOrganization(org)
Influence_MF
isA(crew,Crewmember)
isA(evt,Event)
(crew = isEvtofInterest(evt))
isTerroristCrewmember(crew)
isInfluencedByEvent(crew)
isInfluencedByOEFOIF(crew)
Crewmember_MF
isA(org,Organization)
isA(crew,Crewmember)
isA(rel,Relationship)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)
hasRelationshipWithPerson(rel)

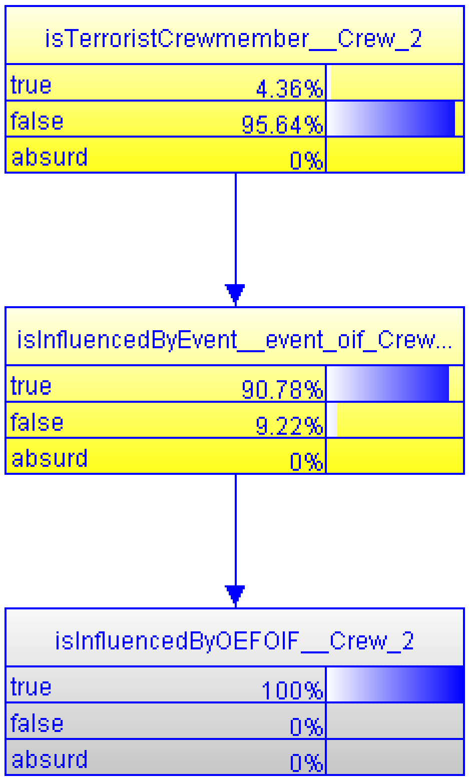
Populate LPD for new sub-attributes

```
isInfluencedByOEFOIF
if any crew have ( isInfluencedByEvent = true ) [
   true = .75,
   false = .25,
   absurd = 0
] else if any crew have ( isInfluencedByEvent = false ) [
   true = .02,
   false = .98,
   absurd = 0
] else [
   true = .167,
   false = .833,
   absurd = 0
]
```

Update legacy LPD with new relationships

No updates required.

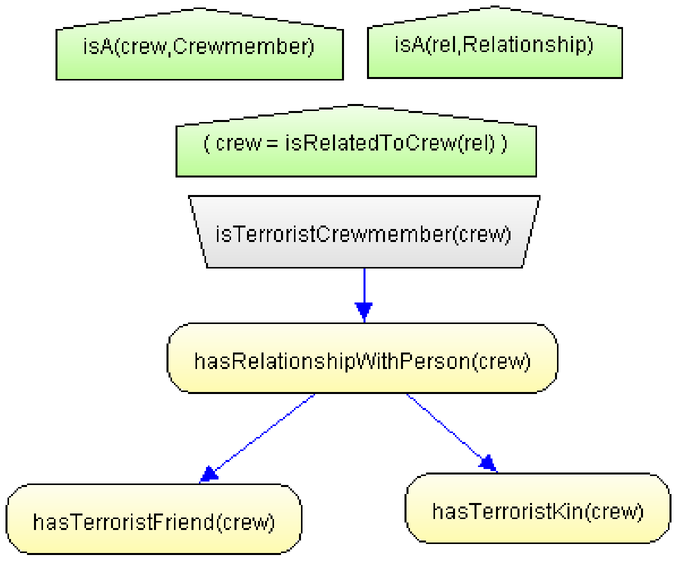
Create SSBN and evaluate logic


isTerroristCrewmember__Crew_2
true 4.36%
false 95.64%
absurd 0%
isInfluencedByEvent__event_oif_Crew...
true 90.78%
false 9.22%
absurd 0%
isInfluencedByOEFOIF__Crew_2
true 100%
false 0%
absurd 0%

D.5.3.2.3 Iteration 3

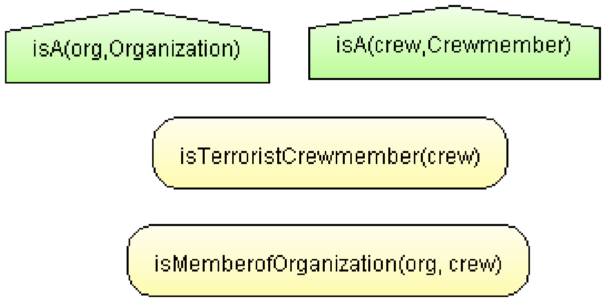
Select and decompose a Tier-one Attribute into sub-attributes

Relationship_MFrag

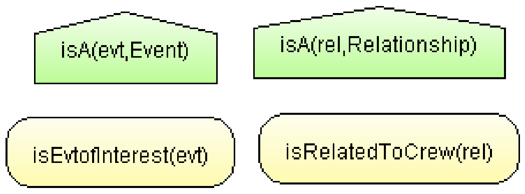

isA(crew,Crewmember)
isA(rel,Relationship)
(crew = isRelatedToCrew(rel))
isTerroristCrewmember(crew)
hasRelationshipWithPerson(crew)
hasTerroristFriend(crew)
hasTerroristKin(crew)

Update model to include new sub-attributes

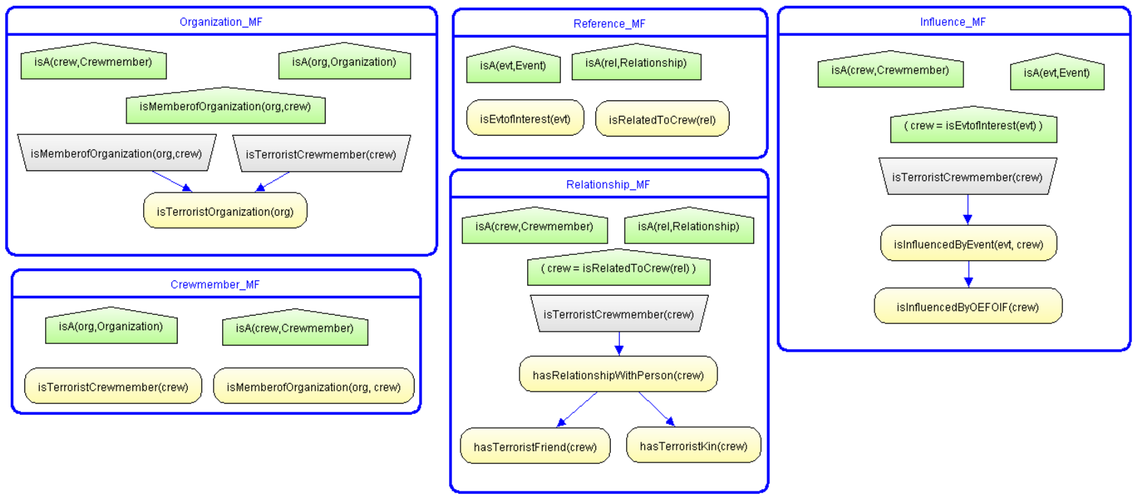
Crewmember_MFrag


isA(org,Organization)
isA(crew,Crewmember)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)

Reference_MFrag



Updated MTheory


Organization_MF
isA(crew,Crewmember)
isA(org,Organization)
isMemberofOrganization(org,crew)
isMemberofOrganization(org,crew)
isTerroristCrewmember(crew)
isTerroristOrganization(org)
Reference_MF
isA(evt,Event)
isA(rel,Relationship)
isEvtofInterest(evt)
isRelatedToCrew(rel)
Influence_MF
isA(crew,Crewmember)
isA(evt,Event)
(crew = isEvtofInterest(evt))
isTerroristCrewmember(crew)
isInfluencedByEvent(evt, crew)
isInfluencedByOEFOIF(crew)
Relationship_MF
isA(crew,Crewmember)
isA(rel,Relationship)
(crew = isRelatedToCrew(rel))
isTerroristCrewmember(crew)
hasRelationshipWithPerson(crew)
hasTerroristFriend(crew)
hasTerroristKin(crew)
Crewmember_MF
isA(org,Organization)
isA(crew,Crewmember)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)

Populate LPD for new sub-attributes

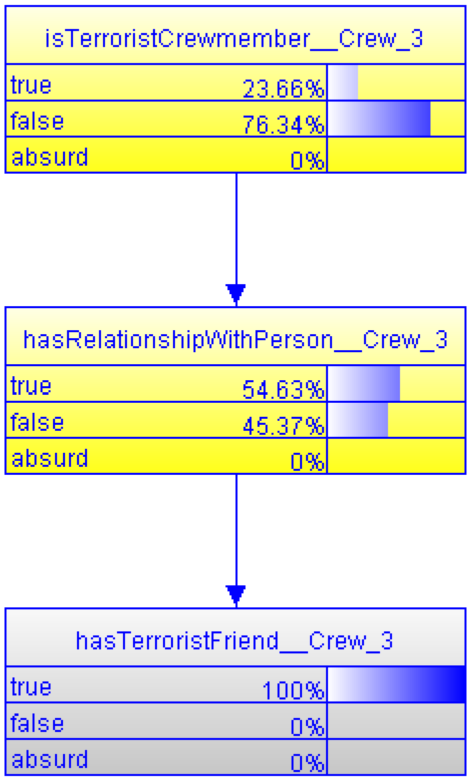
```
hasTerroristFriend
if any crew have ( hasRelationshipWithPerson = true ) [
   true = .68,
   false = .32
] else if any crew have ( hasRelationshipWithPerson = false ) [
   true = .01,
   false = .99
] else [
   true = .011,
   false = .989
]
```

```
hasTerroristKin
if any crew have ( hasRelationshipWithPerson = true ) [
   true = .14,
   false = .86
] else if any crew have ( hasRelationshipWithPerson = false ) [
   true = .01,
   false = .99
] else [
   true = .012,
   false = .988
]
```

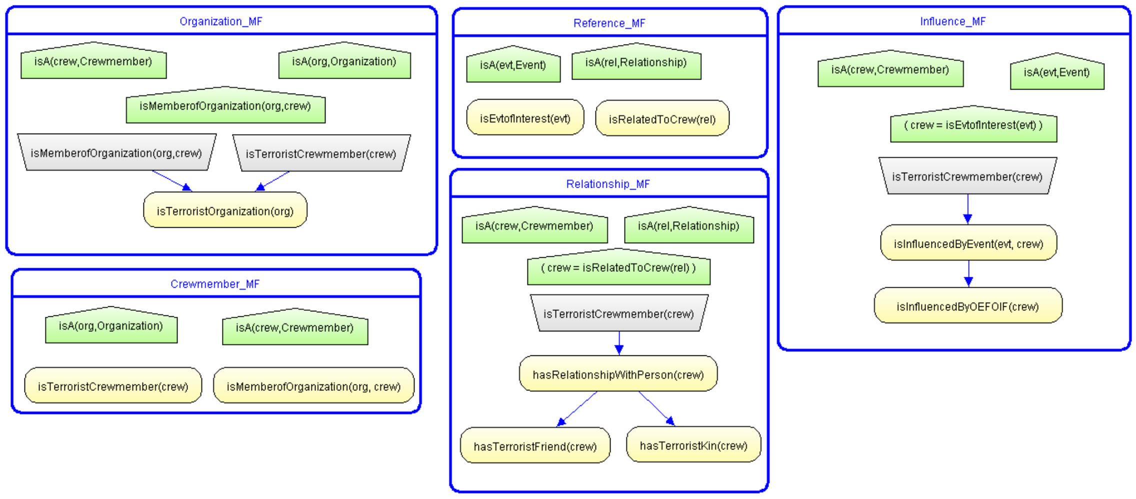
Update legacy LPD with new relationships

No update required.

Create SSBN and evaluate logic



D.5.3.3 Completed Probabilistic Ontology


Organization_MF
isA(crew,Crewmember)
isA(org,Organization)
isMemberofOrganization(org,crew)
isMemberofOrganization(org,crew)
isTerroristCrewmember(crew)
isTerroristOrganization(org)
Reference_MF
isA(evt,Event)
isA(rel,Relationship)
isEvtofInterest(evt)
isRelatedToCrew(rel)
Influence_MF
isA(crew,Crewmember)
isA(evt,Event)
(crew = isEvtofInterest(evt))
isTerroristCrewmember(crew)
isInfluencedByEvent(evt, crew)
isInfluencedByOEFOIF(crew)
Crewmember_MF
isA(org,Organization)
isA(crew,Crewmember)
isTerroristCrewmember(crew)
isMemberofOrganization(org, crew)
Relationship_MF
isA(crew,Crewmember)
isA(rel,Relationship)
(crew = isRelatedToCrew(rel))
isTerroristCrewmember(crew)
hasRelationshipWithPerson(crew)
hasTerroristFriend(crew)
hasTerroristKin(crew)

APPENDIX E: PARTICIPANT STATEMENT OF WORK

E.1 Introduction

E.1.1 Overview

As a Computer Science (CS) or Systems Engineering and Operations Research (SEOR) graduate student or graduate assistant, you have volunteered to participate in a Probabilistic Ontology (PO) development task. After downloading appropriate software and completing two tutorials, you will be given two probabilistic ontology development problems. Your primary task is completion of these problems as specified in the Work Breakdown Structure (WBS), below.

A brief demographic survey is required at the commencement of the study to capture participant experience and personal information. Your responses will be kept strictly confidential. During the tutorial sessions, a help desk will be available to answer any questions regarding the software. The purpose of the help desk is to assist with problems caused by the relative immaturity of the software used for their implementation. It will not be available for questions regarding hints or tips to solve the two case study problems. At the conclusion of the two tasks, each participant will complete a post-project survey to capture individual comments about the problems and recommendations for improvement.

The case study team consists of Dr. Kathryn Laskey in the role of Evaluator and Ph.D. Candidate Richard Haberlin in the role of Investigator. The primary mission of the Evaluator is to provide oversight to the conduct of the study and to ensure the interests of the institution are realized. The George Mason University Human Subject Review Board (HSRB) is chartered with ensuring that students conducting projects involving human subjects do so in a safe and ethical manner. The HSRB has approved this statement of work, and the Investigator will execute the case study protocol including compilation of all documentation, delivery of the software and tutorials, manning of the help desk, and compilation/analysis of data produced by the study. He will maintain regular correspondence with the Evaluator to ensure she is cognizant of all participant issues/concerns related to the project and progress of the study.

E.1.2 Consent Procedures

Informed consent will be obtained from each participant by the Investigator, under the oversight of the Evaluator. Each consent form will be scanned by the Investigator and a digital pdf copy will be sent to the participant.

E.1.3 Compensation

Upon satisfactory completion of all tasks within the WBS and delivery of required materials to the Investigator, the participant will receive a sum of $400 from the Evaluator.

E.2 Work Breakdown Structure

Participants will complete all tasks of the assigned WBS shown in Tables E.1 and E.1, detailed in the statement of work. Detailed instructions and appropriate worksheets for each of the tasks are provided to each participant.

Table E.1 – Work Breakdown Structure (Group 1)

Group I WBS		
0	**Complete Informed Consent Form**	App E.5
I	**Setup and Preparation**	
I.1	Install Protégé 4.1	App E.6.1
I.2	Install UnBBayes 4.11.4	App E.6.2
I.3	Complete entry survey	App E.7
II	**Training**	
II.1	Complete Protégé Tutorial	App E.8.1
II.2	Complete Patient Diagnosis Tutorial	App E.8.2
II.3	Review required documentation	App E.9
III	**Development**	
III.1	Design probabilistic ontology for Vehicle Identification Problem	App E.10 App C
III.2	Review the PODM	App E.12 Ch 4
III.3	Design probabilistic ontology for Terrorist Crewmember Problem	App E.11 App D
IV	**Documentation**	
IV.1	Capture task hours and Feedback	App E.9.2
IV.2	Complete exit survey	App E.13

Table E.2 – Work Breakdown Structure (Group 2)

Group II WBS		
0	**Complete Informed Consent Form**	App E.5
I	**Setup and Preparation**	
I.1	Install Protégé 4.1	App E.6.1
I.2	Install UnBBayes 4.11.4	App E.6.2
I.3	Complete entry survey	App E.7
II	**Training**	
II.1	Complete Protégé Tutorial	App E.8.1
II.2	Complete Patient Diagnosis Tutorial	App E.8.2
II.3	Review required documentation	App E.9
III	**Development**	
III.1	Design probabilistic ontology for Terrorist Crewmember Problem	App E.11 App D
III.2	Review the PODM	App E.12 Ch 4
III.3	Design probabilistic ontology for Vehicle Identification Problem	App E.10 App C
IV	**Documentation**	
IV.1	Capture task hours and Feedback	App E.9.2
IV.2	Complete exit survey	App E.13

E.4 Conduct of Case Study

It is critical that the tasks be completed and documented in accordance with this WBS to gather necessary data for the study. Contact the Investigator immediately via email for questions or clarification of any tasks. Completion of the entire SOW should take no more than 40 hours. A specific delivery date for materials will be coordinated between each participant and the Investigator when the study commences.

E.4.1 Tasks

All tasks are expected to be completed sequentially according to the WBS (Table E.1 or E.2). Detailed procedures for each task (except the development itself) are included in the appendices.

E.4.2 Software and Tutorials

Two software programs and associated tutorials are provided. The first is Protégé, an ontology development environment that the participant will use for ontology development and organization. The second is UnBBayes, a probabilistic ontology development tool that is the required tool for the final PO products. The tutorial associated with UnBBayes enacts the creation of a PO for Patient Diagnosis, commonly used for inferential reasoning instruction in academic environments.

E.4.3 Assistance

Comprehensive assistance will be available to all study participants through the completion of the Chest Clinic Probabilistic Ontology Tutorial and documentation review (SOW I – II.3). After these tasks, helpdesk records will be maintained to log assistance calls from participants regarding software issues. The goal is to keep the project moving forward while acknowledging the immature state of the software and the relative obscurity of the topic. Beginning with SOW Task III.1, the participant will be expected to complete the remaining SOW tasks without input regarding ontology/probabilistic ontology development. Use of external materials (e.g. books, reference materials, examples) is allowed. However, participants should not seek the assistance of other participants, students, or professors. The tutorials may be reviewed at any time.

E.4.4 Provided Materials

Participants will be given electronic access to all materials necessary to complete the assigned tasks, including documents, forms, surveys, software, and an explicit statement of work. It is assumed that each participant will have access to a personal computer with internet access and administrator permissions required to install the Protégé and UnBBayes software.

E.4.5 Documentation

Documentation of model construction, recommendations, and constructive criticism are of great value to the overall investigation. Participants are required to document their probabilistic ontology model and to complete the entry and exit surveys. Inclusion of additional comments, queries, and concerns are desired and add to the depth of the study.

E.5 Informed Consent Form

E.5.1 Research Procedures

This case study is being conducted to support Probabilistic Ontology development research. Your consent to participate commits you to complete all tasks in the WBS according to the Control Group Process Flow. A help desk is provided to submit queries regarding the software or the example questions. Brief survey questionnaires will be issued at the beginning and end of the study.

E.5.2 Risks

There are no foreseeable risks for participating in this research. Any confusion experienced by participants is unintentional and will be corrected by the Investigator at the earliest opportunity. This research has been reviewed according to George Mason University procedures governing your participation in this research.

E.5.3 Compensation

Upon successful completion of all WBS tasks and delivery of all materials to the Investigator, you will receive a payment of $400 from the SEOR Department.

E.5.6 Confidentiality

Personal data in this study will remain confidential. All correspondence, including help desk query response, will be direct between the participant and the Investigator. Only the Evaluator and Investigator will be able to link individuals to survey information and help desk queries. All materials provided by participants become the property of George Mason University and will remain under the cognizance of the Evaluator indefinitely.

E.5.7 Participation

Your participation is voluntary, and you may withdraw from the study at any time and for any reason, forfeiting all monetary compensation.

E.5.8 Contact

This research is being conducted by Richard Haberlin (SEOR) at George Mason University. He may be reached at 904-742-7624 for questions or to report a research-related problem. Oversight is provided by Dr. Kathryn Laskey (SEOR) who can be reached at 703-993-1644. You may contact the George Mason University Office of Research Subject Protections at 703-993-4121 if you have questions or comments regarding your rights as a participant in the research.

E.5.9 Consent

I have read this form and agree to participate in this study.

____

Name

____

Date of Signature

Version date: 20 December 2012

E.6 Protégé and UnBBayes Installation Guides

E.6.1 Installing Protégé 4.1

a. Proceed to the Protégé page http://protege.stanford.edu/download/download.html and register before downloading.

b. Continue to the download page and select the Protégé 4.1 release (support for OWL 2.0) appropriate for your OS

E.6.1.1 Windows Instructions:

Instructions

- After downloading, double-click **install_protege_4.1.exe**

Notes

- If you do not have a Java virtual machine installed, be sure to download the package above which includes one.

E.6.1.2 Mac OS X Instructions:

Instructions

- After downloading, double-click **install_protege_4.1**

Notes

- Be sure you have Java installed. You can download Java from Apple's site.
- The compressed installer should be recognized by Stuffit Expander and should automatically be expanded after downloading. If it is not expanded, you can expand it manually using StuffIt Expander 6.0 or later.
- If you have any problems launching the installer once it has been expanded, make sure that the compressed installer was expanded using Stuffit Expander. If you continue to have problems, please contact technical support.

E.6.2 Installing UnBBayes 4.11.4

a. Proceed to the SourceForge page and select "Download" at http://sourceforge.net/projects/unbbayes/

b. Save the **unbbayes-4.11.4-dist.zip** zip file to your computer

c. Unzip all the files into the Program Files directory

d. Navigate to

e. http://sourceforge.net/projects/unbbayes/files/UnBBayes%20Plugin%20Framework/Plugins/Probabilistic%20Networks/

f. and extract the following files into the **plugin folder** of the unbbayes-4.11.4 directory

- MEBN /1.13.1/unbbayes.prs.mebn-1.13.10-dist.zip
- MEBN /PR-OWL2/1.1.1/ unbbayes.gui.mebn.ontology.protege-1.1.1-ALPHA.zip
- MEBN /1.11.9/unbbayes.prs.mebn-1.11.9.zip
- MEBN/1.4.0/unbbayes.prs.mebn-1.4.0.zip
- PRM/0.1.1-ALPHA/unbbayes.prs.prm-0.1.1-ALPHA.zip
- OOBN/1.3.0/unbbayes.prs.oobn-1.3.0.zip
- MSBN/1.1.0/unbbayes.prs.msbn-1.1.0.zip

g. To execute UnBBayes, select the **unbbayes-4.11.1.jar** file

E.7 Entry Survey

Dr. K. B. Laskey / R. J. Haberlin

Email Address:_____________________________

Dear Participant: Thank you for taking time to fill out this confidential questionnaire. The information you provide will be solely used by the Evaluator and Investigator to capture data about probabilistic ontology development. Your email address will be used for correspondence should you use the help desk for assistance.

Q. Age

In what year were you born? _________

Q. Marital Status

What is your marital status?

o Now married

o Widowed

o Divorced

o Separated

o Never married

Q. Education

What is the highest degree you have completed?

o Associate Degree

o Bachelor of Arts

o Bachelor of Science

o Master of Arts

o Master of Science

o Doctor of Philosophy

o Professional degree (MD, DDS, DVM, LLB, JD)

Q. Education Major

What is your primary field of study? ______________________________

Q. Employment Status

Are you currently...?

o Full-time student

o Self-employed

o Full-time employed

o Retired

Q. Ethnicity

Please specify your ethnicity.

o Arab

o African American

o American Indian/Alaska Native

o Asian/Pacific Islander

o Caucasian/White

o Hispanic/Latino

o Indigenous/Aboriginal

o Other

Q. Language

What is your primary language?

o Arabic

o English

o Spanish

o Other ___________________________

Q. Ontology Knowledge

Describe your level of experience with ontologies and ontology development

o Unfamiliar

o Some familiarity

o Moderate familiarity/use

o Expert familiarity/use

o Please describe ___________________________

Q. Probabilistic Ontology Knowledge

Describe your level of experience with probabilistic ontologies

o Unfamiliar

o Some familiarity

o Moderate familiarity

o Expert familiarity

o Please describe ___________________________

E.8 Tutorials

E.8.1 Patient Diagnosis Ontology Protégé Tutorial

An ontology defines common language to share information about a given domain, allowing a common understanding of the structure and relationships about the information contained within. This tutorial initiates the user to the Protégé 4.1 ontology development tool. The tutorial will demonstrate use of the software to create a useable ontology to support diagnosis of a patient entering a clinic with a fever. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus. For brevity, the specific details of the tutorial are not repeated here, but can be found in Appendix B.2.

E.8.2 Patient Diagnosis Probabilistic Ontology Tutorial

A MEBN Theory (MTheory) allows the introduction of uncertainty into an ontology by implicitly expressing a joint probability distribution over groups of hypotheses that are globally consistent within the domain. This tutorial will demonstrate use of UnBBayes to create a useable Probabilistic Ontology to support diagnosis of a patient entering a clinic with a fever. If he has recently visited an area with a flu epidemic, that could be the cause. Otherwise the patient has another type of virus. The background ontology was previously created within Protégé 4.1 and saved as PatientDiagnosis.owl. For brevity, the specific details of the tutorial are not repeated here, but can be found in Appendix B.3.

E.9 Documentation Requirements

Rigorous documentation is critical to the success of this development analysis. By capturing details about modeling, time, and the software, the effectiveness and teachability of the PODM may be evaluated and the UnBBayes software tool further updated. For each of the three categories below, please provide your thoughts and opinions to the maximum extent possible. It may help to think of this SOW as a software development process, with stringent documentation requirements. Frank and honest feedback is greatly appreciated.

E.9.1 Modeling

E.9.1.1 Missing steps or confusion in methodology

The PODM is still under development. Detailed input on where it falls short in assisting in creation of a probabilistic ontology will provide the community with a better product. Additionally, confusion about the activities or task descriptions will ensure that these are corrected in the final form of the document. This particular feedback is of greatest importance to the study.

E.9.1.2 False Starts, Redirections and Dead-ends

Rare is the occasion in which a developer creates a model from start to finish without a false start, redirection, or dead end. Instead of deleting these occurrences, keep them intact for delivery with the completed SOW. This will help the investigator understand what parts of the PODM may require further embellishment or stronger example.

E.9.1.3 Ontology Descriptions

As the ontology is developed, include descriptions about the classes, properties and instances created to support the models. There is an annotation section within Protégé that is useful to complete this task.

E.9.1.4 Major Restrictions

If some aspect of the software or PODM restricted successful completion of the SOW, or led to a limited model, include details about this restriction.

E.9.1.5 Errors in logic or modeling

Difficulties in producing the desired output are often caused by logic errors. Include description of these errors, as well as those regarding creation of the model.

E.9.2 Time

Use of a prescribed methodology may reduce the time required to complete complex probabilistic ontology development. Provide feedback on required time to complete each of the SOW tasks using the table provided below.

Elapsed Time		
WBS Step	**Title**	**Elapsed Time**
0	**Complete Informed Consent Form**	
I	**Setup and Preparation**	
I.1	Install Protégé 4.1	
I.2	Install UnBBayes 4.11.4	
I.3	Complete entry survey	
II	**Training**	
II.1	Complete Protégé Tutorial	
II.2	Complete Patient Diagnosis Tutorial	
II.3	Review required documentation	
III	**Development**	
III.1	Design probabilistic ontology for Vehicle Identification Problem	
III.2	Review the PODM Execution Checklist	
III.3	Design probabilistic ontology for Terrorist Crewmember Problem	
IV	**Documentation**	
IV.1	Capture task hours and Feedback	
IV.2	Complete exit survey	

E.9.3 Software

E.9.3.1 Software Difficulties

The UnBBayes software tool is still under development. If there are procedures that you find cumbersome, provide a detailed description. Similarly, recommendations that will allow development within the tool to flow more smoothly will aid in focusing the next software update.

E.9.3.2 Software Bugs

Errors in program execution are considered bugs. Document these occurrences, and the factors leading to them so they may be corrected by the UnBBayes development team.

E.9.3.3 GUI Requests

Buttons, panes, screens, and other features that would aid a developer in creating probabilistic ontologies should be described in appropriate detail.

E.10 Vehicle Identification Problem

E.10.1 Background

Intelligence, surveillance and reconnaissance aircraft employ a suite of sensors to allow the operator to classify detected targets. In this problem, you must develop a decision support system that provides the most likely vehicle type (wheeled or tracked) based on incoming evidence. The model may be used to infer the vehicle type from MTI and imaging sensor reports, weather reports and GIS reports. Vehicles may travel onroad, off-road, or on very-rough terrain. Weather affects imaging sensors and can be generally characterized as clear or cloudy.

E.10.2 Task

Considering the domain knowledge presented below, develop a probabilistic ontology for military vehicles that will infer vehicle type (wheeled or tracked) from the MTI and imaging sensor reports, weather reports, and GIS reports. Data to complete the model is omitted here, but is included in Appendix C.

E.11 Terrorist Crewmember Problem

E.11.1 Background

Crewmembers of merchant vessels are regularly multinational and transient. This is one possible way that terrorists or terror organizations can smuggle personnel or material into target countries. Using information about an individual crewmember's relations, influences and group associations may provide insight into the likelihood of that sailor being involved in terrorism. While some affiliations may increase the likelihood that an individual may join a terrorist group and attempt access to a target country via merchant ship, there is always the uncertainty that comes from the human condition. This uncertainty associated with the multitude of factors affecting the character's context must be captured conditionally.

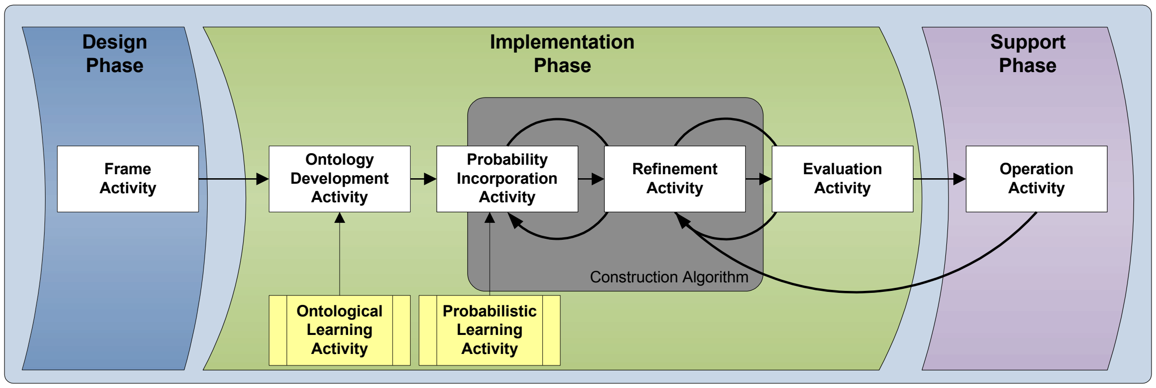
E.11.2 Task

Considering the domain knowledge presented below, develop a probabilistic ontology that may be used to assess the likelihood that that an individual merchant ship crewmember is involved in terrorism given his relationships, influences and group associations. Data to complete the model is omitted here, but is included in Appendix D.

E.12 Probabilistic Ontology Development Methodology (PODM)

E.12.1 PODM Overview

Participants were given the PODM detailed in Sections 4.3. PODM activities span the phases of the Systems Development Life Cycle (SDLC), and are grounded in model-based systems engineering (MBSE) principles. It was assumed that development of the example problems could be completed in a single spiral of the process. The activities of the PODM are illustrated in the figure, below. Completion of these activities establishes a framed solution to a specific decision problem grounded in an inclusive ontology representing its entities and incorporation of probability to represent uncertainty.



E.13 Exit Survey

Dr. K. B. Laskey / R. J. Haberlin

Participant E-mail:________________________

Dear Participant: Thank you for taking time to fill out this confidential questionnaire thoughtfully. The information you provide will be solely used by the Investigator to assess the future development of Probabilistic Ontologies .

Please rate the following on a scale of 5 (strongly agree) to 1 (strongly disagree). NA = Not Applicable.

Protégé:

Ease of Installation

Completeness of Documentation

Organization of Documentation

UnBBayes:

Ease of Installation

Completeness of Documentation

Organization of Documentation

Project:

I find that this project stimulated my interest in learning about this subject after this task

Help Desk/Tutorials:

How many queries did you submit to the help desk?

I found the help desk responses useful

This project would have been impossible without the help desk

The tutorials helped me understand the material

The tutorials helped me understand PO development

This project would have been impossible without the tutorials

Probabilistic Ontology Development:

I found the tutorials adequate preparation for the PO development tasks

The first PO development tasks helped me complete the second PO development task

A detailed methodology would help in PO development

APPENDIX F: HUMAN SUBJECT REVIEW BOARD SUBMISSION FOR A CASE STUDY SUPPORTING "PROBABILISTIC ONTOLOGY REFERENCE ARCHITECTURE AND DEVELOPMENT METHODOLOGY"

F.1 Case Study Instrument

Supporting "Probabilistic Ontology Reference Architecture and Development Methodology"

F.1.1 Aims and Purposes

The artificial intelligence (AI) community is lacking a comprehensive methodology for the development, implementation, and evaluation of probabilistic ontologies. While it is recognized that ontology use is on the rise, and a means to incorporate uncertainty is a necessity, little has been done to establish a methodology for production of *probabilistic* ontologies. This case study introduces a model-based systems Engineering (MBSE) approach to probabilistic ontology development that will formalize the process. The overarching purpose of this study is twofold:

3. Evaluate the effectiveness of the new methodology
4. Evaluate the teachability of the methodology

A group of Systems Engineering and Operations Research (SEOR) and Computer Science (CS) graduate students enrolled in the Volgenau School of Engineering at George Mason University will be the test population for the analysis. Those who choose to participate in the study will be given a comprehensive statement of work (SOW). A brief demographic survey offered at the commencement of the study will capture student personal information. At the conclusion of the project, each participant will complete a post-project survey to capture individual recommendations and comments about the methodology. Additional data available from the study will be the separate probabilistic ontologies created by the participants, and the help desk information accumulated in a database grouped by category of query (technical, knowledge, software, etc.).

The case study team will consist of Dr. Kathryn Laskey in the role of Evaluator and Richard Haberlin in the role of Investigator. The primary mission of the Evaluator is to provide oversight on the methodology delivery to ensure a positive academic experience is afforded the participants and the interests of the university are maintained. The Investigator will execute the case study protocol described below, including compilation of all documentation, delivery of the methodology and materials, manning the help desk, and compilation/analysis of data produced by the study. He will maintain regular correspondence with the Instructor to ensure she is cognizant of all participant issues/concerns related to the project and progress of the study.

F.1.2 Characteristics of the Intended Sample

The Probabilistic Ontology Development Methodology is primarily intended for utilization by academic and professional users with baseline expertise in the area of applicability. It is therefore assumed that those participating in the study will have knowledge and interest in both probabilistic ontology development and its application. The Evaluator will gauge the background knowledge available by each of the volunteers and the likelihood of success within the semester timeline.

F.1.2.1 Number

The maximum number of participants in the study is limited by the funds available and will be capped at approximately four.

F.1.2.2 Age

All participants will be adults enrolled in advanced graduate education in the Volgenau School of Engineering at George Mason University, and therefore assumed to be in the range of age from 22 to 60. Age data will be collected in the demographics survey to capture age-related issues with technology or experience.

F.1.2.3 Sex

Gender is unrestricted and will not be captured in population demographic statistics. This study is not interested in evaluating the learning capacity of different genders, but seeks only to explain significant departures from a baseline of knowledge and experience which may be explained by language barriers, experience, or age.

F.1.2.4 Ethnicity

Ethnicity is anticipated to vary with the student population. Ethnicity demographics will be captured to identify possible effects caused by language barrier on implementing the methodology.

F.1.2.5 Health

Student health will not be affected by this study. It is assumed that all subjects in the population are able to use a computer for research and model development.

F.1.2.6 Education

All participants will have college undergraduate degrees and will be working toward a masters or doctorate in systems engineering, operations research, computer science, or information technology areas of study. It is therefore assumed that all students are motivated to perform well on the assigned SOW.

F.1.3 Criteria for Inclusion or Exclusion

There are no restrictions on who may participate in the study other than a basic understanding of the subject material that will facilitate comprehension. A Help Desk Database will be compiled for the purpose of tracking type of error and solution. It will not be available during the study, but will be published after analysis of the case study data is complete for future use.

F.1.4 Relationship to Participants

None of the students are related to the Evaluator or Investigator of the study.

F.2 Protocol

F.2.1 Benefit to Participants

Participants will gain understanding of probabilistic ontology development through a concise methodology, where one currently does not exist. Tools for development are in their infancy, and navigating their interaction is unclear and often requires multiple sub-steps. The participating group will have first access to cutting-edge techniques and tutorials for production of knowledge-engineered ontological systems. The study will assess the usability and teachability of the proposed Probabilistic

Ontology Development Methodology in the academic setting. Further, each participant will receive a $400 payment for successful completion of the SOW.

F.2.2 Identification of Participants

Those who choose to participate will sign a *Student Informed Consent Form* and provide demographic information. No minors will participate in the study.

F.2.3 Consent Procedures

Informed consent will be obtained from each participant by the Investigator, under the oversight of the Evaluator. Each consent form will be scanned by the Investigator and a digital pdf copy will be sent to the participant using their George Mason University email account. The consent form is included in the SOW.

F.2.4 Compensation

Participants will receive $400 compensation for participation and successful completion of the entire SOW.

F.2.5 Design & Methodology

As introduced above, this case study will provide participants with a methodology that may assist development of probabilistic ontologies. Explicit details of requirements are included in the SOW.

The *Entry Survey* (Appendix E.7) is designed to capture basic demographic information and will be administered to participant during the introduction.

Participants will still be required to follow the SOW. Issues and queries relating to the software or the methodology may be submitted via the help desk. The Investigator will respond to these queries within 24 hours of receipt and collect statistics in the Help Desk Database.

Upon completion of the project, participants will complete the *Exit Survey* (Appendix E.13) to capture issues and recommendations relating to the methodology. Overall, it is expected that the two surveys will take less than an hour for participants to complete.

F.2.6 Confidentiality

The preferred method of communication between participants and the Investigator is through the help desk so that query data can be captured. All queries will be associated with a random number assigned by the Investigator. Occasional correspondence between participants and the Investigator may be required. These emails will be stored in a project email database maintained by the Investigator.

There will be no physical, psychological, social, or legal risks to the participants associated with use of the methodology, nor will participants be recorded in any manner. Use of the methodology and access to the help desk is designed to be straightforward without deception. Any confusion experienced by participants is unintentional and will be corrected by the Evaluator or Investigator at the earliest opportunity.

F.2.7 Data Collection Instruments

Several collection instruments will be used to collect data for the case study. Draft versions of these documents are included as appendices and briefly described below.

G. *Entry Survey*. This questionnaire will be distributed to participants to capture basic information regarding education, ethnicity and age.

H. *Informed Consent Form*. This form will be signed by all participants acknowledging participation and that they will receive $400 compensation for successful completion of the SOW. A digital copy will be given to each participant by the Investigator.

I. *Exit Survey.* This questionnaire will be distributed to participants at the completion of the study to capture issues and recommendations about the proposed methodology and the overall project experience.

J. *Help Desk Submission Page.* This stylized example represents the front page of a web-enabled help desk that will capture queries and allow collection of data in a study database. Participants will submit software and technical queries to the Investigator using the help desk.

K. *Tutorials Describing Probabilistic Ontology Development.* This tutorial will generally follow a Probabilistic Ontology Development Methodology (PODM) described in the SOW.

L. *Probabilistic Ontology Development Methodology.* Participants will be provided with this methodology which must be followed to complete the project. Issues and concerns with the methodology should be submitted to the Investigator via the help desk.

F.2.8 Cooperating Organizations

This case study will be conducted completely under the cognizance of George Mason University staff and students. No outside organizations will have access to the participants or collected data.

REFERENCES

[1] Khalid Albarrak, An Extensible Framework for Generating Ontology from Various Data Models, May 2013, PhD Dissertation.

[2] Scott W. Ambler. (2012, November) AgileModeling. [Online]. http://www.agilemodeling.com/artifacts/classDiagram.htm

[3] Grigoris Antoniou and Frank Van Harmelen, "Web Ontology Language: OWL," in *Handbook on Ontologies in Information Systems*.: Springer-Verlag, 2003.

[4] Jr., James E. Armstrong, "Issue Formulation," in *Handbook of Systems Engineering and Management*. Hoboken: John Wiley & Sons, 2009, pp. 1027-1089.

[5] Victor Asal, C. Christine Fair, and Stephen Shellman, "Consenting to a Child's Decision to Join a Jihad: Insights from a Survey of Militant Families in Pakistan," *Studies in Conflict & Terrorism*, vol. 31, pp. 973-994, 2008.

[6] Ricardo Balduino. (2012, December) Eclipse Process Framework Project. [Online]. http://www.eclipse.org/proposals/beacon/Basic%20Unified%20Process.pdf

[7] Amandine Bellenger and Sylvain Gatepaille, "Uncertainty in Ontologies: Dempster-Shafer Theory for Data Fusion Applications," in *Proceedings of the Workshop on Theory of Belief Functions*, Brest, 2010.

[8] M. K. Bergman. (2011, May) AI3 Adaptive Information Innovation Adaptive Infrastructure. [Online]. http://www.mkbergman.com/906/a-brief-survey-of-ontology-development-methodologies/

[9] Time Berners-Lee. (1988, September) Semantic Web Roadmap. [Online]. http://www.w3.org/DesignIssues/Semantic.html

[10] Chris Biermann, "Ontology Learning from Text: A Survey of Methods," *LDV Forum*, pp. 75-93, 2005.

[11] Anne-Claire Boury-Brisset, "Ontological Approach to Military Knowledge Modeling and Management," in *Proceedings of the Symposium on Military Data*

and Information Fusion, Prague, 2003, pp. 1-13.

[12] Anne-Claire Boury-Brisset, "Ontology-based Approach for Information Fusion," in *Proceedings of the 6th International Conference on Information Fusion*, Cairns, 2003, pp. 522-529.

[13] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth, "A Survey of First-Order Probabilistic Models," *Innocations in Bayesian Networks*, pp. 289-317, 2008.

[14] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth, "Lifted First-Order Probabilistic Inference," in *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT Press, 2007, ch. 15, pp. 433-452.

[15] J. S. Breese, "Construction of belief and decision networks," *Computational Intelligence*, vol. 8, pp. 624-647, 1991.

[16] Sabrina Bruaux, Gilles Kassel, and Gilles Morel, "An Ontological approach to the construction of problem-solving models," Amiens, 2005.

[17] Dennis M. Buede, *The Engineering Design of Systems: Models and Methods*. New York: John Wiley & Sons, 2000.

[18] Paul Buitelaar and Bernardo Magnini, "Ontology Learning from Text: An Overview," in *Ontology Learning from Text: Methods, Applications and Evaluation*.: IOS Press, 2005, pp. 3-12.

[19] Rommel Novaes Carvalho. (2011, June) PhD George Mason University. [Online]. http://hdl.handle.net/1920/6616

[20] Rommel Novaes Carvalho, Richard J. Haberlin, Paulo Cesar G. da Costa, Kathryn B. Laskey, and KC Chang, "Modeling a Probabilistic Ontology for Maritime Domain Awareness," in *Proceedings of the 14th International Conference on Information Fusion*, Chicago, 2011, pp. 1-8.

[21] Rommel Novaes Carvalho, Kathryn B. Laskey, and Paulo Cesar G. Da Costa, "PR-OWL 2.0 - Bridging the Gap to OWL Semantics," in *Proceedings of the 9th International Sematic Web Conference*, Bonn, 2010, pp. 1-15.

[22] Rommel Novaes Carvalho et al., "Probabilistic Ontology and Knowledge Fusion for Procurement Fraud Detection in Brazil," in *Proceedings of the International Semantic Web Conference* , Washington D.C., 2009, pp. 3-14.

[23] Rommel N. Carvalho, L. L. Santos, M. Ladeira, and P. C.G. Costa, "A GUI tool for plausible reasoning in the semantic web using MEBN," *IEEE Computer Society*, pp. 381-386, 2007.

[24] Alexander Garcia Castro et al., "The use of concept maps during knowledge elicitation in ontology development processes-the nutrigenomics use case," *BMC Bioinformatics*, pp. 1-14, 2006.

[25] Hsinchun Chen et al., "Uncovering the Dark Web: A Case Study of Jihad on the Web," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 8, pp. 1347-1359, 2008.

[26] Robert T. Clemen, *Making Hard Decisions: An Introduction to Decision Analysis*. Pacific Grove: Duxbury Press, 1996.

[27] T. R. Coffman and S. E. Marcus, "Pattern Classification in Social Network Analysis: A Case Study," in *Proceedings of the 2004 IEEE Aerospace Conference*, vol. 5, Big Sky, 2004, pp. 3162-3175.

[28] Paulo Cesar de Costa, *Bayesian Semantics for the Semantic Web*. Fairfax: George Mason University, 2005.

[29] Paulo Cesar G. da Costa. (2005, July) PhD George Mason Univeristy. [Online]. http://hdl.handle.net/1920/455

[30] Paulo Cesar G. da Costa, K.C. Chang, Kathryn B. Laskey, and Rommel Novaes Carvalho, "A Multidisciplinary Approach to High Level Fusion in Predictive Situational Awareness," in *Proceedings of the 11th International Conference of the Society of Information Fusion*, Seattle, 2009.

[31] Paulo Cesar G. da Costa, Herencia-Zapana Hebner, and Kathryn B. Laskey, "Uncertainty Representation and Reasoning for Combat Models," in *Engineering Principles of Combat Modeling and Distributed Simulation*.: John Wiley & Sons, 2012.

[32] Paulo Cesar G. da Costa et al., "A First-Order Bayesian Tool for Probabilistic Ontologies," in *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, Coconut Grove, 2008, pp. 1-6.

[33] Paulo Cesar G. da Costa and Kathryn B. Laskey, "PR-OWL: A Framework for Bayesian Ontologies," in *Proceedings of the 4th International Conference on Formal Ontology in Information Systems*, Baltimore, 2006, pp. 237-249.

[34] Paulo Cesar G. da Costa, Kathryn B. Laskey, and K.C. Chang, "PROGNOS: Applying Probabilistic Ontologies to Distributed Predictive Situation Assessment in Naval Operations," in *Proceedings of the 14th International Command and Control Research and Technology Symposium*, Washington D.C., 2009.

[35] Paulo Cesar G. da Costa, Kathryn B. Laskey, Kenneth J. Laskey, and Edward J. Wright, "Probabilistic Ontologies: The Next Step for Net-Centric Operations," in *Proceedings of the 12th International Command and Control Research and Technology Symposium*, Newport, 2007, pp. 1-18.

[36] Paulo Cesar G. da Costa, Kathryn B. Laskey, and Thomas Lukasiewicz, "Uncertainty Representation and Reasoning in the Semantic Web," in *Web Technologies: Concepts, Methodologies, Tools, and Applications*. Victoria: IGI Global, 2009, pp. 1852-1877.

[37] James Cussens, "Logic-based Formalisms for Statistical Relational Learning," in *Introduction to Statistical Relational Learning*. Cambridge: MIT Press, 2007, ch. 9, pp. 269-290.

[38] Cycorp. (2013, June) CycL: The Cyc Knowledge Representation Language. [Online]. http://www.cyc.com/cyc/cycl

[39] Adnan Darwiche, *Modeling and Reasoning with Bayesian Networks*. Cambridge: Cambridge Univeristy Press, 2009.

[40] Dictionary.com LLC. (2013) Dictionary.com. [Online]. http://dictionary.reference.com/

[41] Dictionary.com, LLC. (2013, June) "axiom" in Dictionary.com Unabridged. [Online]. http://dictionary.reference.com/browse/axiom?s=t

[42] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy, "Ontology Matching: A Machine Learning Approach," in *Handbook on Ontologies*. Berlin: Springer-Verlag, 2009, pp. 385-404.

[43] Anhai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy, "Ontology Matching: A Machine Learning Approach," in *Handbook on Ontologies in Information Systems*.: Springer, 2003, pp. 397-416.

[44] Matthew J. Dombroski and Kathleen M. Carley, "NETEST: Estimating a Terrorist Network's Structure," *Computational & Mathematical Organization Theory*, vol. 8, pp. 235-241, 2002.

[45] Pedro Domingos and Matthew Richardson, "Markov Logic: A Unifying Framework for Statistical Relational Learning," in *Introduction to Statistical Relational Learning*. Cambridge: The MIT Press, 2007, pp. 339-371.

[46] Johan Eltes, The Reference Architecture - a foundation for successful projects, 2004.

[47] Miriam Fernandez, Ivan Cantador, and Pablo Castells, "CORE: A Tool for Collaborative Ontology Reuse and Evaluation," in *Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web*, Edinburgh, 2006, pp. 1-9.

[48] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon, "Logical Bayesian networks and their relation to other probabilistic logical models," in *Proceedings of the 15th International Conference on Inductive Logic Programming*, Bonn, 2005, pp. 121-135.

[49] Sanford Friedenthal, Alan Moore, and Rick Steiner, *A Practical Guide to SysML: The Systems Modeling Language*. Amsterdam: Elsevier, 2008.

[50] Sanford Friedenthal, Alan Moore, and Rick Steiner, *OMG Systems Modeling Language Tutorial*.: Object Management Group, 2008.

[51] Steven Galovich, *Doing Mathematics: An Introduction to Proofs and Problem Solving*. Fort Worth: Saunders College, 1993.

[52] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Ben Taskar, "Probabilistic Relational Models," in *Introduction to Statistical Relational Learning*. Cambridge: The MIT Press, 2007, pp. 129-174.

[53] S. Glessner and D. Koller, "Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases," *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pp. 217-226, 1995.

[54] R. P. Goldman and E. Cherniak, "A language for construction of belief networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 3, pp. 196-208, 1993.

[55] Harry Goldstein, "Modeling Terrorists: New Simulators Could Help Intelligence Analysts Think Like the Enemy," *IEEE Spectrum*, pp. 26-34, September 2006.

[56] Asuncion Gomez-Perez, Fernandez-Lopez Mariano, and Oscar Corcho, *Ontological Engineering with Examples from the Areas of Knowledge*

Management, e-Commerce and the Semantic Web. London: Springer-Verlag, 2010.

[57] Jeffrey O. Grady, *System Requirements Analysis*. New York: McGraw-Hill, Inc., 1993.

[58] John M. Green, "Establishing System Measures of Effectiveness," in *Proceedings of the 2nd Biennial National Forum on Weapon System Effectiveness*, Laurel, 2001, pp. 1-5.

[59] Alan Grosskurth and Michael W. Godfrey, "A Reference Architecture for Web Browsers," in *International Conference on Software Maintenance*, Budapest, 2005.

[60] Thomas R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies*, pp. 907-928, 1995.

[61] Richard J. Haberlin, Paulo Cesar G. da Costa, and Kathryn B. Laskey, "An Ontology for Hypothesis Management in the Maritime Domain," in *Proceedings of the 16th International Command and Control Research and Technology Symposium*, Fairfax, 2011, pp. 1-18.

[62] Richard J. Haberlin, Paulo Cesar G. da Costa, and Kathryn B. Laskey, "Hypothesis Management in Support of Inferential Reasoning," in *Proceedings of the 15th International Command and Control Research Symposium*, Santa Monica, 2010, pp. 1-22.

[63] Richard J. Haberlin and David A. Schum, "Establishing a Source Pedigree Through Inferential Reasoning," in *Proceedings of the 79th Military Operations Research Society Symposium*, Monterey, 2011, pp. 1-11.

[64] P. Haddaway, "Generating Bayesian networks from probability logic knowledge bases," *Uncertainty in Artificial Intelligence*, vol. 10, pp. 262-269, 1994.

[65] Daniel T. Halstead and Kenneth D. Forbus, "Transforming between propositions and features: Bridging the gap," in *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, 2005, pp. 777-782.

[66] Ahmed E. Hassan and Richard C. Holt, "A Reference Architecture for Web Servers," in *Proceedings of the Seventh Working Conference on Reverse Engineering*, Brisbane, 2000.

[67] Reid Hastie and Robyn M. Dawes, *Rational Choice in an Uncertain World: The*

Psychology of Judgment and Decision Making. Los Angeles: Sage, 2010.

[68] David Heckerman, "A Tutorial on Learning with Bayesian Networks," Redmond, 1996.

[69] David Heckerman, Chris Meek, and Daphne Koller, "Probabilistic Entity-Relationship Models, PRMs, and Plate Models," in *Introduction to Statistical Relational Learning*. Cambridge: The MIT Press, 2007, pp. 201-238.

[70] Peter D. Hoff, *A First Course in Bayesian Statistical Methods*. Dordrecht: Springer, 2009.

[71] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, and Chris Wroe, *A Practical Guide to Building OWL Ontologies Using the Protege-OWL Plugin and CO-ODE Tools*. Manchester: The University of Manchester Press, 2004.

[72] M. Horsch and D. Poole, "A dynamic approach to probabilistic inference using Bayesian networks," in *Proceedings of the 6th Conference of Uncertainty in Artificial Intelligence*, San Francisco, 1990, pp. 155-161.

[73] IEEE, *IEEE Standard for Developing Software Life Cycle Processes*. New York: IEEE Computer Society, 1996.

[74] IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. New York: IEEE Computer Society, 1990.

[75] IFIP-IFAC Task Force on Architectures for Enterprise Integration, "GERAM: Generalised Enterprise Reference Architecture and Methodology v1.6.3," 1999.

[76] INCOSE, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Seattle, 2008.

[77] INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*.: INCOSE, 2006.

[78] Institute for Formal Ontology and Medical Information Science. (2013, March) BFO: Basic Formal Ontology. [Online]. http://www.ifomis.org/bfo

[79] Institute of Cognitive Science and Technology Italian National Research Council. (2013, June) WonderWeb. [Online]. http://www.loa.istc.cnr.it/DOLCE.html

[80] International Standards Organization, "Information technology - Common Logic (CL): a framework for a family of logic-based languages," International Standards Organization, Standard ISO/IEC 24707:2007(E), 2007.

[81] M. Jaeger, "Relational Bayesian netwroks," in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, Providence, 1997, pp. 266-273.

[82] Leonid Kalinichencko, Michele Missikoff, Federica Schiappelli, and Nikolay Skvortsov, "Ontological Modeling," in *Proceedings of the 5th Russian Conference on Digital Libraries*, St. Petersburg, 2003, pp. 1-7.

[83] C. Maria Keet, "Dependencies between Ontology Design Parameters," *International Journal of Metadata, Semantics and Ontologies*, pp. 265-284, 2010.

[84] K. Kersting and L. DeRaedt, "Bayesian logic programs," in *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, London, 2000, pp. 138-155.

[85] Minsoo Kim and Minkoo Kim, "Developing Protege Plug-in: OWL Ontology Visualization using Social Network," *Journal of Information Processing Systems*, pp. 61-66, 2008.

[86] Rajiv Kishore, Raj Sharman, and Ram Ramesh, "Computational Ontologies and Information Systems: Foundations," *Communications of the Association for Information Systems*, vol. 14, pp. 158-183, 2004.

[87] David Koelle et al., "Applications of Bayesian Belief Networks in Social Network Analysis," in *Proceedings of the 4th Bayesian Modeling Applications Workshop*, Cambridge, 2006.

[88] D. Koller and A. Pfeffer, "Learning probabilities for noisy first-order rules," *International Joint Conference on Artificial Intelligence*, pp. 1316-1323, 1997.

[89] Kevin B. Korb and Ann E. Nicholson, *Bayesian Artificial Intelligence*. Boca Raton: CRC Press, 2011.

[90] Gerald Kotonya and Ian Sommerville, *Requirements Engineering Processes and Techniques*. Chichester: John Wiley & Sons, 1998.

[91] Pawel Kozak and Karsten Tolle, "Analyzing Web Profiles using Probabilistic Ontologies," in *Proceedings of the 3rd International Conference on Web Science*, Koblenz, 2011.

[92] Valdis E. Krebs, "Mapping Networks of Terrorist Cells," *Connections*, vol. 24, pp. 43-52, 2001.

[93] Heather Kreger, Vince Brunssen, Robert Sawyer, Ali Arsanjani, and Rob High. (2012, Jan) IBM Developer Works. [Online]. http://www.ibm.com/developerworks/webservices/library/ws-soa-ref-arch/

[94] Serguei Krivov, Fernando Villa, Richard Williams, and Xindong Wu, "On Visualization of OWL Ontologies," in *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*.: Springer, 2007, pp. 205-219.

[95] Philippe Kruchten, *The Rational Unified Process: An Introduction*. Upper Saddle River: Addison-Wesley, 2004.

[96] Kathryn B. Laskey, "MEBN: A Language for First-Order Bayesian Knowledge Bases," *Artificial Intelligence*, pp. 140-178, 2008.

[97] Kathryn B. Laskey, Paulo Cesar G. da Costa, and Terry Janssen, "Probabilistic Ontologies for Knowledge Fusion," in *Proceedings of the 11th International Conference on Information Fusion*, Cologne, 2008, pp. 1-8.

[98] Kathryn B. Laskey, Paulo Cesar G. da Costa, and Terry Janssen, "Probabilistic Ontologies for Multi-INT Fusion," in *Proceedings of the 2010 Conference on Ontologies and Semantic Technologies for Intelligence*, Fairfax, 2010, pp. 147-161.

[99] Kathryn B. Laskey and Suzanne M. Mahoney, "Network Engineering for Agile Belief Network Models," *IEEE Transactions on Knowledge and Data Engineering*, pp. 487-498, 2000.

[100] Kathryn B. Laskey, Suzanne M. Mahoney, and Ed Wright, "Hypothesis Management in Situation-Specific Network Construction," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, Seattle, 2001, pp. 301-309.

[101] Alexander Levis, "System Architectures," in *Handbook of Systems Engineering and Management*.: John Wiley & Sons, 2009, pp. 479-506.

[102] Man Li, Xiao-Yong Du, and Shan Wang, "Learning Ontology from Relational Database," in *Proceedings of the 4th International Conference on Machine Learning and Cybernetics*, Guangzhou, 2005, pp. 3410-3415.

[103] Alexander Maedche and Steffan Staab, "Ontology Learning for the Semantic Web,"

IEEE Intelligent Systems, pp. 72-79, 2001.

[104] George M. Marakas, *Decision Support Systems in the 21st Century*. Upper Saddle River: Prentice Hall, 2003.

[105] Kneale T. Marshall and Robert M. Oliver, *Decision Making and Forecasting*. New York: McGraw-Hill, Inc., 1995.

[106] Massimiliano Dal Mas. (2010, September) Cornell University Library. [Online]. http://arxiv.org/abs/1009.2084

[107] Christopher J. Matheus, David Tribble, Mieczyslaw M. Kokar, Marion G. Ceruti, and Scott C. McGirr, "Towards a Formal Pedigree Ontology for Level-One Sensor Fusion," in *Proceedings of the 10th International Command & Control Research and Technology Symposium*, McLean, 2006, pp. 1-16.

[108] Shou Matsumoto et al., "UnBBayes: a Java Framework for Probabilistic Models in AI," in *Java in Academia and Research*, Ke Cai, Ed. Annerley, Australia: iConcept Press, Ltd., 2011, ch. Chapter 9, p. 34.

[109] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling, "Lifted Probabilistic Inference with Counting Formulas," in *Proceedings of the 23rd National Conference on Artificial Intelligence*, Chicago, 2008, pp. 1062-1068.

[110] Riichiro Mizogushi and Mitsuru Ikeda, "Toward Ontology Engineering," Osaka, 1996.

[111] James Moody, "Fighting a Hyrdra: A Note on the Network Embeddedness of the War on Terror," *Structure and Dynamics*, vol. 1, August 2006.

[112] Il-Chul Moon and Kathleen M. Carley, "Modeling and Simulating Terrorist Networks in Social and Geospatial Dimensions," *IEEE Intelligent Systems*, vol. 22, pp. 40-49, 2007.

[113] Gerrit Muller, "A Reference Architecture Primer," Buskerud, 2012.

[114] J. Neville, Statistical Models and analysis techniques for learning in relational data, 2006, PhD Dissertation, University of Massachusetts Amherst.

[115] Norsys Software Corp. (2013, June) Netica. software. [Online]. http://www.norsys.com/index.html

[116] Chris Nowak, "On ontologies for high-level information fusion," in *Proceedings of the 6th International Conference on Information Fusion*, Cairns, 2003, pp. 657-664.

[117] Natalia F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, 2001.

[118] OASD/NII, "DoD Reference Architecture Description," Arlington, 2010.

[119] Object Management Group, "Model Driven Architecture Guide Version 1.0.1," 2003.

[120] Object Management Group. (2003, Jun) Object Management Group Model Driven Architecture. [Online]. http://www.omg.org/cgi-bin/doc?omg/03-06-01

[121] Object Management Group. (2006, Sep) SOA Practitioner's Guide Part 2: SOA Reference Architecture. [Online]. http://soablueprint.com/yahoo_site_admin/assets/docs/SOAPGPart2.290211443.pdf

[122] Office of the Assistance Secretary of Defense for Networks and Information Integration (OASD/NII), "Reference Architecture Description," Arlington, 2010.

[123] Joseph Ogando, "Engineering's Terrorism Link?," *Design News*, vol. 63, pp. 25-28, April 2008.

[124] Cheol Young Park, Kathryn B. Laskey, Paulo C.G. Costa, and Shou Matsumoto, "Multi-Entity Bayesian Networks Learning for Hybrid Variables in Situation Awareness," in *Proceedings of the 16th International Conference on Information Fusion (submitted)*, Istanbul, 2013, pp. 1-8.

[125] Cheol Young Park, Kathryn B. Laskey, Paulo C.G.N. Costa, and Shou Matsumoto, "Multi-Entity Bayesian Networks Learning in Predictive Situation Awareness," in *Proceedings of the 18th International Command and Control Research and Technology Symposium*, Alexandria, 2013, pp. 1-19.

[126] Mary C. Parmelee, "Toward an Ontology Architecture for Cyber-Security Standards," in *Proceedings of the 5th International Conference on Semantic Technologies for Intelligence, Defense, and Security*, Fairfax, 2010, pp. 1-8.

[127] F. G. Patterson, "Systems Engineering Life Cycles: Life Cycles for Research, Development, Test, and Evaluation; Acquisition; and Planning and Marketing," in *Handbook of Systems Engineering and Management*.: John Wiley & Sons, 2009, pp. 65-115.

[128] Judea Pearl, *Causality: Models, Reasoning, and Inference*. Cambridge: Cambridge University Press, 2009.

[129] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, 1988.

[130] Avi Pfeffer, "The Design and Implementation of IBAL: A General-Purpose Probabilistic Language," in *Introduction to Statistical Relational Learning*. Cambridge: The MIT Press, 2007, pp. 399-432.

[131] D. Poole, "Probabilistic Horn abduction and Bayesian networks," *Artificial Intelligence*, vol. 64, no. 1, pp. 81-129, 1993.

[132] Jialun Qin, Yilu Xhou, Edna Reid, Guanpi Lai, and Hsinchun Chen, "Analyzing terror campaigns on the internet: Technical sophistication, content richness, and Web interactivity," *International Journal of Human-Computer Studies*, vol. 65, pp. 71-84, 2007.

[133] Hoda Rashad, Magued Osman, and Farzaneh Roudi-Fahimi, "Marriage in the Arab World," Washington D.C., 2005.

[134] Kumar Ravi and Sheopujan Singh, "Risk Prediction for Production of an Enterprise," *International Journal of Computer Applications Technology and Research*, vol. 2, no. 3, pp. 237-244, 2013.

[135] Paul Reed. (2002, Sep) IBM Developer Works. [Online]. http://www.ibm.com/developerworks/rational/library/2774.html

[136] Steve Ressler, "Social Network Analysis as an Approach to Combat Terrorism: Past, Present, and Future Research," *Homeland Security Affairs*, vol. II, no. 2, July 2006.

[137] William B. Rouse and Andrew P. Sage, *Handbook of systems engineering and management*. Hoboken: John Wiley & Sons, 2009.

[138] Marc Sageman, *Understanding Terror Networks*. Philadelphia: University of Pennsylvania Press, 2004.

[139] V Santos Costa, D. Page, M. Qazi, and J. Cussens, "Constraint logic programming for probabilistic knowledge," in *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, 2003, pp. 517-552.

[140] John W. Satzinger, Robert B. Jackson, and Stephen D. Burd, *Systems Analysis and Design in a Changing World*. Boston: Course Technology, 2004.

[141] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, and M. Wimmer, "Towards a Common Reference Architecture for Aspect-Oriented Modeling," in *Proceedings of the 8th International Workshop on Aspect-Oriented Modeling*, Bonn, 2006.

[142] David A. Schum, "A Science of Evidence: Contributions from Law and Probability," *Law, Probability and Risk*, pp. 197-231, 2009.

[143] David A. Schum, "Probabilistic Reasoning and the Science of Complexity," in *Decision Science and Technology: Reflections on the Contributions of Ward Edwards*. Boston: Kluwer-Academic, 1999, pp. 183-209.

[144] David A. Schum, *The Evidential Foundations of Probabilistic Reasoning*. Evanston: Northwestern University Press, 2001.

[145] David A. Schum and Jon R. Morris, "Assessing the Competence and Credibility of Human Sources of Intelligence Evidence: Contributions from Law and Probability," *Law, Probability and Risk*, pp. 247-274, 2007.

[146] David A. Schum, Gheorghe Tecuci, and Mihai Boicu, "Analyzing Evidence and Its Chain of Custody: A Mixed-Initiative Computational Approach," *International Journal of Intelligence and Counter Intelligence*, pp. 298-319, 2009.

[147] Scott Ambler & Associates. (2012) Agile Modeling (AM) Home Page. [Online]. http://www.agilemodeling.com/

[148] Prithviraj Sen, Amol Deshpande, and Lise Getoor, "Bisimulation-based Approximate Lifted Inference," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, Arlington, 2009, pp. 496-505.

[149] Stewart Shapiro. (2009, Nov) Classic Logic. [Online]. http://plato.stanford.edu/entries/logic-classical/

[150] Jude Shavlik and Sriraam Natarajan, "Speeding Up Inference in Markov Logic Networks by Preprocessing to Reduce the Size of the Resulting Grounded Network," in *Proceedings of the International Joint Conference in AI*, Pasadena, 2009, pp. 1-6.

[151] Barry Smith, "Beyond Concepts: Ontology as Reality Representation," in *Proceedings of the 3rd International Conference of Formal Ontology in*

Information Systems, Amsterdam, 2004, pp. 73-84.

[152] John Sowa. (2001) Ontology. [Online]. http://www.jfsowa.com/ontology/

[153] Peter Spyns, Robert Meersman, and Mustafa Jarraf, "Data modelling versus Ontology engineering," *SIGMOD Record*, vol. 31, pp. 12-17, 2002.

[154] Stanford University. (2011, January) Protege. [Online]. http://protege.stanford.edu/

[155] Umberto Straccia, "Managing Uncertainty and Vagueness in Description Logics, Logic Programs and Description Logic Programs," in *Reasoning Web*. Berlin: Springer-Verlag, 2008, pp. 54-103.

[156] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probabilistic models for relational data," in *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, Edmonton, 2002, pp. 1-8.

[157] Octavian Udrea, Deng Yu, Edward Hung, and V. S. Subrahmanian, "Probabilistic Ontologies and Relational Databases," in *Proceedings of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems*, Agia Napa, 2005, pp. 1-17.

[158] United States Air Force. (2003, February) Software Technology Support Center. [Online]. http://www.stsc.hill.af.mil/resources/tech_docs/gsam4/chap10.pdf

[159] University of Brasilia - UnB. (2010) UnBBayes. [Online]. http://unbbayes.sourceforge.net/index.html

[160] Lise Urbaczewski and Stevan Mrdalj, "A Comparison of Enterprise Architecture Frameworks," *Information Systems Journal*, pp. 18-23, 2006.

[161] W3C Incubator Group. (2008, March) Uncertainty Reasoning for the World Wide Web. [Online]. http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/

[162] Gabriel Weimann, "Virtual Terrorism: How Modern Terrorists Use the Internet," in *The Internet and Governance in Asia*. Dresden, 2007, pp. 189-216.

[163] Hui Yang and James Callan, "Human-Guided Ontology Learning," in *Proceedings of Human-Computer Interaction and Information Retrieval (HCIR)*, 2008, pp. 26-29.

[164] Hui Yang and Jamie Callan, "Human-Guided Ontology Learning," in *Proceedings of the 2nd Workship on Human-Computer Interaction and Information Retrieval*, Redmond, 2008, pp. 1-4.

[165] Robert Yin, *Case study research: design and methods*. Los Angeles: Sage Publications, 2009.

[166] Xinye Zhao, Zhongchen Fan, Shanliang Yang, and Kedi Huang, "Towards a Course of Action Probability Ontology for Logistic Supply Destruction Operations," in *Proceedings of the Asia Simulation Conference 2012*, Shanghai, 1012, pp. 1-9.

[167] G. Zhao, J. Zheng, and R. Meersman, "An Architecture Framework for Ontology Development and Deployment," in *Proceedings of the E-Society 2003 IADIS International Conference*, Lisbon, 2003.

BIOGRAPHY

Richard J. Haberlin Jr. graduated from Lely High School, Naples, Florida, in 1986. He received his Bachelor of Science from the United States Naval Academy in 1990. He received his Master of Science in Operations Research from the Naval Postgraduate School in 1997 and retired from the United States Navy after 20 years of service in 2010.