A Reference Architecture for Probabilistic Ontology Development

Richard J. Haberlin, Jr.
EMSolutions, Inc.
Arlington, Virginia
rjhaberlin@comcast.net

Paulo C. G. da Costa
Kathryn B. Laskey
Systems Engineering and Operations Research
George Mason University
Fairfax, Virginia
pcosta, klaskey @gmu.edu

Abstract - **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. 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. This paper introduces a systematic approach to probabilistic ontology development through a reference architecture which captures the evolution of a traditional ontology into a probabilistic ontology implementation for real-world problems. The Reference Architecture for Probabilistic Ontology Development catalogues and defines the processes and artifacts necessary for the development, implementation and evaluation of explicit, logical and defensible probabilistic ontologies developed for knowledge-sharing and reuse in a given domain.**

Keywords—probabilistic ontology, knowledge engineering, reference architecture

I. INTRODUCTION

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*[1]*."*

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 [1] [2]. 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 [1].

A. Background

Development of the RAPOD provides synergy of effort within the Semantic Technology (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 [3]. 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.

B. 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. The Stakeholder Decision Maker (DM), Subject-Matter Expert (SME) and Probabilistic

Ontology Developer 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.

C. *Model Implementation and Viewpoint*

The concept behind the RAPOD is to establish intellectual control of the probabilistic ontology (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 [4]. 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. 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.

II. 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 processes and artifacts 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.

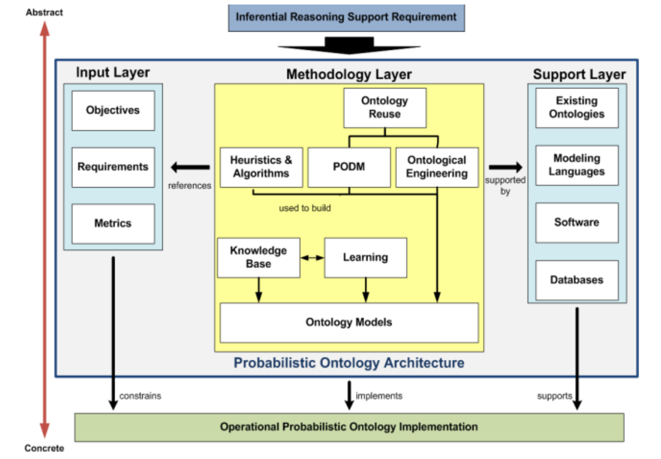

Figure 1

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. For example, a military commander may be charged with creating a decision support system that assists in the determination of an opposing force given limited sensor information. A Naval application example is given in [3]. 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.

A. *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.

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 [5]. 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 [6]:

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 [6]. 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.

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 [7] [8]. In general, requirements elicitation approaches may be categorized as structured or unstructured [8] 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 [7]. Grady categorizes three strategies for requirements analysis: structured analysis, cloning, and freestyle [8]. 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. There is inefficiency and risk involved in the unstructured methods as there is nothing to prevent duplicative work, incompleteness, conflicts and misdirection.

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

<u>Measures of Effectiveness</u>. A measure of system performance within its intended environment (e.g. overall system effectiveness).

<u>Measures of Performance</u>. A measure of one attribute of system behavior derived from its parameters (e.g. probability of correct identification).

<u>Parameters</u>. Properties or characteristics whose values determine system behavior (e.g. error rate).

Armstrong [6] 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 [9]. In all cases, appropriate metrics depend on the system under development and its ultimate purpose (objectives).

B. *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.

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 shown in the below activity diagram (Figure 2) and further detailed in [3]. These activities fit well within both Waterfall and Spiral Development Life Cycle processes where in Spiral Development iteration is explicitly anticipated.

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.

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 [10]. 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; but ultimately each

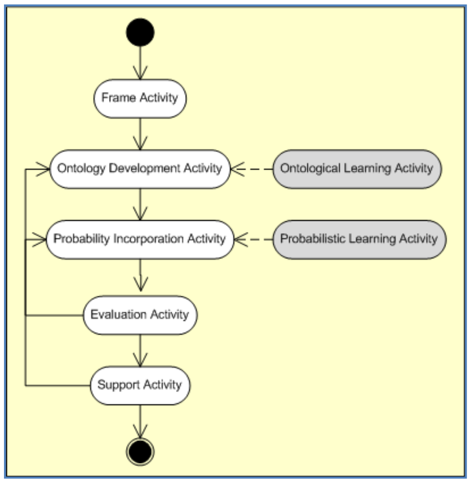
Frame Activity
Ontology Development Activity
Ontological Learning Activity
Probability Incorporation Activity
Probabilistic Learning Activity
Evaluation Activity
Support Activity

Figure 2

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 [11].

3) Ontology Reuse

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 [10]. 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 [10][11][12][13][14]. The literature liberally addresses the concept of ontology reuse, but there is little guidance offered for selection of 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.

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 $\infty \ldots 1$ 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 [3].

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.

a) 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 [15]:

- 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 [16]. 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, and taxonomies. 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, and the Semantic Web. Albarrak developed an extensible framework for generating ontologies from Relational Database (RDB) and Object-Relational Database (ORDB) data models [17]. Li et al. introduce a novel set of 12 learning rules that build a complete OWL ontology of classes, properties, characteristics, cardinality and instances [18]. 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 [19] [20]. 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.

b) 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 [21]:

– Extracting generic hypothesis evidence-relationships from records of experience, and

– 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:

<u>Maximum Likelihood</u> [22][23] – Parameters are estimated from a set of empirical data using a likelihood weighting algorithm.

<u>Bayesian Learning</u> [22][23] – Prior knowledge about parameters is encoded and data is treated as evidence to reduce the learning process to calculation of posterior distributions.

Learning is segregated into the categories of structure learning and parameter estimation [23][24]. In parameter estimation, the dependency structure of the probabilistic representation is known. The learning task is to define the parameters of the Local Probability Distributions (LPDs). The goal of structure learning is to extract the structure of the probabilistic representation from the dataset.

Learning a Probabilistic Relational Model (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 Conditional Probability Disributions (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 [23].

Recall that the structure of a Markov Logic Network (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 [25].

Multi-Entity Bayesian Network (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 [26]. More recently, Park et al. has extended the MEBN learning algorithm to include both discrete and continuous random variables [27].

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 [28]*."*

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 [11] [13]. 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.

a) 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.

b) 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.

C. 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.

1) Existing Ontologies

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 below.

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.

a) 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 [5].
- Unified Process (UP) – UP is an iterative, comprehensive development approach adapted to object oriented models, tools and techniques [29]. 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.

b) 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 [29].
- Systems Modeling Language (SysML) – SysML extends UML language with semantic foundation for representing requirements, behavior, structure, and properties of systems and components [30] [31].

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.

c) 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 [32]:

- 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 [10]:

- 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 [33].

- 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 [34].

- 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 [35].

- Basic Formal Ontology (BFO) – An upper-level ontological framework used in support of domain ontologies developed for scientific research [36].

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 [13]. Of the above ontology languages, only OWL allows expression of probabilistic information along with an ontology through the PR-OWL extension.

d) Probabilistic Ontology Representation

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 [13]. 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 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 [37]. 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.

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. The PO Developer selects software tools with the correct fidelity to represent relevant viewpoints and provide the desired communication and inferential reasoning representation. A combination of these tools gives the PO Developer flexibility in creating necessary views for communication, as well as operational ontology and probabilistic ontology models.

a) 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. Tools such as Microsoft Visio and MagicDraw assist in visual representation to simplify complex concepts.

b) 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.

c) 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).

III. Summary

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.

References

[1] Office of the Assistance Secretary of Defense for Networks and Information Integration (OASD/NII), "Reference Architecture Description," Arlington, 2010.

[2] 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/.

[3] Richard J. Haberlin, *Probabilistic Ontology Reference Architecture and Design Methodology*, PhD George Mason University, 2013.

[4] Philippe Kruchten, *The Rational Unified Process: An Introduction*. Upper Saddle River: Addison-Wesley, 2004.

[5] Dennis M. Buede, *The Engineering Design of Systems: Models and Methods*. New York: John Wiley & Sons, 2000.

[6] James E. Armstrong, "Issue Formulation," in *Handbook of Systems Engineering and Management*. Hoboken: John Wiley & Sons, 2009, pp. 1027-1089.

[7] Gerald Kotonya and Ian Sommerville, *Requirements Engineering Processes and Techniques*. Chichester: John Wiley & Sons, 1998.

[8] Jeffrey O. Grady, *System Requirements Analysis*. New York: McGraw-Hill, Inc., 1993.

[9] 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.

[10] 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.

[11] Thomas R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies*, pp. 907-928, 1995.

[12] Michael K. Bergman, "A Brief Survey of Ontology Development Methodologies," 2011, [Online]. http://www.mkbergman.com/906/a-brief-survey-of-ontology-development-methodologies/

[13] Paulo Cesar G. da Costa. *Bayesian Semantics for the Semantic Web*, PhD George Mason Univeristy, 2005. [Online]. http://hdl.handle.net/1920/455 .

[14] Maria C. Keet, "Dependencies between Ontology Design Parameters," *International Journal of Metadata, Semantics and Ontologies*, pp. 265-284, 2010.

[15] 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.

[16] John Sowa. (2001) Ontology. [Online]. http://www.jfsowa.com/ontology/ .

[17] Khalid Albarrak, *An Extensible Framework for Generating Ontology from Various Data Models*, May 2013, PhD Dissertation.

[18] 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.

[19] 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.

[20] 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.

[21] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, 1988.

[22] Adnan Darwiche, *Modeling and Reasoning with Bayesian Networks*. Cambridge: Cambridge Univeristy Press, 2009.

[23] 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.

[24] James Cussens, "Logic-based Formalisms for Statistical Relational Learning," in *Introduction to Statistical Relational Learning*. Cambridge: MIT Press, 2007, ch. 9, pp. 269-290.

[25] 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.

[26] 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.

[27] 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.

[28] George M. Marakas, *Decision Support Systems in the 21st Century*. Upper Saddle River: Prentice Hall, 2003.

[29] John W. Satzinger, Robert B. Jackson, and Stephen D. Burd, *Systems Analysis and Design in a Changing World*. Boston: Course Technology, 2004.

[30] Sanford Friedenthal, Alan Moore, and Rick Steiner, *A Practical Guide to SysML: The Systems Modeling Language*. Amsterdam: Elsevier, 2008.

[31] Sanford Friedenthal, Alan Moore, and Rick Steiner, *OMG Systems Modeling Language Tutorial*.: Object Management Group, 2008.

[32] Grigoris Antoniou and Frank Van Harmelen, "Web Ontology Language: OWL," in *Handbook on Ontologies in Information Systems*.: Springer-Verlag, 2003.

[33] Cycorp. (2013, June) CycL: The Cyc Knowledge Representation Language. [Online]. http://www.cyc.com/cyc/cycl .

[34] 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.

[35] Institute of Cognitive Science and Technology Italian National Research Council. (2013, June) WonderWeb. [Online]. http://www.loa.istc.cnr.it/DOLCE.html .

[36] Institute for Formal Ontology and Medical Information Science. (2013, March) BFO: Basic Formal Ontology. [Online]. http://www.ifomis.org/bfo .

[37] 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.