19th International Command and Control Research and Technology Symposium

"C2 Agility: Lessons Learned from Research and Operations"

A Probabilistic Ontology Development Methodology

Topic 3: Data, Information,and Knowledge
Topic 5: Modeling and Simulation
Topic 7: Autonomy

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

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

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2014		**00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Probabilistic Ontology Development Methodology	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **EMSolutions, Inc,1401 South Clark Street Suite 200,Arlington,VA,22202**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 18th International Command & Control Research & Technology Symposium (ICCRTS) held 16-19 June, 2014 in Alexandria, VA. U.S. Government or Federal Rights License

14. ABSTRACT

The use of ontologies is on the rise, as they facilitate interoperability and provide support for automation. Today, ontologies are popular 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. We introduce a systematic approach to probabilistic ontology development which focuses on evolving a traditional ontology from conceptualization to probabilistic ontology implementation for real-world problems. The Probabilistic Ontology Development Methodology is an efficient, teachable, and repeatable technique for the development, implementation and evaluation of explicit, logical and defensible probabilistic ontologies developed for knowledge-sharing and reuse in a given domain.

15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**36**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

A Probabilistic Ontology Development Methodology

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 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. We introduce a systematic approach to probabilistic ontology development which focuses on evolving a traditional ontology from conceptualization to probabilistic ontology implementation for real-world problems. The Probabilistic Ontology Development Methodology is an efficient, teachable, and repeatable technique 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, methodology, inferential reasoning

I. INTRODUCTION

The use of ontologies is on the rise, as they facilitate interoperability and provide support for automation. Today, ontologies are popular in areas such as the Semantic Web [1], knowledge engineering, Artificial Intelligence (AI) and knowledge management [2]. However, many real world problems in these disciplines are burdened incomplete information and other sources of uncertainty which traditional ontologies cannot represent [3]. 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 … [4]."

This paper introduces a systematic approach to probabilistic ontology development which focuses on evolving a traditional ontology from conceptualization to probabilistic ontology implementation for real-world problems. The Probabilistic Ontology Development Methodology is an efficient, teachable, and repeatable means to develop, implement and evaluate explicit, logical and defensible probabilistic ontologies developed for knowledge-sharing and reuse in a given domain [5].

A. Probabilistic Ontology Development Methodology

We define a Probabilistic Ontology Development Methodology (PODM) that extends Carvalho [4] and 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 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.

B. Terminology

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

Taxonomy: A taxonomy is a classification structure for ordering objects into categories [6].

Ontology: An ontology is an explicit specification of a conceptualization [7]. It should include well-define syntax and semantics, efficient reasoning support, and sufficient expressive power [8].

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 [9].

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 [10].

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

Activity: An activity is a constituent undertaking of a process [11].

Task: A task is the smallest unit of work subject to management accountability [2]. It is a well-defined work assignment for one or more project members. Related tasks are grouped to form activities [11].

II. BACKGROUND

The Probabilistic Ontology Development Methodology (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 1 [12]. 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.

Complex development problems require a series of 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 1, and their PODM-specific tasks, are overlaid onto a Spiral Development Cycle in Figure 2 [13][14][15][16].

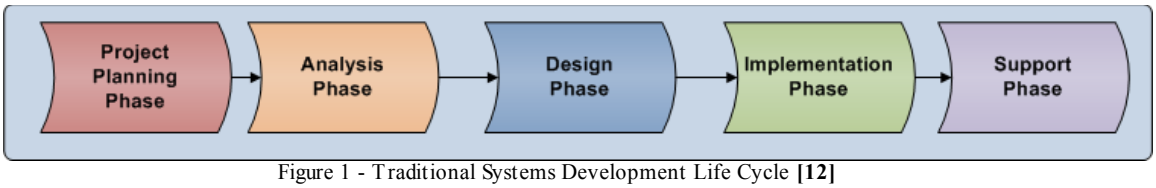


Figure 1 - Traditional Systems Development Life Cycle **[12]**

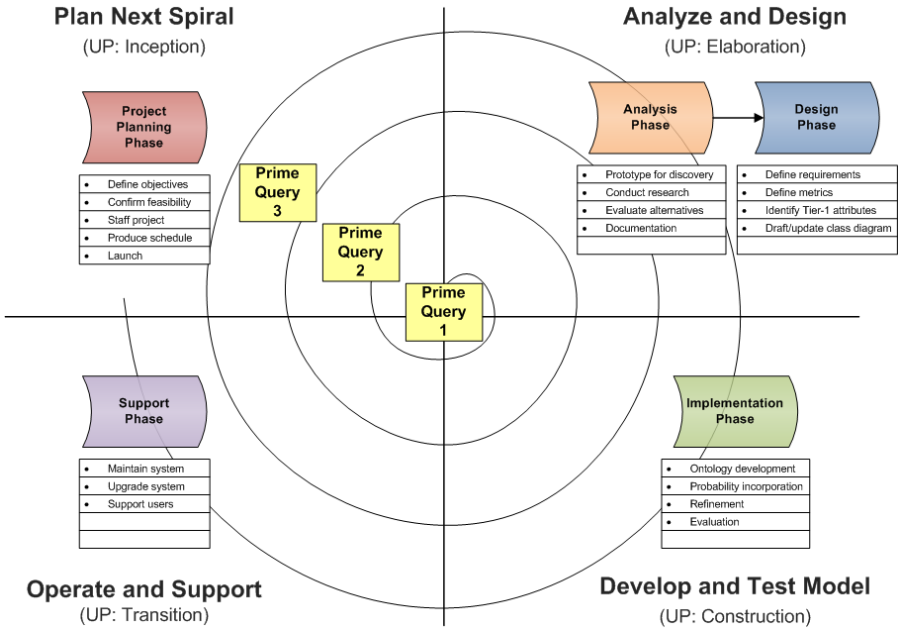


Figure 2 - PODM in Spiral Development Cycle

The Spiral Development Cycle phases also correspond to similar phases in the Unified Process (UP): Inception, Elaboration, Construction, and Transition [15]. The PODM is equally applicable to all three development processes, as captured in Table 1. 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 is completed by 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 in each spiral, as discussed below.

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 described below.

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

Table 1 - PODM Development Cycle Alignment

SDLC Phase [12]			
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

III. 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.

A. *Plan Spiral (Project Planning Phase / UP: Inception)*

Planning a 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 [12]. 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.

B. *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 [12]. 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.

C. *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 cycle 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.

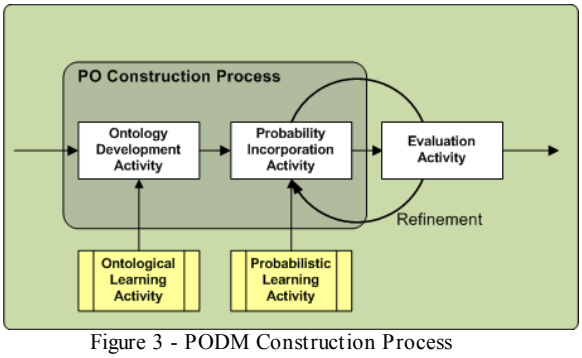


Figure 3 - PODM Construction Process

The shaded area in Figure 3 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.

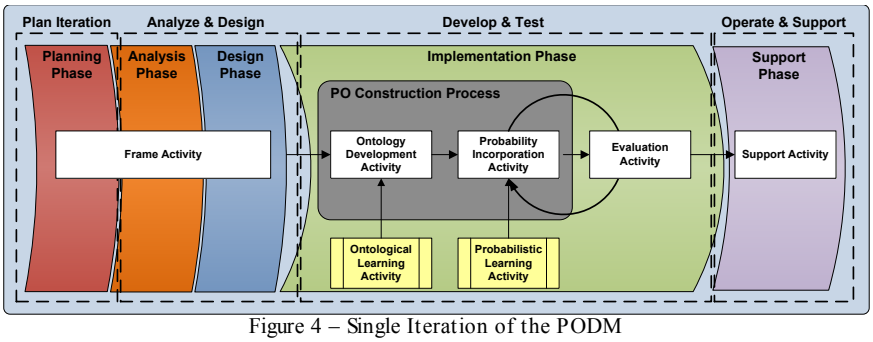


Figure 4 – Single Iteration of the PODM

D. *Operate & Support (Support Phase / UP: Transition)*

The Operate & Support Phase includes three major functions: maintenance, improvement, and operational support [12]. 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.

IV. 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 4. The remainder of this section delves further into the details of the PODM. The activities of the PODM and their tasks are illustrated in Figures 5-8. 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. A complete description of each activity and its component subtasks follows.

A. *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 Top-level project Objective Statement. The five tasks shown in Figure 5 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.

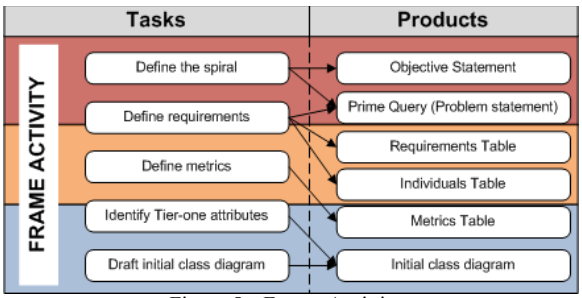


Figure 5 - Frame Activity

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.

1) *Frame Activity Tasks*

a) *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.

b) *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 [17]. 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.

c) *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. 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 [18]. 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.

d) 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:

Based on the Prime Queries, identify the class of objects about which the reasoning will occur.

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 [19].

At the most general level, identify the classes that are affected by these relationships. The established relationships and classes populate the initial class diagram.

e) Draft Initial Class Diagram

The initial class diagram enables the PO Developer to visualize and communicate 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.

2) Frame Activity Products

a) 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.

b) 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.

c) 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. Requirements definition requires close collaboration between the PO Developer, the Stakeholder DM, SMEs and users. The literature describes several methods for elicitation of requirements.

d) Individuals Table

Each class within the ontology contains individuals that are specific to the domain of interest. However, the same model could be employed for inferential reasoning support for a similar problem in the same domain. 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. A limited set of instances is used to demonstrate feasibility of the methodology without stressing the software application. The final operational application of an ontology may have hundreds of instances.

e) 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 [18]. These metrics are used to validate the spiral model against its requirements.

f) 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. 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.

g) 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. 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.

B. Ontology Development Activity

An ontology is used to capture consensual knowledge about a domain of interest [2]. The Ontology Development activity summarizes the non-trivial ontological engineering tasks required to produce a working ontology, shown in Figure 6. 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 6 and discussed below.

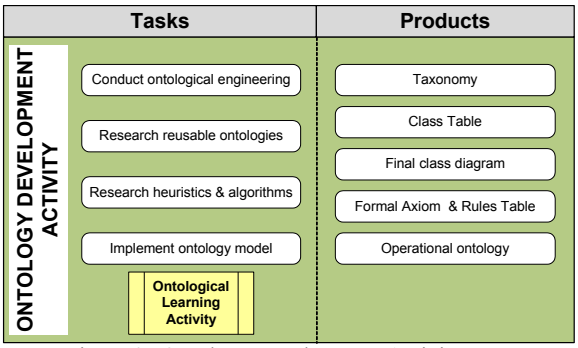


Figure 6 - Ontology Development Activity

1) *Ontology Development Activity Tasks*

a) *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 [2]. 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. [2]. 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

Identify Classes: *what objects are acting or acted upon?*
Develop Context: *where or when are the actions occurring?*
Identify Relationships: *what objects are affected?*
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.

b) *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. 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. [2]. 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 techniques in Gomez-Perez et al. [2].

c) *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 [20]. They are captured in a Formal Axiom & Rules Table for incorporation into the spiral model.

d) *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 [2]. Protégé is based on the Frames construct and supports first-order logic reasoning [2] [21].

e) *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. The interested reader will also find further information on these topics in Gomez-Perez et al. [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 6 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.

a) *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. Relationships between classes in the ontology are described by object properties.

b) *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. A concise Class Table allows implementation of the ontology in one of several ontological packages introduced previously.

c) *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.

d) *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 [2]. The Formal Axioms and Rules Table also captures heuristics and algorithms that act as constraints for the model. The entries in this table transform plain language constraints into formal, machine processable form.

e) *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 [7]*."*

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

C. *Probability Incorporation Activity*

The Probability Incorporation Activity is the heart of the PODM, illustrated in Figure 7. 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 PODM has been tested using Multi-Entity Bayesian Networks to model a 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 [9]. 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 [22][23][24]. In the following sections, illustrations of appropriate MEBN components captured in the UnBBayes software tool are provided for clarification.

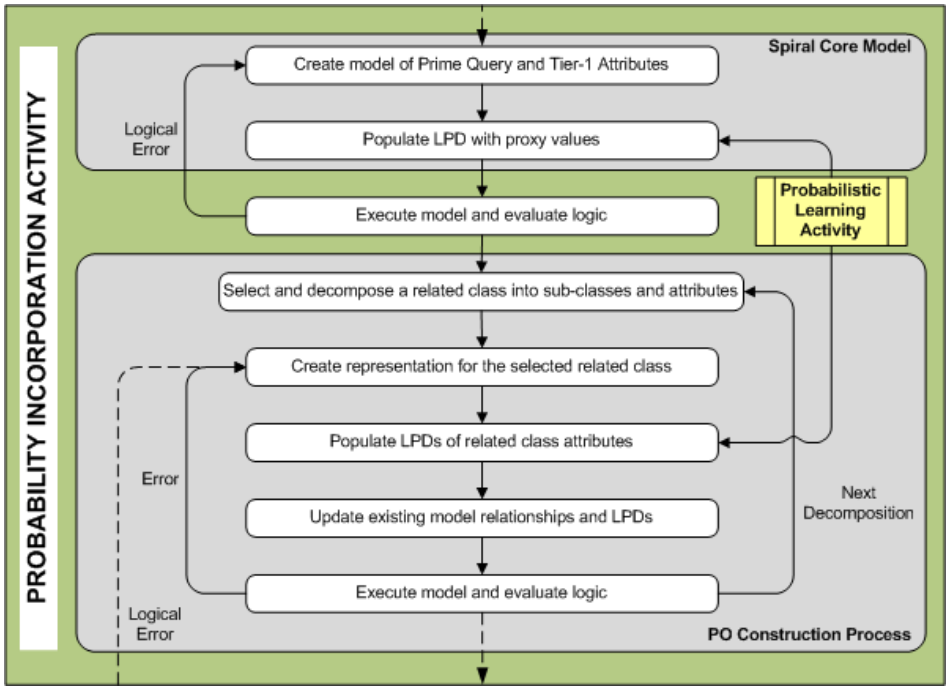


Figure 7 - Probability Incorporation Activity and PO Construction Process

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 9) shows how the Spiral Core Model for the first spiral of the development process for a 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.

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.

a) *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 Model, captured in the simple MEBN below (Figure 8).

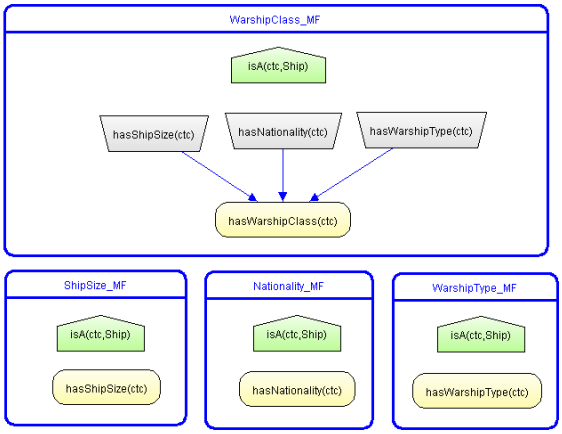


Figure 8 - MilShip PO Core MEBN Model

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.

b) *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.

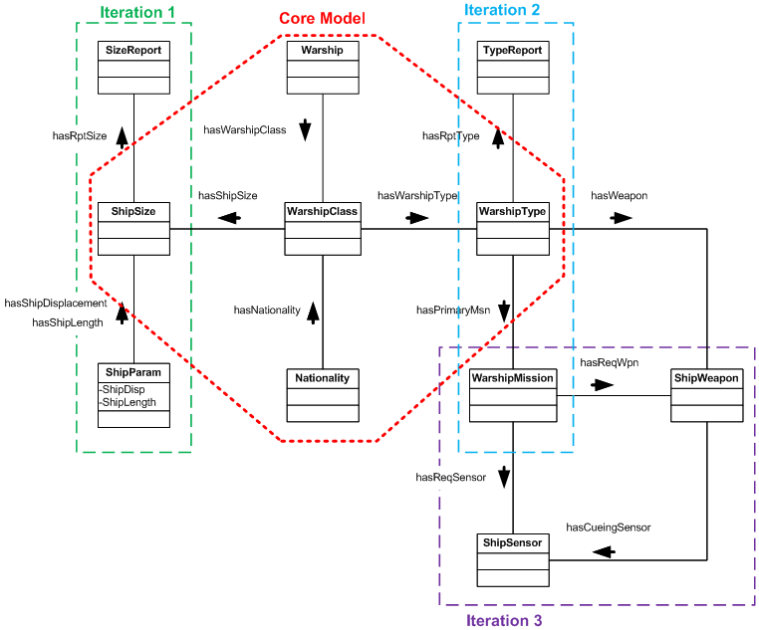


Figure 9 - PO Construction Process Iteration Plan

a) Execute Model and Evaluate Logic

Using the software tool, the model is executed with test evidence. 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.

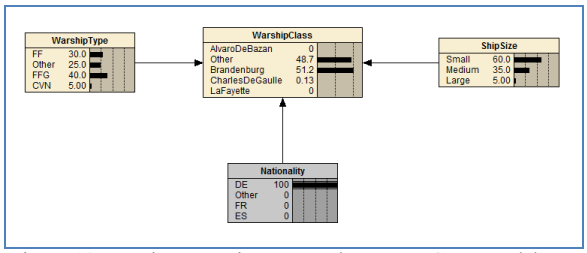


Figure 10 - Netica Bayesian Network to Test Core Model

The Bayesian network in Figure 10 was created using the Netica [25] 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.

b) 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) [26].

2) Probabilistic Ontology Construction Process

The iterative steps follow the PO Construction Process, shown in Figure 7 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

a) 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.

b) 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.

In the MilShip PO, the UnBBayes MEBN MFrag for the Ship Size is shown above (Figure 11). 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).

a) 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. 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.

a) 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. 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.

a) 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

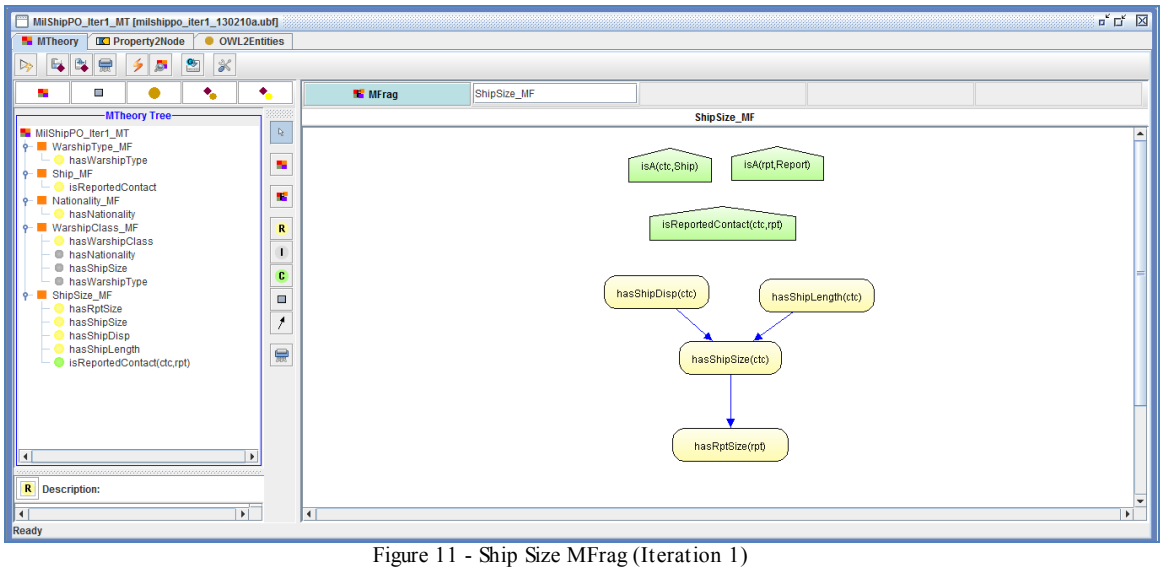


Figure 11 - Ship Size MFrag (Iteration 1)

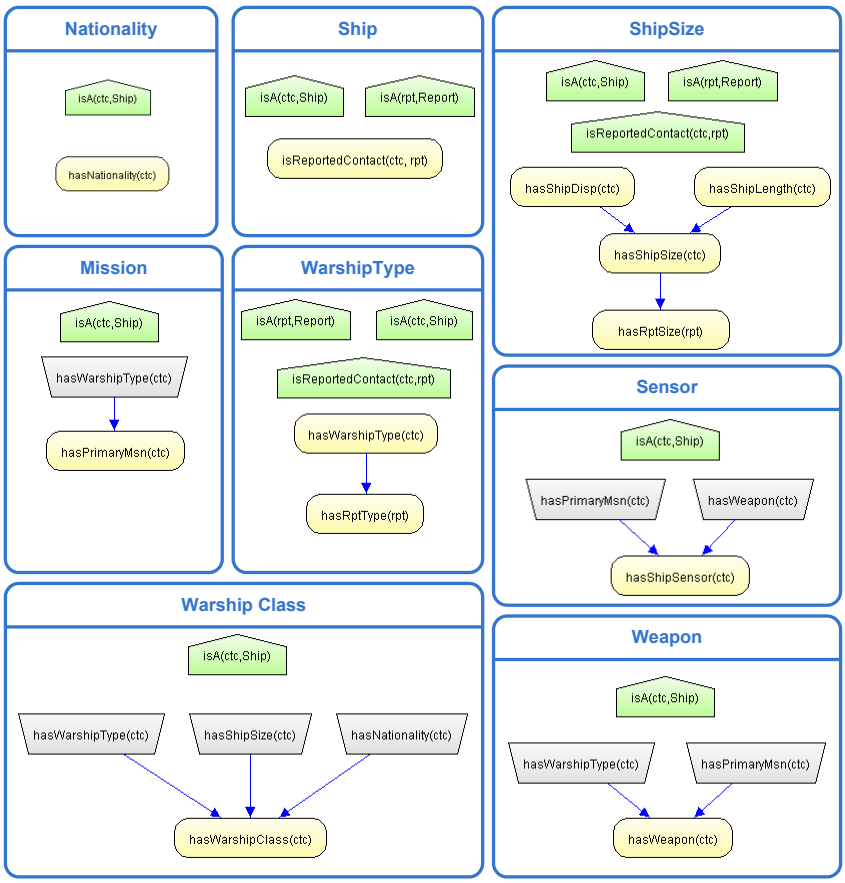


Figure 12 - Completed MilShip PO MTheory

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. 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.

Three iterations of the model are conducted to complete the first spiral of the Military Ship PO as shown in

Figure 9 and discussed above. The completed MTheory model is shown in Figure 12.

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.

D. Evaluation Activity

The Evaluation Activity completes the PODM as shown in Figure 13 by two methods. 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.

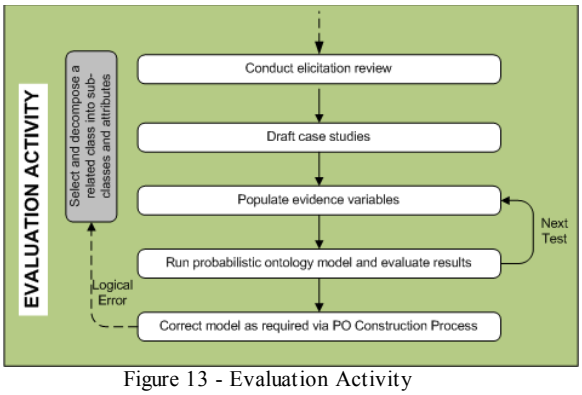


Figure 13 - Evaluation Activity

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 [27]. This is a qualitative assessment provided by expert reviewers including the Stakeholder DM, SMEs, and users.

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 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.

3) Populate Evidence Variables

For each case study, the appropriate evidence is incorporated into the PO model using FOL statements.

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 7 and Figure 13). A test case that performs as expected is documented, and the next test case is applied.

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.

E. Support Activity

As introduced in Section III.D, the Operate & Support Phase of the SDLC includes three functions: maintenance, improvement, and operational support [12]. 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.

V. SUMMARY

The Probabilistic Ontology Development Methodology provides a specific, guided methodology to implement the reference architecture introduced in [28]. 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.

A. 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 methodology supports development of a PO from

conceptualization to operation, but the PODM is equally useful incorporating and testing uncertainty into an existing ontology.

B. *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. However, it would be a straightforward task to increase the individuals within the existing framework by populating the ontology with additional 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.

VI. REFERENCES

[1] Time Berners-Lee. (1988, September) Semantic Web Roadmap. [Online]. http://www.w3.org/DesignIssues/Semantic.html

[2] Time Berners-Lee. (1988, September) Semantic Web Roadmap. [Online]. http://www.w3.org/DesignIssues/Semantic.html

[3] W3C Incubator Group. (2008, March) Uncertainty Reasoning for the World Wide Web. [Online]. http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/

[4] Rommel Novaes Carvalho. (2011, June) PhD George Mason University. [Online]. http://hdl.handle.net/1920/6616

[5] Richard J. Haberlin, Probabilistic Ontology Reference Architecture and Design Methodology, 2013, PhD Dissertation.

[6] Dictionary.com LLC. (2013) Dictionary.com. [Online]. http://dictionary.reference.com/

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

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

[9] Paulo Cesar G. da Costa. (2005, July) PhD George Mason Univeristy. [Online]. http://hdl.handle.net/1920/455

[10] IEEE, IEEE Standard Glossary of Software Engineering Terminology. New York: IEEE Computer Society, 1990.

[11] IEEE, IEEE Standard for Developing Software Life Cycle Processes. New York: IEEE Computer Society, 1996.

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

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

[14] 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.

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

[16] INCOSE, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Seattle, 2008.

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

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

[19] Kevin B. Korb and Ann E. Nicholson, Bayesian Artificial Intelligence. Boca Raton: CRC Press, 2011.

[20] Dictionary.com, LLC. (2013, June) "axiom" in Dictionary.com Unabridged. [Online]. http://dictionary.reference.com/browse/axiom?s=t

[21] Stanford University. (2011, January) Protege. [Online]. http://protege.stanford.edu/

[22] University of Brasilia - UnB. (2010) UnBBayes. [Online]. http://unbbayes.sourceforge.net/index.html

[23] 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.

[24] 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.

[25] Norsys Software Corp. (2013, June) Netica. software. [Online]. http://www.norsys.com/index.html

[26] 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.

[27] 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.

[28] Richard J. Haberlin, Paulo C.G. Costa, and Kathryn B. Laskey, "A Reference Architecture for Probabilistic Ontology Development," in Proceedings of the 9th International Conference on Semantic Technology for Intelligence, Defense, and Security, Fairfax, VA, 2013, pp. 1-9.

[29] 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.

[30] 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.

[31] William B. Rouse and Andrew P. Sage, Handbook of systems engineering and management. Hoboken: John Wiley & Sons, 2009.

[32] 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.

BIOGRAPHIES

Richard J. Haberlin, Jr. is a Senior Analyst and Subject-Matter Expert for EMSolutions, Inc. in Arlington, Virginia. Dr. Haberlin is a retired U.S. Naval Flight Officer with extensive experience in anti-submarine warfare and airborne intelligence, surveillance and reconnaissance operations in the Arctic, Atlantic, Mediterranean and Middle East. His research interests include inferential reasoning, probabilistic ontology development, Bayesian networks, and model-based systems engineering.

Kathryn Blackmond Laskey is Professor of Systems Engineering and Operations Research and Associate Director of the Center of Excellence in Command, Control, Communications, Computing and Intelligence at George Mason University. Dr. Laskey teaches courses in systems engineering, computational decision theory, and decision support. She has published extensively in the areas of inference, knowledge representation, learning, and information fusion. She developed Multi-Entity Bayesian Networks (MEBN), a language and logic that extends first-order logic to support probability. She was a key contributor to the development of PR-OWL, an upper ontology that allows MEBN theories to be represented in OWL ontologies.

Paulo Cesar G Costa is Associate Professor of Systems Engineering and Operations Research and Research Director for C2 Activities of the Center of Excellence in Command, Control, Communications, Computing and Intelligence at George Mason University. Dr. Costa is a retired Brazilian Air Force Flight Officer with extensive experience in electronic warfare, C4I, operations research and military decision support. He teaches courses in decision theory and systems engineering, and has developed PR-OWL, a probabilistic extension of the OWL ontology language. As an invited professor at University of Brasilia, he was a key contributor to the development of UnBBayes-MEBN, an implementation of the MEBN probabilistic first-order logic.

A Probabilistic Ontology Development Methodology

International Command & Control Research & Technology Symposium

June 16-19, 2014

Richard J. Haberlin Jr. *(EMSolutions)*
Paulo C.G. da Costa *(George Mason University)*
Kathryn B. Laskey *(George Mason University)*

EMSolutions

Background

Ontology Defined

An ontology is an explicit, formal representation of knowledge about a domain of application. This includes

- Types of entities that exist in the domain;
- Properties of those entities;
- Relationships among entities;
- 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 [Costa, 2005].

"An ontology is an explicit specification of a conceptualization [Gruber, 95]*."*

Ontology

- Ontologies provide a hierarchical structure of entity classes and a formal way of expressing their relationships
  - First-order expressivity
  - Supports logical reasoning
- There is significant literature on engineering traditional ontologies
- Ontologies lack built-in, principled support to adequately account for uncertainty
  - Annotating ontologies with simple probability annotations fails to convey structure of probabilistic representation
  - Less expressive probability schemes do not convey ontology structure, and so are inadequate

Probabilistic Ontology Defined

Ontology

Uncertainty

A *probabilistic* ontology is an explicit, formal representation of knowledge about a domain of application. This includes

- Types of entities that exist in the domain;
- Properties of those entities;
- Relationships among entities;
- Processes and events that happen with those entities;
- **Statistical regularities that characterize the domain;**
- **Inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge related to entities of the domain;**
- **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 described and reasoned about within the domain of application [Costa, 2005].

A probabilistic ontology extends a traditional ontology to represent uncertainty.

EMSolutions

Probabilistic Ontology

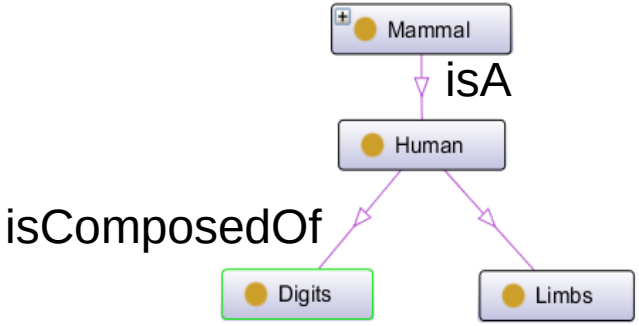
- Integrates inferential reasoning power of probabilistic representations with first-order expressivity of ontologies
- Provides a means to represent and reason with uncertainty
- Limited literature on construction

Comprehensively describes knowledge about a domain and the uncertainty embedded in that knowledge in a principled, structured and sharable way [Brisset, 2003].

"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 [Carvalho, 2011]."

Why Probabilistic Ontologies?

- Suppose an ontology of organisms contains the following classes and relationships:



- Humans *usually* have:
  - 2 arms & 2 legs
  - 10 fingers & 10 toes
- However, if a man loses a limb….
  - Is he no longer human?

Premise of an argument can be uncertain (e.g. Humans have 2 legs): (in)validity of the argument imposes no condition on the certainty of the conclusion (an amputee is Human).

EMSolutions

The Problem

- The Semantic Technologies (ST) community needed a comprehensive methodology for the development, implementation, and evaluation of probabilistic ontologies
  - Ontology use is on the rise
  - A means to incorporate uncertainty is a necessity
  - Limited literature on production of probabilistic ontologies
- Ontological engineering ensures ontologies developed for knowledge-sharing and reuse are explicit, logical and defensible
- Standard ontological engineering methods provide insufficient support for complexity of probabilistic ontology development

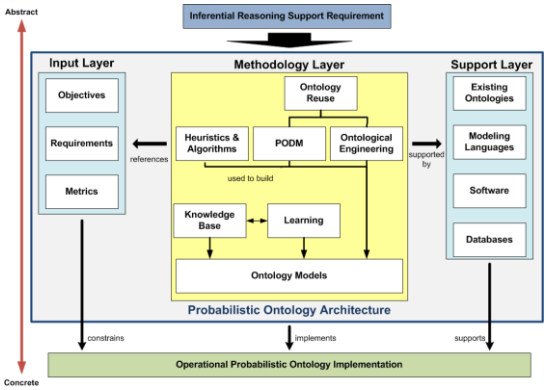
A similar methodology is needed for development of probabilistic ontologies

Our Solution

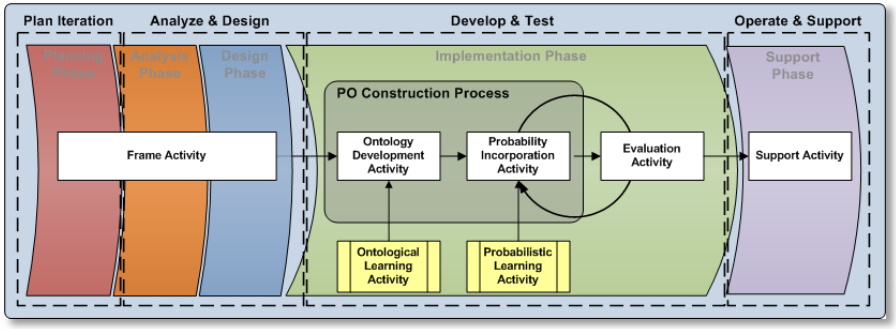
Create a systematic approach to probabilistic ontology development

- Facilitated through a reference architecture
  - Formalizes the application of the methodology
  - Extensible to various domains
- Follows an iterative methodology applicable to any Systems Engineering development process
  - Allows continuous expansion and evaluation
  - Simplifies development and logic checking through spiraling
- Ensures the implemented design meets requirements

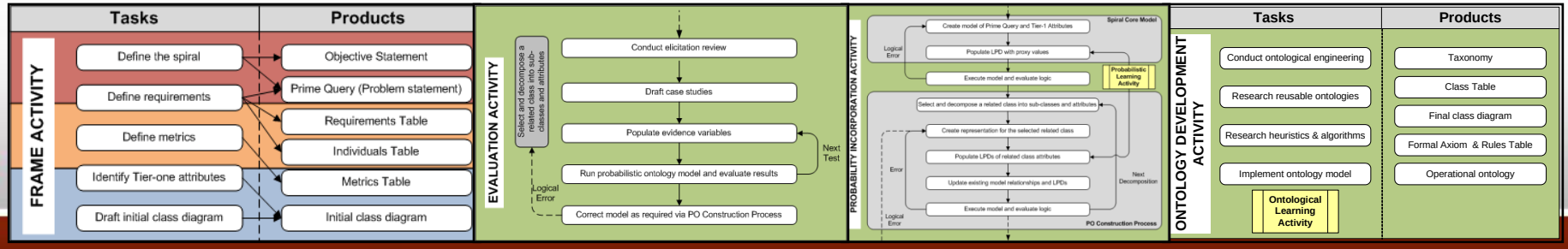
The Process of Probabilistic Ontology Development



Reference Architecture for PO Development



PO Development Methodology (PODM)



Probabilistic Ontology Development Methodology (PODM)

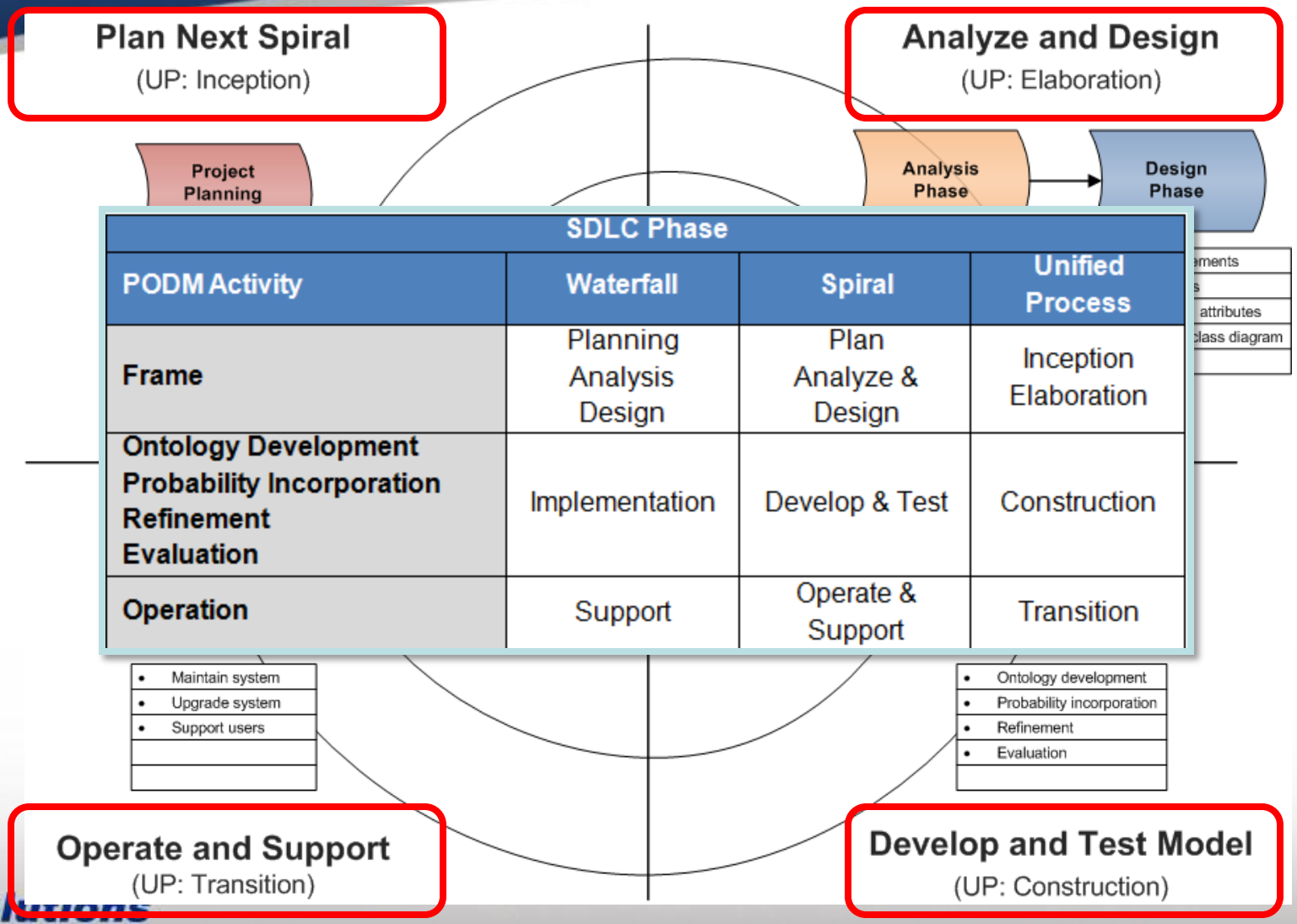
EMSolutions

PODM Background

- PODM addresses evolution of requirements into an ontology that is probabilistically-integrated
  - Explicitly describes the iterative tasks required to produce a PO with in-situ evaluation steps
- Suitable for both spiral and waterfall development processes
- Application of PODM
  - Specific decision problem
  - Grounded in an inclusive ontology representing its entities
  - Incorporates probabilities to represent uncertainty

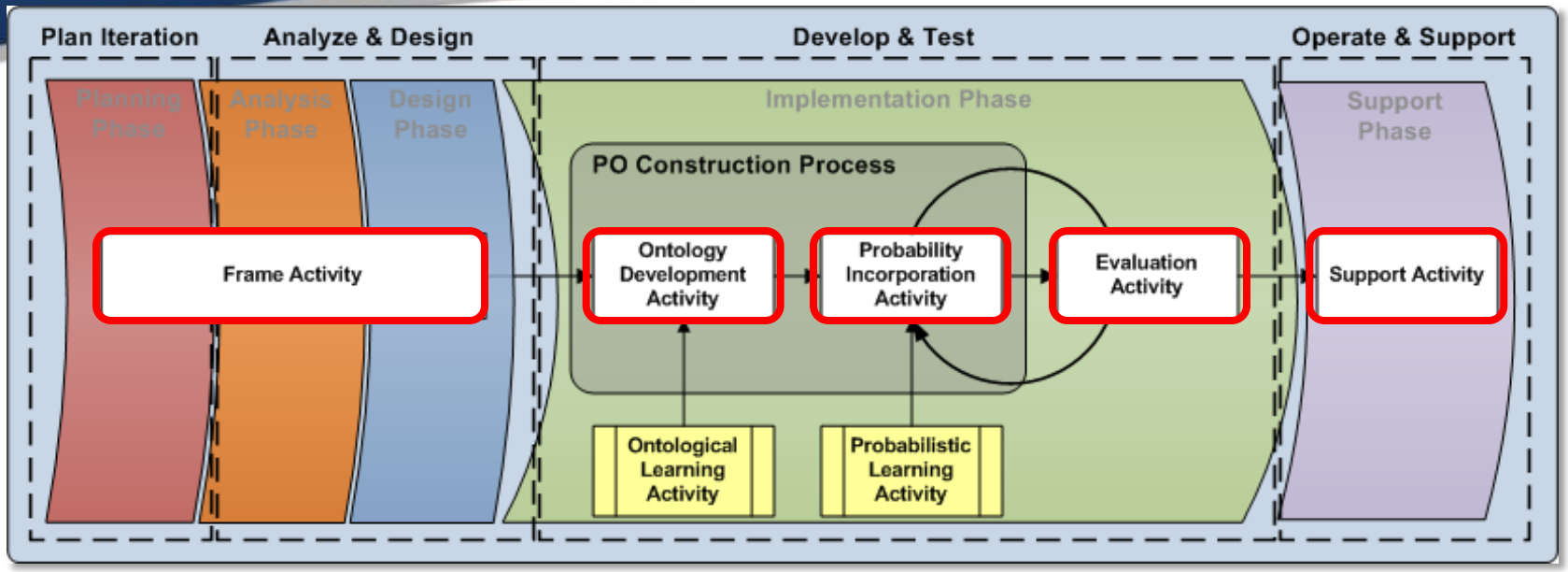
Establishes a solution grounded in an inclusive ontology representing its entities and incorporation of probabilities to represent uncertainty

PODM in Spiral Development Cycle



SDLC Phase			
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

Probabilistic Ontology Development Methodology

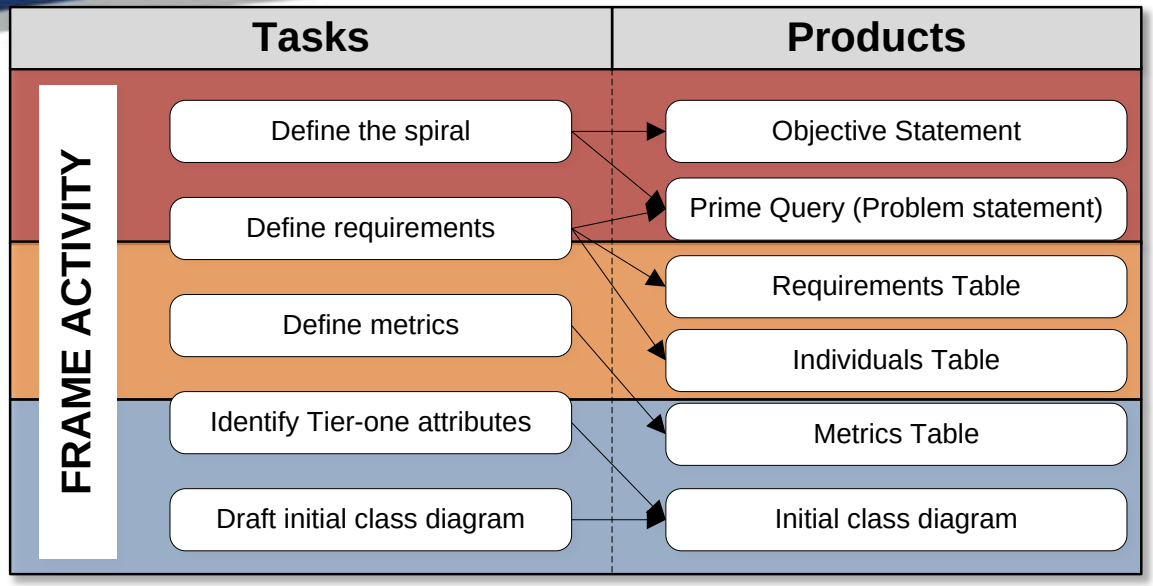


Maintenance, improvement and operational support tasks to maintain currency on the current build and support user operation.

processes

- Ensures interim steps evaluated for valid relationships and correct logic

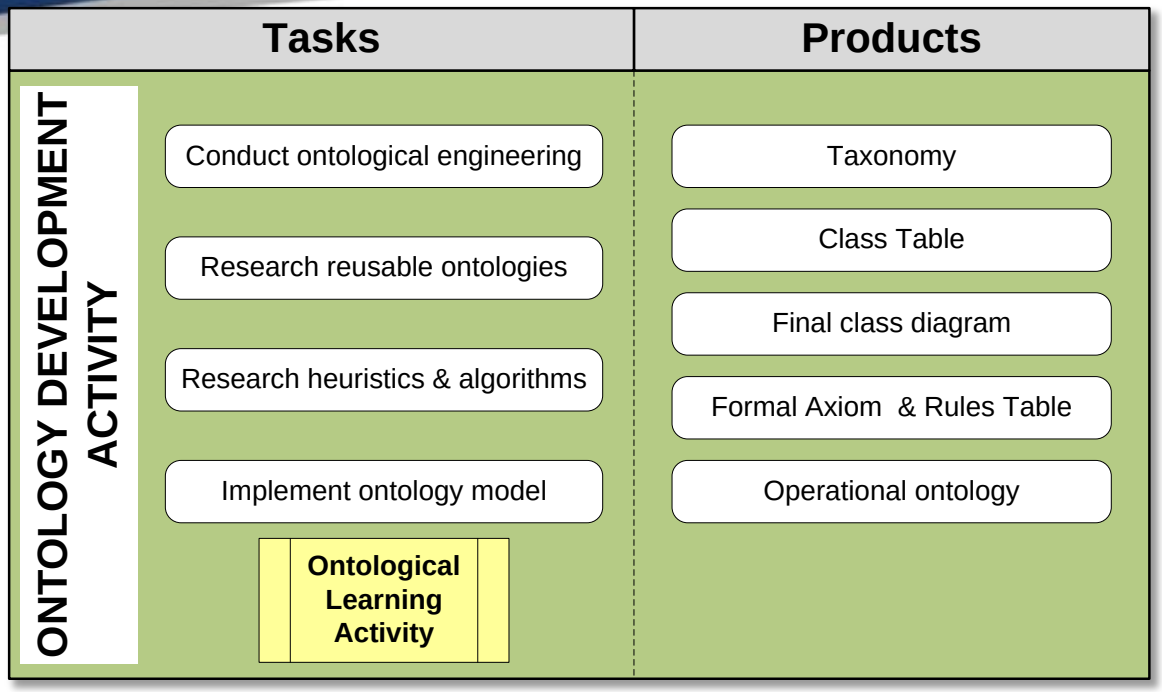
Frame Activity



- Based on the Overarching and Spiral Objective Statements
- Defines the Spiral Core Model
  - Prime Queries
  - Tier-one Attributes
- Scopes the spiral with the stakeholder's requirements and constraints

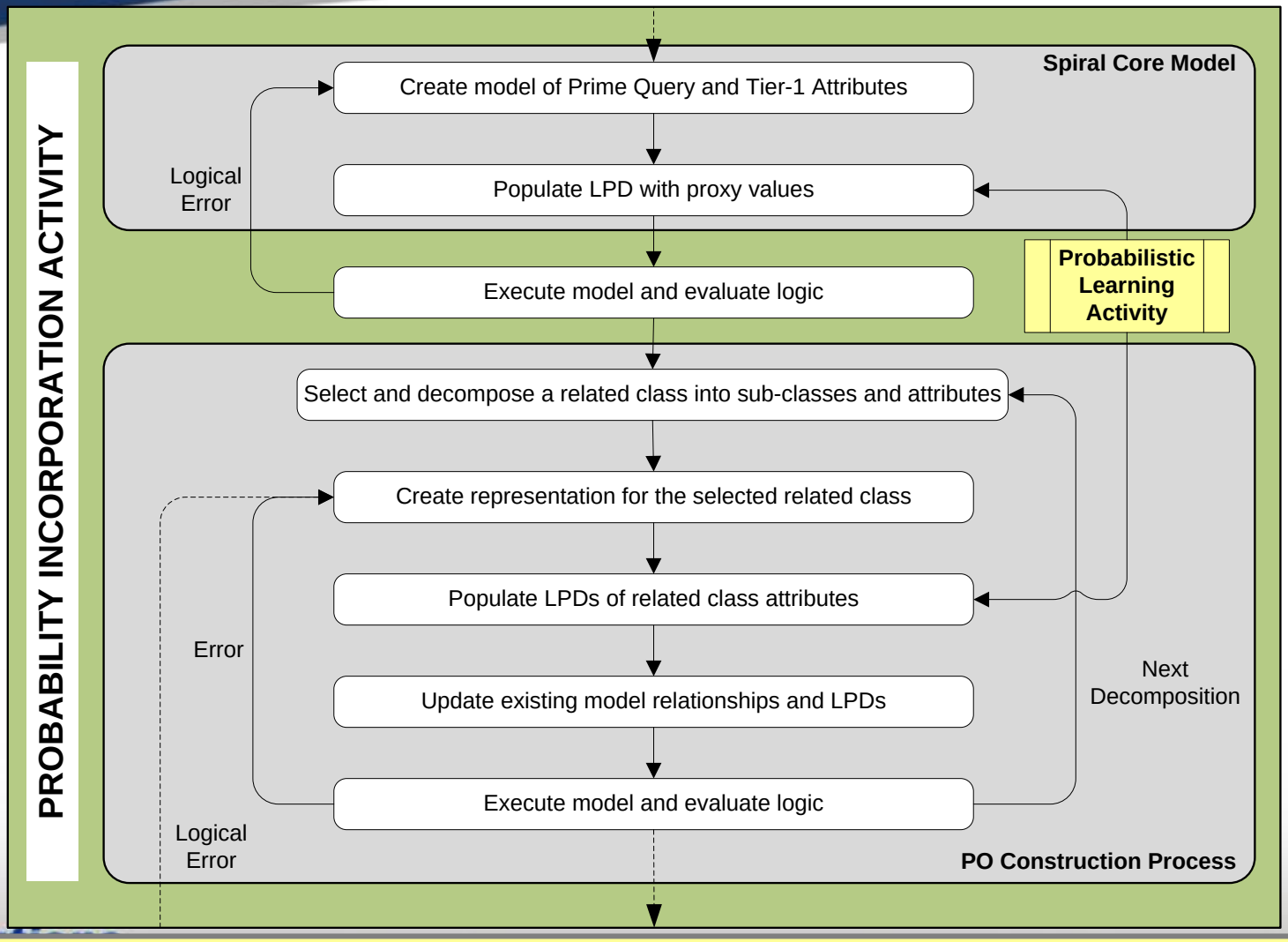
The Prime Queries and their associated Tier-one attributes define the spiral Core Model iterated in the construction cycle to create the necessary PO for inferential reasoning

Ontology Development Activity



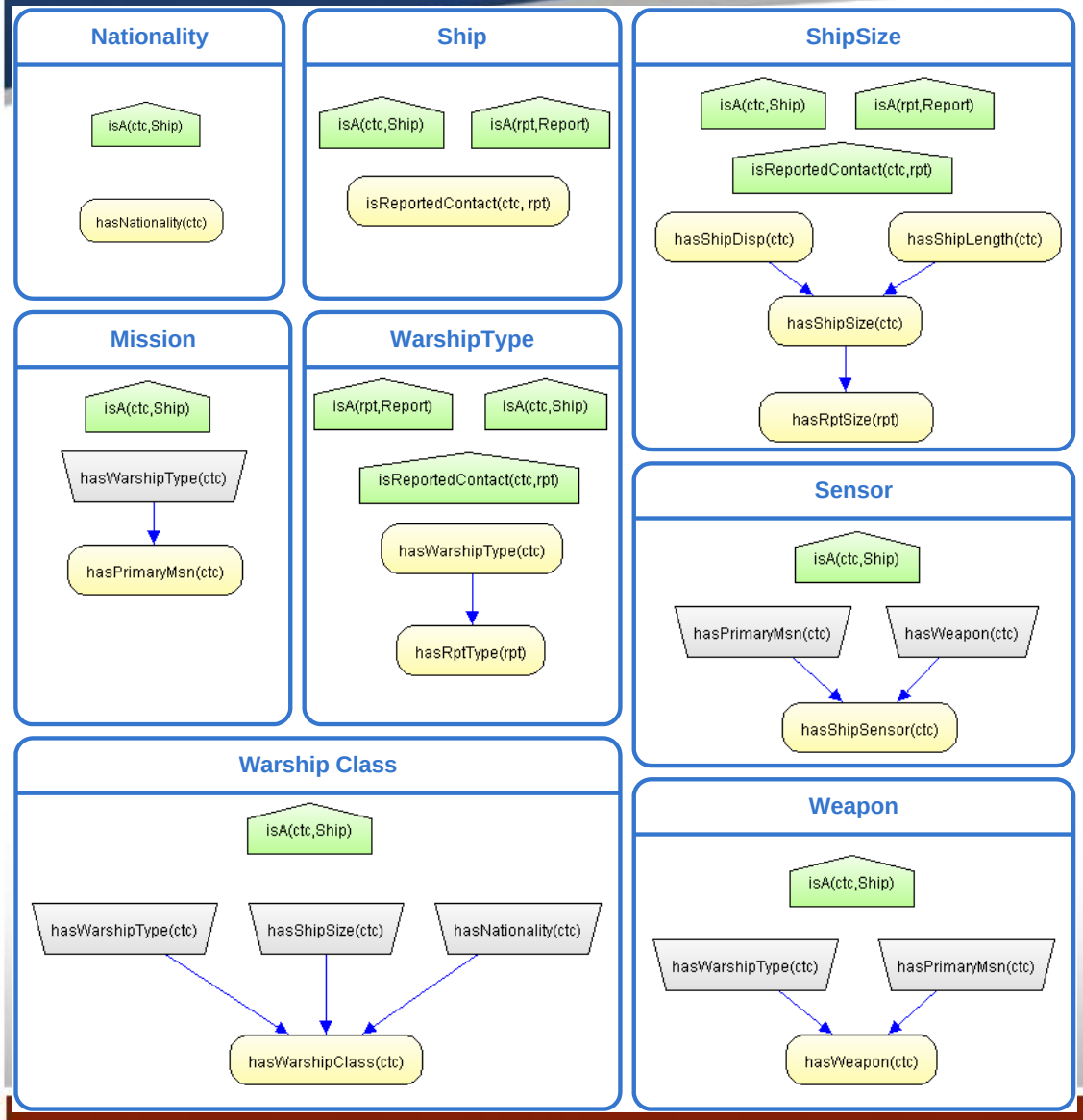
- Summarizes engineering tasks required to produce a working ontology
- Selection of an ontological engineering methodology is context dependent
- Fidelity of the ontological model is context dependent
- There are tasks and products common to each of these processes

Probability Incorporation Activity



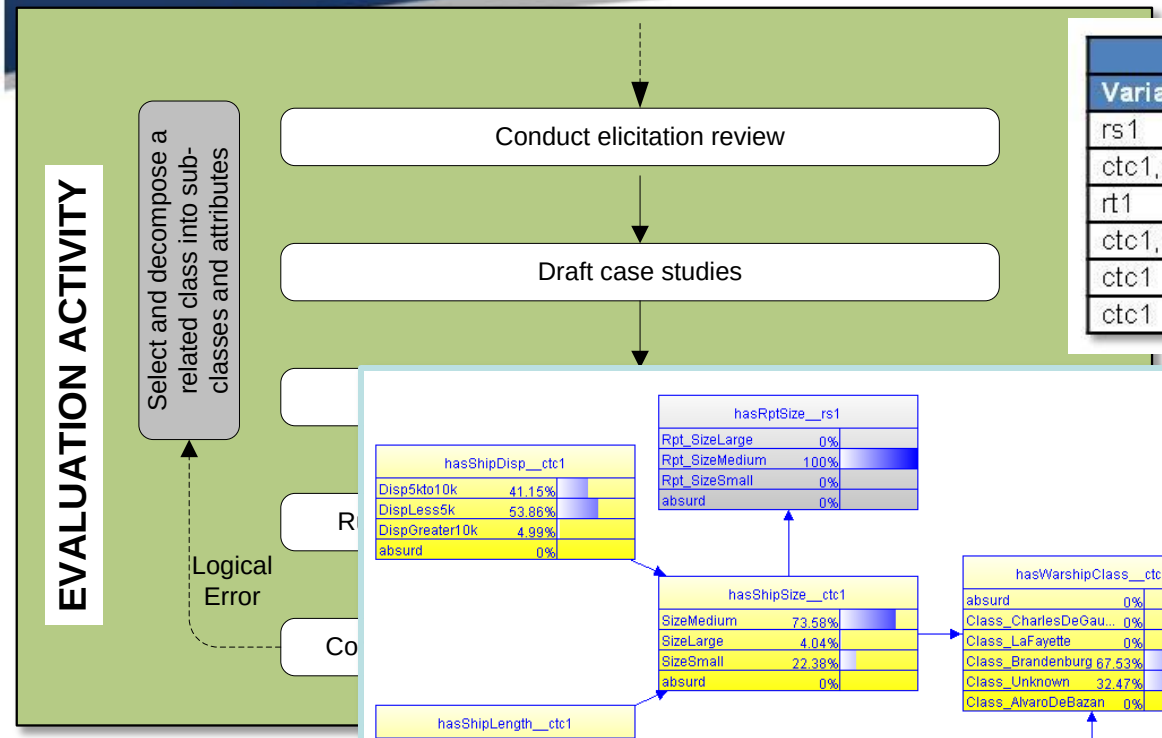
Prime Query + Tier-1 Attributes = Spiral Core Model

Completed Military Ship PO

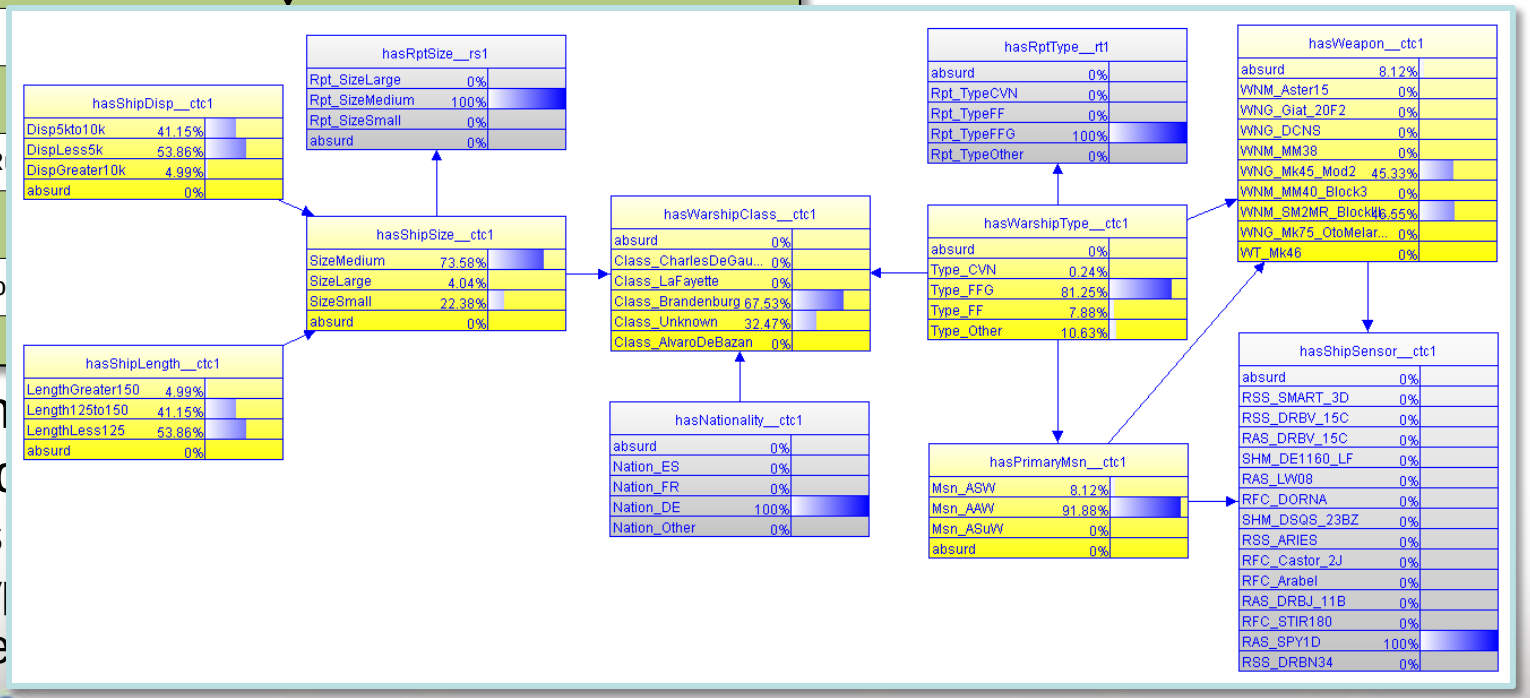


From an AOR-specific library (ontology), the MilShip PO infers the warship cass of an unknown contact based on limited or conflicting reports of varying pedigrees (uncertainty) .

Evaluation Activity



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



- Elicitation
- Case stud
  - Cases
    - Ty
  - Errone

Test cases ensure model meets requirements specified by objective

Conclusion

Problem

- Ontological engineering methodologies are unsuitable for production of probabilistic ontologies.
- The literature on probabilistic ontology development is extremely limited.

Solution

- Reference Architecture for Probabilistic Ontology Development
- Probabilistic Ontology Development Methodology

A Probabilistic Ontology Development Methodology

Richard Haberlin
Paulo Costa
Kathy Laskey

EMSolutions