Applying Model-Based Ontology Coverage Analysis to Mission Architectures

Yaniv Mordecai
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
yanivm@mit.edu

Aleksandra Markina-Khusid
The MITRE Corporation

202 Burlington Rd,
Bedford, MA 01730
amk@mitre.org

Greg Quinn
The MITRE Corporation

202 Burlington Rd,
Bedford, MA 01730
gquinn@mitre.org

Edward F. Crawley
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
crawley@mit.edu

***Abstract*—This paper introduces a method for Model-based Ontology Coverage Analysis (MOCA) and applies it to SysML models of mission architectures. An ontology is a set of concepts that constitute a common language, standard terminology, and consistent pattern reference across multiple models within an organization, industry, or domain. The purpose of MOCA is to assess the overlap between a system architecture model and a given ontology, and thereby the architecture model's compliance with the ontology and the ontology's utilization by the architecture. We demonstrate MOCA on a SysML model of a humanitarian airlift mission, using a conceptual mission architecting SysML profile model that serves as the ontology. MOCA automates and simplifies reasoning over models, and creates digital model-based artifacts that support stakeholders in concept validation, decision making, and system/mission design. Thus, MOCA enhances digital systems engineering.**

***Keywords*—Digital Engineering; Model-Based Systems Engineering; MBSE; Mission Architecture; Mission Engineering; Ontology; Ontological Analysis**

TABLE OF CONTENTS

1. INTRODUCTION

Model-based Mission Engineering (MBME) is the use of formal models to capture and analyze complex operational architectures, such as space exploration missions, defense campaigns, dedicated operations, and enterprise activities. MBME is a derivative of Model-based systems engineering (MBSE), which applies formal modeling languages (FMLs) to specify complex systems and processes. The Unified Modeling Language (UML) [1], Systems Modeling Language (SysML) [2], Business Process Modeling Notation [3], Object Process Methodology (OPM) [4], [5], and Petri Nets [6], [7] are common FMLs. MBME uses FMLs to describe mission models [8]–[10]. Digital Systems Engineering is the utilization of MBSE toolchains, models, and model-based artifacts [11]–[13] to enhance engineering and operational enterprise processes. Digital Mission Engineering similarly draws on MBME.

FMLs can represent mission architectures, thanks to the relatively domain-agnostic, general-purpose syntax and semantics of the building blocks and constructs of FMLs. For example, concepts like object, process, state, block, class, activity, and actor constitute the vocabulary of FMLs. These concepts are typically ascribed to unique geometrical shapes that can be deployed and associated in diagrams. They have a relatively broad scope of applicability.

An ontology is a set of concepts and patterns that are typical to a domain. These include domain terminology, business rules, organizational structures, professional jargon, and cross-cutting aspects (e.g., communication, logistics, security, and safety). General-purpose FML concepts are often insufficient or inadequate for capturing enterprise and operational contexts of mission architectures, from package delivery to manned interplanetary exploration campaigns. Mission architectures must comply with such domain patterns. Accordingly, mission architecture models must comply with ontological models.

ABBREVIATIONS AND ACRONYMS

Acronym	Full Term
ACR	Architecture Compliance Report
archifact	architectural artifact
ASoT	Authoritative Source of Truth
CMGVC	Concept-Model-Graph-View Cycle
FML	Formal Modeling Language
MBME	Model-Based Mission Engineering
MBSE	Model-Based Systems Engineering
MOCA	Model-based Ontology Coverage Analysis
OAO	Ontology-Architecture Overlap
OUR	Ontology Utilization Report
SysML	Systems Modeling Language
UML	Unified Modeling Language

978-1-7281-7436-5/21/$31.00 ©2021 IEEE

Many reference frameworks, terminologies, vocabularies, and pattern libraries can be encoded as ontologies, by clearly listing the set of concepts and the set of relations among concepts in each reference. Mission architecture references that can be encoded as ontologies include: a) The Department of Defense Architecture Framework (DoDAF) [14], [15]; b) The Department of Defense Mission Engineering reference [13], [16]; c) MITRE's ATT&CK (pronounced: Attack) Framework for cyber resilience [17]; d) Leveson's System-Theoretic Accident Model and Processes [18], [19]; e) Systems Engineering and Conceptual Architecting references, such as those suggested in [20]–[23]; and f) standard, domain-specific, mission-specific, or problem-driven performance criteria [24], [25].

We can significantly enhance our assessment, verification, validation, and revision of mission architectures based on the ontologies that govern them. This requires: a) encoding concept and pattern references as ontologies; b) encoding mission architectures as formal conceptual models; c) classifying mission architecture artifacts according to ontological concepts; and d) applying ontological analysis to the architectural and ontological models.

Ontology-based frameworks for conceptual model analysis and validation have been shown to contribute to the reduction of semantic vagueness and ambiguity in conceptual models. Semantic vagueness is a common phenomenon that results from subjective human perception, conceptualization, and modeling decision-making [26], [27]. A formal model can be syntactically correct-by-construction by using modeling software tools that enforce syntax. However, enforcing compliance with an ontology involves methodological and computational challenges.

Manual or visual compliance analysis may be possible for simple models, thanks to human intuition and cognition, but it does not scale out for complex multi-model, multi-aspect architectures, with multiple governing ontologies. Automated compliance analysis, on the other hand, allows the analyst to focus on sense-making, anomaly detection, decision-making, and conclusion drawing, rather than searching and matching, which are considered simpler cognitive tasks. Some of the intuition and reasoning patterns that we are able to apply as humans to representations can be codified and automated, thereby reducing the cognitive load on the analysts and allowing them to focus on the more advanced and impactful cognitive tasks.

When we review a system or mission architecture, we interpret and expect the elements and constructs that it contains to resemble patterns and comply with ontologies we may have in mind, due to familiarity, experience, or our own adherence to well-defined specifications, international standards, federal regulations, or professional jargon. For instance, a communication engineer may expect to find items in a system architecture that have the semantics of communication: signal receiving and transmitting, often denoted as Rx and Tx, encoding and decoding, modulating and demodulating, etc. These can be reflected in the names of components, functions, and signals. An operator who studies a manual expects it to include steps and directions to procedures (e.g., activation of a system), interfaces, controls, indications, situations, and troubleshooting steps.

Complex representations and designs rely in many cases on the perception and sense-making that stem from the author's or designer's conception and interpretation. It is difficult to guarantee that any reader or viewer will interpret these representations exactly as intended by the author or designer. And yet, as humans, we communicate orally, textually, and graphically with other humans with more or less success as a matter of fact. Unfortunately, we cannot rely on the best effort of human cognition to guarantee correct, effective, and efficient realization of our ideas, when we design complex systems and missions. Bridging and closing the interpretation gap is possible if we apply ontological semantics to architecture and design artifacts.

To be more informed and constructive about the extent to which an architecture complies with an ontology, we would want to assess that compliance quantitatively, and iteratively strive to improve it. The compliance of the architecture with the ontology can be thought of as an overlap between two sets of concepts: those that the ontology defines as patterns or templates, and those that the architecture specifies as instances, as shown in **Figure 1**. Moreover, we aspire to maximize the overlap (**Z_2** in **Figure 1**) to improve that compliance, and concurrently minimize uncovered architecture (**Z_1**) and uncovered ontology (**Z_3**).

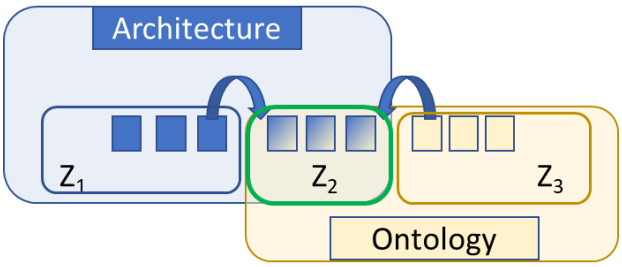


Figure 1. Maximizing the overlap between the Architecture and the Ontology provides a) better classification and compliance of architectural artifacts and b) better utilization of standard concepts

There are two major problems to consider: a) determining the coverage of an architecture by an ontology, which reflects the compliance of the architecture with the ontology, and b) determining the coverage of an ontology by an architecture, which reflects the utilization of the ontology by the architecture. These are not equivalent sets, as clearly illustrated in **Figure 1**. The former, denoted as $Compliance(Arch, Ont)$ in (1), measures the value of the overlap $v(Z_2)$ relative to the value of the architecture's scope – i.e. how much of the whole architecture $(Z_1 \cup Z_2)$ is classified (Z_2). The latter, denoted and defined in (2) as

$Utilization(Ont, Arch)$ measures how much of the ontology's scope $(Z_2 \cup Z_3)$ is utilized.

$$Compliance(Arch, Ont) = v(Z_2)/v(Z_1 \cup Z_2) \quad (1)$$

$$Utilization(Ont, Arch) = v(Z_2)/v(Z_2 \cup Z_3) \quad (2)$$

Maximizing ontological compliance has a synergistic and two-way effect of both better classification of architectural artifacts, and better utilization of ontological concepts. It is a win-win situation for both the stakeholders of the architecture (customers, integrators or implementors, and operators) and stakeholders of the ontology (national, professional, and organizational regulators, quality assurers, certification providers, and certified operators).

However, attempting to reach comprehensive ontological compliance in the ever-spreading ontology quagmire may be cumbersome, frustrating, and potentially Sisyphean. The scope of ontologies that an architecture may have to comply with is practically unbounded. Ontologies change all the time and changes may be hard to catch up with. The scope of applicable ontologies may result in redundancy, ambiguity, conflict, or contradiction in conceptual classifications of architectural artifacts. For example, the same architectural artifact can be thought of as both a functional feature of the system, a mechatronic technology application, an operational asset, a safety hazard, an enabler of mission success, a cost attractor, etc. Each one of these classifications comes from a separate ontology, and trying to classify the same system component according to all of these ontologies may be both cumbersome, hard to appreciate in terms of added value, and at times technically and cognitively difficult. A coherent approach to the assessment of ontological compliance may help resolve or reduce some of these challenges.

Mission architectures may consist of multiple models that use a variety of general-purpose and domain-specific FMLs as well as various, possibly overlapping, partial, or inconsistent ontologies. A mission architecture may be expected to comply with multiple ontologies to facilitate stakeholder understanding, adherence to conventions and standards, and architecture analysis. The challenge is in ensuring, assessing, and utilizing the ontological compliance of the mission architecture's ensemble of models.

To conclude, the problem of model-based mission architectures ontological coverage includes several aspects:

a. multiplicity of architecture models, conceptual modeling approaches, and modeling languages;

b. multiplicity of applicable references and partial formulation of such references as ontologies or ontological models;

c. difficulty in classifying artifacts according to multiple ontological concepts; and

d. lack of appropriate methodology for analyzing and assessing ontological compliance.

We propose a method for Model-based Ontology Coverage Analysis (MOCA). The purpose of MOCA is to provide aggregate and detailed information about i) a given architecture's ontology coverage vis-à-vis an expected set of ontologies, ii) each ontology's architecture coverage, which attests to the adoption and utilization of the ontology, and iii) the overlap between the two. These outputs capture the architectural and ontological artifacts in the three zones illustrated in **Figure 1**. The MOCA report should highlight specific issues in the mission architecture, such as architectural artifacts (*archifacts*) that are not associated with any ontology, or ontology concepts that are not instantiated anywhere in the architecture.

2. ONTOLOGIES AND MODELING

Ontology Defined

The term *Ontology* has several meanings: as a branch of philosophy, as an abstract theory, and as a practical artifact [28]. Ontology as a science (and perhaps art) is the study, categorization, and organization of the structure of nature, reality, existence, and perception – of things that be or can possibly be. Gruber asserted that "*an ontology is an explicit specification of a conceptualization"* [29].

An ontology of some domain is a conceptual organization of that particular domain. A vocabulary, a terminology of some domain of the universe, and the set of relations among terms, constitute a theory of the structure of the domain. Thus, an ontology is a set of abstract concepts and patterns that bind them, that exist or may exist in the domain that the ontology describes. Finally, an ontology is a knowledge storing artifact, which records the concept organization and can be instantiated as a model or applied to one.

Ontologies differ in their applicability to reality. Several levels of ontological precision determine the extent to which an ontology is binding, indisputable, and unavoidable: a) a catalog of items to choose from, b) a glossary that a specification should adhere to, c) a taxonomy that a hierarchy must adhere to, d) a Thesaurus, which provides a closed set of terms to work with, e) a data structure, which binds a data processing system to comply with, and f) an axiomatic theory, which determines or governs the laws of the universe [28]. Conceptualization schemes of types (e) and (f) are respectively either computationally constraining entities that reside and run in cyberspace (like the structure of a database), or scientifically constraining for things that exist or operate in reality (like the laws of Physics). They are therefore indisputable and unavoidable from the implementation or realization's perspective – i.e., a given instantiation cannot overcome them. The lower-precision grades of ontology are essentially disputable, electable, and avoidable – subject to will, belief, and perception – since they apply to conceptions and ideations rather than to implementations and realizations.

The ability to mandate and enforce ontologies in the creation and forming of ideas and conceptions is limited. Some may also argue that this is undesired, as it restricts the creative process. Ontologies mostly attempt to instill order into conceptualizations of complex natural and artificial systems as they are conceived, ideated, and designed. The adoption of an ontology depends on the acceptance and motivation of the people involved in the conceptualization process. For this reason, it becomes imperative to ensure that documenting and formulation of such concepts and designs also include the application of ontologies that a) propose a valuable reference for domain concepts, b) enrich the model with notions from particular ontologies of relevance and interest, and c) restrict model entities to adhere to the standard terminology and patterns dictated by the ontology, such that they protect designers from making mistakes or diverging from the enacted design language.

Ontology-Driven Modeling

Ontology-driven modeling is the process of building conceptual models using syntactically-finite modeling languages, and reinforcing the semantics of the model by classifying model elements and constructs according to ontological classifiers. As conceptual modeling languages provide syntactical certainty, the semantics of model constructs, which are essentially instantiations of syntactical-grammatical patterns, are unbounded due to the attribution of meaning by human perception. Perception may vary across model authors and readers, lead to ambiguity, and undermine confidence in the model's ability to serve as an Authoritative Source of Truth (ASoT).

Essentially, any conceptual model is, to some extent, a concept model, a meta-model, or an ontology of the solution domain [30]. For example, a model of a specific banking system could be considered as a representative and characteristic specification of banking systems in general. It is obvious that one specific banking system does not precisely describes any other banking system, but it is a good starting point. System models should be constructed as generic representations of the domains in which the modeled systems operate, to ensure solution rigor and robustness.

Metamodeling is the modeling of modeling languages [31], [32]. Metamodeling received a significant boost due to the availability of FMLs, particularly UML. Conceptual modeling has been harnessed for ontology modeling, i.e. for creating models that may constitute ontological, generic domain representations [33]–[37]. Metamodeling and ontology have some distinctions between them, but they are potentially intertwined and complementary ideas. Metamodeling addresses syntactical formality while ontology concerns domain concept formality. Both can be obtained separately, but a fusion of the two paradigms of ontology and metamodeling as the emerging paradigm of Ontology-Driven Systems Engineering has the potential to generate significant synergy for the formality of system and mission architectures [22], [38].

In some modeling systems, including UML, SysML, and OPM, a metamodel, or *profile*, can constrain a model through the classification of model elements as instances of profile elements, or *stereotypes*. Profiles can be imported into models, and the stereotypes defined in the former can classify artifacts specified in the latter. UML was designed to allow extensibility and adaptation through profiles and stereotypes [39]. Thus, SysML is a UML profile: it contains original UML types and some stereotypes like Block, Port, and Requirement that can be instantiated as artifacts in system models [2]. The Unified Profile for DoDAF and MoDAF – the British counterpart of DoDAF (UPDM), and the Unified Architecture Framework (UAF) are UML profiles [40], [41]. Additional UML-based profiles (or ontologies that can be profiles) are available [13], [16]–[22].

Ensuring the validity, applicability, and usefulness of ontologies, encoded as profiles in MBSE, has become a major challenge. Evidently, no profile is comprehensive enough to capture every aspect of a complex system. Specialized and standardized ontologies must be used, wherever possible, to create standardized architectures, facilitate stakeholder engagement and communication, reduce ambiguity and confusion, and enhance models' and systems' integration and interoperability [38], [42].

Functional decomposition of complex systems is one of the main activities that leverage conceptual modeling. An Ontology of Functions (OF) – a functional decomposition ontology [43], defines the function concept and the possible relations among functions. The definition of each function F should include a label $Label(F)$, a set of requirements, $Req(F)$, a set of goals $Goal(F)$, and a set of functional items $FItem(F)$. We add that each function's definition should also include a unique identifier $UID(F)$ and a version identifier $Ver(F)$, for better identification and configuration management of function specifications. Notably, the proponents of OF chose a set-theoretic, functional formulation, rather than an object-oriented one: *F.Label*, *F.Req*, etc. Relations among functions include instantiation, specialization, participation, and realization. Such relations are explicitly defined in the UML-SysML and OPM syntax [1], [2], [4], [5], so including them in OF only benefits modeling languages that do not provide syntax for such relations. An ontology for operational-functional unified specification that uses OPM syntax was introduced in [44].

Structural relations such as those captured in UML Class Diagram and SysML Block Definition Diagram are relatively easy to convert to ontologies. However, the behavioral and procedural aspects of the system, which are typically captured in Activity Diagrams and Sequence Diagrams, are more challenging to transform into ontologies or the behavioral sections of an ontology, as they also require the recording of order and precedence, as well as reasoning about order consistency and variation [45].

Reference architectures can constitute ontologies that specific solution architectures should or must adhere to. The

breadth of technologies and aspects in complex systems calls for integrated, hybrid, or multidimensional ontologies. For example, autonomous vehicles involve autonomy, mechatronics, robotics, artificial intelligence, advanced driver engagement concepts, communication, safety assurance, etc. Ontology-driven modeling of autonomous vehicle architectures can ensure that every aspect of the system complies with its own vocabulary. Concurrently, system-level interactions are specified by coherent abstract representations of system functionality, behavior, and modular structure. The OASys-driven Engineering Methodology (ODEM) is an ontology-driven MBSE approach that fuses an Autonomous Systems Ontology, OASys, with a systems engineering ontology [46]. ODEM uses a conceptual ontology layer where functional concepts are used for representing the autonomous system's elements, and they are then being instantiated into concepts drawn from an ontology of autonomy, consisting of such concepts as Localization, Navigation, Scanning, Perception, and Motion. In turn, these concepts are transformed into technology-specific implementations, such as 3D-Mapping, Obstacle detection, and Emergency Braking.

Mission Architecture Ontologies

Mission architecture ontologies are of particular interest for operating, acquiring, architecting, and building solutions with a clear mission statement in mind. Mission supporting ontologies may span the entire scope of science, engineering, business, and operations. Researchers at the Jet Propulsion Lab (JPL) pioneered the formulation and utilization of ontologies in model-based space mission architectures, as part of the Integrated Model-Centric Engineering (IMCE) initiative [47]–[52]. Ontologies for modeling and analyzing system states, behaviors, mission plans, and system failures, were found to be of particular interest. JPL's SysML-underpinned Behavior Ontology includes concepts like *Behaving_Element*, *Element Behavior*, *Interaction Behavior*, *Interaction*, *State*, and *State Variable* . These are captured as syntactical SysML elements with stereotypes. For example, *Element Behavior* is captured as a SysML Component's *Constraint Block*. The Behavior Ontology also encodes special relations between concepts, such as *characterizes*, *constrains*, and *uses*. A model-based formulation of DoDAF's Operational Viewpoint as a reference ontology for an operational architecture was suggested in [53].

Ontology Coverage

Ontological analysis is important and sometimes critical for assuring, measuring, verifying, validating, and enforcing ontological compliance of system models. It is quite difficult to measure and evaluate the extent of adoption and utilization of ontologies by large-scale system models, and therefore automated algorithms must be used.

Ontological analysis includes measurement and estimation of metrics like coverage, correctness, and similarity between reference ontologies and ontology instances or adaptations [54], [55]. Serial reference keyword similarity search as an estimator of coverage was also proposed in [54], using the a similarity index $Sim(O_1, O_2)$ that assesses similarity between two ontologies O_1, O_2 as a function of the number of pairs of items in each ontology. The similarity index is valid if the two ontologies have the potential to be identical.

Matching ontologies to an existing and evolving body of knowledge may be a significant challenge. An ontology-driven study of Earth-Scientific literature found inconclusive results, particularly in tracking the coverage of domain-specific ontology-defined axioms [55]. The study focused on four types of coverage: a) class coverage (CC), b) subclass coverage (SC), c) equivalence coverage (EC), and d) breadth coverage (BC), which is a linear combination of CC, SC, and EC. Coverage is calculated as the fraction of terms from the reference ontology that are present in a scientific article. As such, it expresses a $Utilization(Ont, Arch)$ coverage type. It was initially difficult to automatically detect semantic similarity, although the assertions were intuitively similar. Synonym Synergy, an enhanced method that uses thesauri to capture subclass relations, improves coverage detection.

The multiplicity, hierarchy, and need for integration of ontologies in the analysis of models and more broadly knowledge bases has been discussed in [56]. During early conceptual design, e.g., of a new aircraft or component, designers may benefit from the integration of semantic search within existing knowledge bases. A two-dimensional framework for classifying ontologies by Generality and Coverage was suggested in order to improve the precision of such semantic search. We found the suggested classification a bit ambiguous: the Coverage axis, which includes Documentation, Modeling, and Design Ontologies, included several confusing overlaps among these ontology types. This may imply that it is difficult to distinguish between ontologies according to their contribution to these activities, as the boundaries among them are blurry to begin with. The authors highlighted the importance of ontology coverage assessment, but did not introduce a solution.

Category-Theoretic Analysis of System Models

Previous studies on robust methods for model analysis found Applied Category Theory and Applied Graph Theory as ideal mechanisms for the generic representation and transformation of models [57]. A graph data structure (GDS) is a highly-robust data representation format that is amenable to a variety of analyses [57]–[59]. Since the GDS is a relational data structure, it is possible to store and analyze it as both a relational database [57], [60] and as a graph database [61], [62].

The Concept→Model→Graph→View Cycle (CMGVC) is a cognitive-computational representation transformation cycle defined in [57], and applied in [58]. The CMGVC utilized Category Theory to capture the different representation methods as categories, and the mappings between representation methods as functors – mappings between categories. Concepts (C) – systems, ontologies, needs, and ideas – are encoded as models in various modeling languages

(M). the graph data structure (G) is an intermediate layer that helps extract information from models. Analytical views (V) are extracted from G rather than from M. This approach yields multiple benefits such as the linearization of the number of transformations from multiple modeling notations to multiple information visualizations, and the abstraction of modeling languages in the process of decision-supporting visualization, insight generation, and concept validation.

A rigorous model-based ontology coverage approach is apparently necessary. Model-based mission architecting, which draws on ontologies, and not only on syntax, is an emerging approach, as evident in the literature. However, the ability to analyze ontological coverage and derive value of such analysis is limited. Particularly, the dual role of ontology coverage in both architectural compliance and ontology utilization must be explored in more depth. Novel model analysis approaches, based on category theory and applied graph theory, enables us to study the ontology coverage problem and provide significant added value to ontology and architecture stakeholders alike.

3. MODEL-BASED ONTOLOGY COVERAGE ANALYSIS (MOCA)

MOCA is a model-based method that provides: i) the overlap between an ontology model and an architecture model, ii) the overlap relative to the set of architectural artifacts, which indicates the architecture's compliance with the ontology, and iii) the overlap relative to the set of ontology stereotypes, which indicates the ontology's utilization by the architecture.

Ontologies and architectures are both, essentially, collections of models. Model-based ontologies are models that include elements that constitute stereotypes. A profile is a model that includes stereotypes. A stereotype is an element that can be applied to elements in other models. An architecture consists of one or more models that describe the structure and behavior of a system. In some FMLs, models can include profiles next to architecture constructs. Table 1 provides a glossary of the primary concepts we use, and our intention when we use them. Interestingly, this glossary can be an ontology for ontology-driven mission architectures, but reflective ontological analysis of MOCA is not our goal.

Stereotyping

We denote ontological concepts or stereotypes using the double-angle quotation marks: «Concept». This convention distinguishes concepts from their instances. We refer to elements defined in the architectural model as architectural artifacts, or *archifacts*. An archifact can be stereotyped. When a stereotype is applied to an archifact, the applied stereotype is recorded in the architecture model as a pointer to the master stereotype. The architectural model must include references to predefined profile models, or packages that constitute profiles. Stereotypes from these profiles can be applied to archifacts. Stereotyping may be subject to syntactical matching: an archifact may assume a stereotype only if there is a match between the syntactical type of the archifact and the syntactical type of the stereotype. This constraint may be overcome by defining stereotypes as the most abstract concept in the language, e.g., the UML Element type, or by removing the prior classification of the archifact.

Stereotyping is a more rigorous mechanism to classify model elements, compared to the embedding of topical keywords in element names. For example, it is wiser to classify a block named 'Obstacle Detection Sensor' as a «Sensor», and optionally rename it to 'Obstacle Detector'. Furthermore, if we have a more fine-grained ontology where specific types of sensors are specified, we can also apply a stereotype like «Object Detector», which may differ, for example, from «Light Sensor», or «Motion Sensor». If a device has multiple roles, it can adopt multiple stereotypes and keep basic name.

Ontology-agnostic or ontology-skeptic engineers may prefer implicit stereotyping through naming. However, reducing the ontological classification problem to textual similarity may seriously hinder architecture analyzability. It may be a good starting point to generate recommended classifications for legacy, ontology-agnostic architecture models.

Extracting Archifacts and Stereotypes from Models

Stereotypes must be classified as stereotypes in the profile models that contain them. They must be compliant with the stereotype retrieval mechanism's logic so that the stereotypes in a profile will appear as classifiers in the model where the profile is used. It is recommended that profiles will not be mixed with architecture models for two reasons: i) to avoid confusion between concepts and instances, and ii) to allow for the reuse of the profile across multiple architectures.

A set of optionally classified model items may be extracted from any model, whether it is an architecture, ontology, or hybrid model. When the model is referred to as part of the ontology, model items that are stereotypes are the ontological reference. When the model is referred to as part of the architecture, model items that are classified as anything but stereotypes (not to be confused with items classified by applied stereotypes) make up the architectural reference.

Calculating the Ontology—Architecture Overlap

The overlap between an architecture A and an ontology Ω, $Z_2(\mathrm{A}, \Omega)$ is defined in **Error! Reference source not found.** as the set of archifacts $\mathrm{A_i}$, such that each $\mathrm{A_i}$ is classified by another architecture item A_j (the applied stereotype), which matched a profile item, Ω_k, which is classified as a stereotype. Accordingly, the uncovered portion of the architecture, $Z_1(\mathrm{A}, \Omega)$, is defined in (4) as the subset or architecture items that are not covered at all, or not covered by applied stereotypes that match items in the reference ontology, or not matched by profile stereotypes. Finally, the uncovered portion of the ontology, $Z_3(\mathrm{A}, \Omega)$, is defined in **Error! Reference source not found.** as the set of stereotypes in the ontology that are not matched with applied stereotypes, or matched with applied stereotypes that are not applied to any archifact.

Table 1. Glossary for a Model-based Ontology Coverage Analysis Framework

Term	Definition	Relation to other terms
architectural artifact (archifact)	object in a model, which has a formal syntactical type in the FML, and semantics of a conceptual, logical, or physical component in a system or mission architecture	An architecture model includes many archifacts.
architecture	conceptual organization of the structure and behavior of a system in order to fulfill its functionality	An architecture includes one or more architectural model.
architecture compliance with R	extent of coverage of an architecture by R, which may be an ontology or another reference (e.g., requirements)	An architecture may partially or fully comply with multiple ontologies
capability	attribute or operation [of someone or something] that represents solution-neutral value delivery	Capabilities enables missions; missions enable capabilities
concept	idea that maps meaning to structure, function to form, or concept to concept	An ontology consists of concepts. An architecture consists of concepts.
graph data structure (GDS)	Set of tuples that represent directed relations between nodes in a graph	A GDS represents a model
mission	set of operational goals to be accomplished by a system, person, or organization	A mission model specifies a mission architecture
model	formal specification of a part of a system, built according to the syntax of a FML, stored as an alphanumeric data structure (e.g., XML file), which represents a concept	A model represents an architecture or an ontology.
ontology	conceptual organization of a domain, based on a set of concepts and relations among them	An ontology consists of one or more profiles.
ontology utilization	extent of coverage of an ontology by an architecture	An ontology may be partially or fully utilized by multiple architectures
profile	model that includes objects that represent concepts, which are specified as stereotypes according to the syntax of the FML	A profile is any model or part of a model where stereotypes are defined
stereotype	object in a profile model, which represents a concept, and which can be applied to other objects in other models (e.g. archifacts) in order to give additional semantics to these objects according to the concept	A stereotype is defined in a profile. A stereotype can have multiple applied stereotypes.
stereotype, applied	object in a model, which points to a stereotype, and serves as the mediating object between a stereotype and archifacts in the model	An applied stereotype maps one stereotype to one archifact

$$Z_2(\mathrm{A},\Omega) = \left\{ \begin{array}{c} \mathrm{A_i}| \\ \exists classification(\langle\langle A_j\rangle\rangle, \mathrm{A_i}), i \neq j \ \wedge \\ \exists MATCH(\langle\langle A_j\rangle\rangle, \langle\langle \Omega_k\rangle\rangle) \ \wedge \\ \exists classificaton(\Omega_k, \langle\langle Stereotype\rangle\rangle) \end{array} \right\} \quad (3)$$

$$Z_1(\mathrm{A},\Omega) = \left\{ \begin{array}{c} \mathrm{A_i}| \\ \nexists classification(\langle\langle A_j\rangle\rangle, \mathrm{A_i}), i \neq j \ \vee \\ \nexists MATCH(\langle\langle A_j\rangle\rangle, \langle\langle \Omega_k\rangle\rangle) \ \vee \\ \nexists classificaton(\Omega_k, \langle\langle Stereotype\rangle\rangle) \end{array} \right\} \quad (4)$$

$$Z_3(\mathrm{A},\Omega) = \left\{ \begin{array}{c} \Omega_k| \\ \nexists classificaton(\Omega_k, \langle\langle Stereotype\rangle\rangle) \vee \\ \nexists MATCH(\langle\langle \Omega_k\rangle\rangle, A_j) \ \vee \\ \nexists classification(\langle\langle A_j\rangle\rangle, \mathrm{A_i}), i \neq j \end{array} \right\} \quad (5)$$

The rules of finding members of Z_1, Z_2, and Z_3 are illustrated in Figure 2: archifacts (rectangles) must be classified by applied stereotypes in the architecture models (flipped trapezoids with lowercase letters), which must be matched with stereotypes in any of the profiles in the ontology (trapezoids with capital letters).

The rules of inclusion in the three zones use the language-agnostic relation patterns: «classification» and «MATCH» (which are denoted as stereotypes because they pertain to an ontology of types of relations in models). Specifications of classification relations in models may not be immediately available in the form $classificaton(X,\mathrm{Y})$. This is FML-specific. The $MATCH(X,\mathrm{Y})$ mapping is defined between models, not within a model, and requires suitable logic. It could be a simple comparison or something more elaborate.

Having defined the zones in the architecture-ontology map, $Z_1(\mathrm{A},\Omega), Z_2(\mathrm{A},\Omega), Z_3(\mathrm{A},\Omega)$ we are able to derive metrics of overlap, compliance, and utilization. The simplest metrics are based on counting the number of items in each zone, as defined in equations (6), (7), and (8), respectively.

$$Overlap(Arch, Ont) = |Z_2| \quad (6)$$

$$Compliance(Arch, Ont) = |Z_2|/(|Z_1| + |Z_2|) \quad (7)$$

$$Utilization(Ont, Arch) = |Z_2|/(|Z_2| + |Z_3|) \quad (8)$$

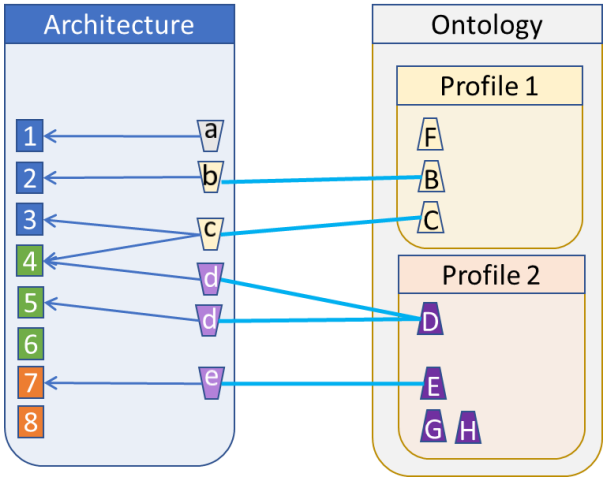


Figure 2. Mapping Applied Stereotypes (flipped trapezoids) of Artifacts (rectangles) in the Architecture (left blue shield) to Stereotypes (trapezoids) in Profiles that Compose the Ontology (shields to the right).

A more sophisticated assessment scheme may assign different importance factors to different architecture elements (e.g., by their connectivity in the architecture) and to different stereotypes (e.g., by their connectivity in the ontology), and weigh each match accordingly. We leave the exploration of such assessment schemes to future research.

Application of MOCA to SysML Models

Applying MOCA requires transition from abstract theory to a specific modeling environment, with existing modeling and analysis capabilities. The application presented here is not the exclusive way for conducting MOCA, and other approaches may be found suitable in the future.

The modeling and analysis process consists of two phases:

- Phase I: Preparation: building an ontology, adopting, or customizing or a given profile for implementation and analysis.
- *Phase II: Application:* building the mission architecture model, applying stereotypes to archifacts, and analyzing its growing compliance using the MOCA algorithm.

Phase I: Building a reference ontology profile model.

We need a SysML profile model for mission architectures. Such a profile must be based on a set of acceptable concepts in the field. For this purpose, we adapted a profile from a conceptual architecting framework that was introduced in [23] and enhanced in [20]. We also included internal jargon elicited from discussions with mission engineering subject matter experts (SMEs) at the MITRE Corporation.

The original conceptual architecture reference model in [20] consists of five domains : D1) Stakeholders, D2) Solution-Neutral Environment, D3) Solution-Specific Environment, D4) Integrated Solution, and D5) Concept of Operations (ConOps). We transformed this set of conceptual domains into a set of UML package stereotypes using the «uml:package» type. A package in UML (and SysML) is a conceptual container that helps organize the model. Packages, like diagrams, are constructs that pertain to the model – not to the architecture it represents. In other words – there are no packages and diagrams in the actual architecture but we use these constructs to help us manage the complexity of the architecture. Accordingly, our profile model consists of five package stereotypes, as summarized in Table 2. The profile diagram is defined in Figure 3.

Table 2. Package Stereotypes for a Conceptual Architecture Profile Model and Contained Stereotypes

Package Stereotype	Original Concept Domain	Stereotypes
«Stakeholders Domain»	D1) Stakeholders	«Stakeholder» «Need»
«Mission Domain»	D5) Concept of Operations	«Mission» «Goal» «Operator»
«Capability Domain»	D2) Solution-Neutral Environment	«Capability» «Operand»
«Tradespace Domain»	D3) Solution-Specific Environment	«SpecificFunctionality» «SpecificForm» «SpecificOperand» «PerformanceAttribute»
«Solution Domain»	D4) Integrated Solution	«Component» «Function»

The «Tradespace Domain» package captures «Tradespace» elements. The architecting process involves the discovery and analysis of solution candidates (that make up the tradespace), consideration of candidate solutions according to «*PerformanceAttribute*» and the selection of the most suitable one to elaborate into a solution architecture. That is also what the original D3 focused on – the derivation of solution-specific function and structure to accommodate the solution-neutral functionality—capability defined in D2.

The «Solution Domain» specifies one or more aspects of the «Solution» is equivalent to the integrated solution concept defined in D4 as the high-level conceptual design that addresses the solution-specific functionality defined in D3.

The «Mission Domain» defines the «Mission», «Goal», and «Operator» stereotypes. Mission artifacts specify missions to be performed, goals to be achieved, and operators: organizations, individuals, and technologies. This clearly maps to ConOps, but underscores the expectation that this domain will include a mission-oriented specification with specific goals and suitable operational configuration in mind.

The «Mission» package has been promoted to be the second one, right after identifying the stakeholders and their needs. The intention was to start with a formal specification of the initial mission architecture, and establish a clear context for the necessary operational and technological capabilities – as

opposed to the situation in which stakeholder needs drive the formulation of solution-neutral functions which are later integrated into an operational architecture, concept, or application. Although the conceptual architecting process is iterative, the order of the first iteration is critical. Therefore, the order of the packages was modified.

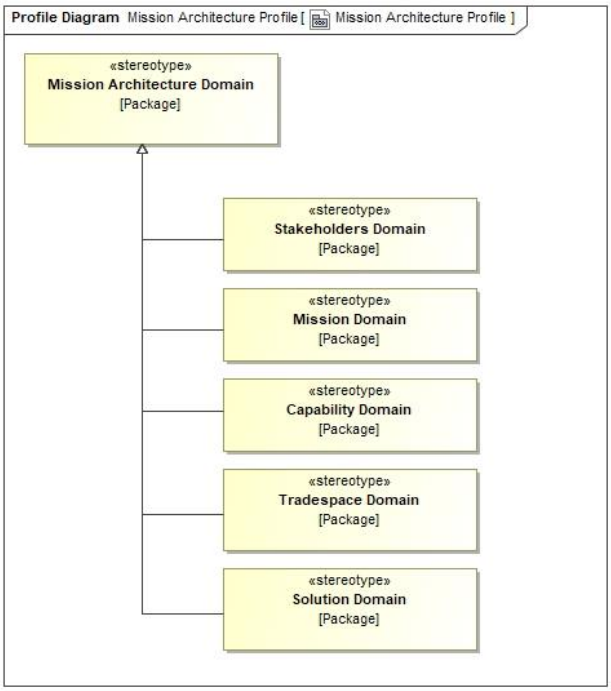


Figure 3. A UML Mission Architecture Profile Diagram defining five conceptual architecture domain packages as stereotypes, derived from the Package type.

The architect may choose to utilize only some packages and some stereotypes, generate multiple instances of specific stereotypes, and specify packages and archifacts that do not adhere to our profile. For instance, a particular mission architecture model may include a Safety package, which is typed «Package» but not stereotyped as any of the above reference packages. This means that it is explicitly or implicitly drawn from another ontology, e.g., an ontology of Systems Safety & Security. Similarly, the architect may define packages with names like "Missions" and "Services". This is an example of implicit stereotyping where the names of the packages imply a terminology that the architect had in mind. It may not be sufficient even one level down the hierarchy where the specification takes shape. For example, packages with the names "Humanitarian Airlift" or "Perimeter Defense" may not be identifiable as the names of a mission or a service (capability) unless they are properly stereotyped as «Mission» and «Capability».

Template models include the instantiated packages with proper stereotyping, which ensures that at least the initial model is stereotyped. An example is shown in Figure 4. This approach resembles template creating for a system specification document or even for this IEEE Aerospace Conference article. While it may help guide the author—modeler about what needs to be included in the specification, there is a caveat: a template model cannot enforce the preservation of headings, items, their order, their internal structure, or their content. It is still a good practice to start with a template model, and ensure through training and usage monitoring that the template is adhered to.

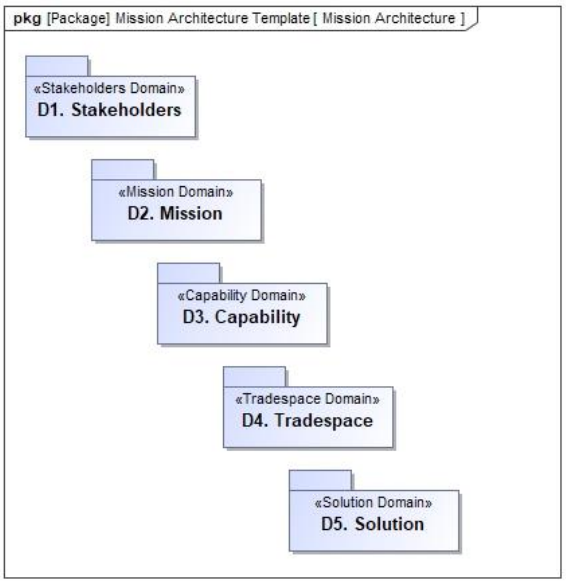


Figure 4. A UML Package Diagram in a Mission Architecture Template Model defining five conceptual packages and stereotyping them according to the corresponding package stereotypes.

Model Transformation

The generic relations we rely on call for a transformation mechanism that can robustly identify and apply mappings within models and between models. This gives rise to the adoption of a category-theoretic, graph-data-based approach.

MOCA is a category-theoretic and graph-based approach: it applies the concepts of transformation from the model category to a graph data structure (GDS) category where it operates on a representation of the model as a set of relation-source-target (RST) tuples. The ontology is also encoded as a model (a *profile*) and then transformed into a GDS. MOCA maps the architectural GDS to the ontological GDS. Thanks to this rigorous approach, MOCA can apply to models from any FML with a valid GDS transform. We work with a relational GDS implementation, which utilizes a SQL Server database and SQL queries to create the necessary datasets for presentation and further analysis.

A CMGV Cycle of MOCA, which extends the general CMGV pattern [57], includes the following mappings:

a) C→M: from ontological and architectural concepts to ontology and architecture models.

b) M→G: from ontology and architecture models to ontology and architecture graphs.

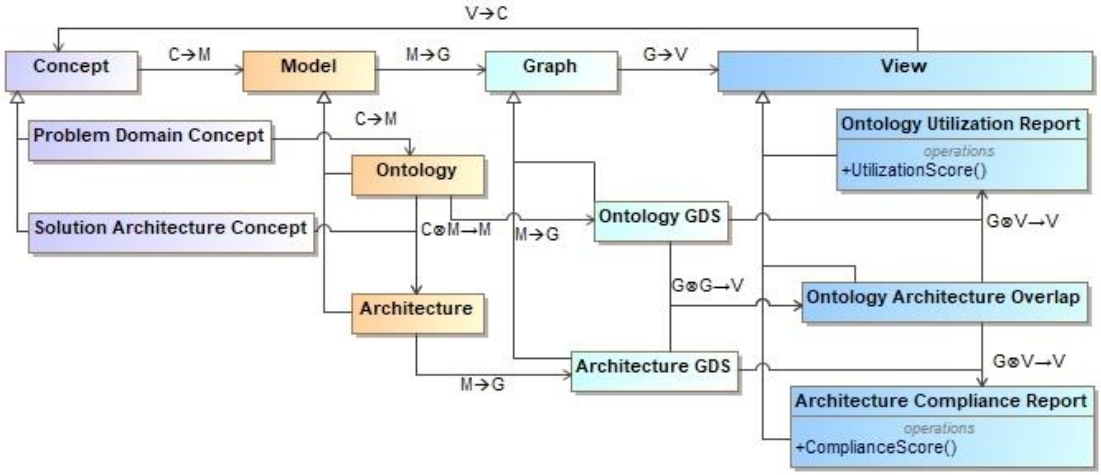


Figure 5. A UML Class Diagram specifying the CMGV Cycle for MOCA: domain ontology and solution architecture (concepts) are encoded as models (M). The models are converted to Graph Data Structures (G). The GDS representations are fused in order to create three views: Ontology-Architecture Overlap (OAO), Ontology Utilization Report (OUR), and Architecture Compliance Report (ACR)

c) G→V: from ontology and architecture graphs to ontology coverage reports.

d) V→C: from ontology coverage reports to concept review or revision, triggering another cycle.

The MOCA CMGV Cycle is illustrated in Figure 5.

C→M: We observe two parallel and intertwining cycles –the ontological and the architectural. The ontology model stems from a problem domain concept, while the architecture model stems from a solution architecture concept and from the ontology model. Thus, the generation of an architecture is in fact a $C \otimes M \rightarrow M$ functorial mapping, where the cartesian product operator $\otimes$ abstracts the fusion of concepts with a reference ontology model (the ontological profile model) to create the architecture model.

M→G: Each model is converted into a GDS, which is stored in a GDS Table in the database. This step relies on existing algorithms to transform models based on their modeling notation to a set of relation-source-target (RST) tuples.

G→V: generating the MOCA views is based on several mappings from G to V. First, the Ontology-Architecture Overlap (OAO) is a cartesian product of two graphs, hence it adheres to the form $G \otimes G \rightarrow V$. This type of fusion of model graphs extends the linear CMGVC model defined in [57]. OAO is fused with the architecture graph to create the Architecture Compliance Report (ACR), and with the ontology graph to create the Ontology Utilization Report (OUR). Both ACR and OUR adhere to the transformation form $G \otimes V \rightarrow V$. ACR is aggregated to create an Architecture Compliance Score (ACS), while OUR data is aggregated to create an Ontology Utilization Score (OUS).

V→C: In the conceptual phase of the cycle, model stakeholders revise the concept and the architecture model according to MOCA results. They may consider the adoption of more stereotypes, replace stereotypes, or even define an underlying ontology for the domain, as a reference for the solution architecture. This is another example of the notion that reasoning about solution architectures may result in a domain ontology that substantiates the architecture.

The algorithm for retrieving OAO, ACR, and OUR is illustrated as a SysML Activity Diagram in Figure 6. The two inputs are the Architecture Tuple Set and Ontology Tuple Set. The outputs include eight sets, of which three are byproducts and five are views.

The algorithm includes the following steps:

1.	Retrieve the archifacts from the architecture GDS.
2.	Retrieve the stereotypes from the profile GDS.
3.	Match the applied stereotypes of archifacts to ontology stereotypes; return the intersection as the Ontology-Architecture Overlap (OAO)
4.	Match the archifacts with OAO; return the item set with/without stereotypes as the Architecture Compliance Report (ACR)
5.	Calculate the proportion of compliant archifacts out of the number of all archifacts as the Architecture Compliance Score (ACS).
6.	Match the stereotypes with OAO; return the item set with/without instances as the Ontology Utilization Report (OUR)
7.	Calculate the proportion of utilized stereotypes out of the number of all stereotypes as the Ontology Utilization Score (OUS).

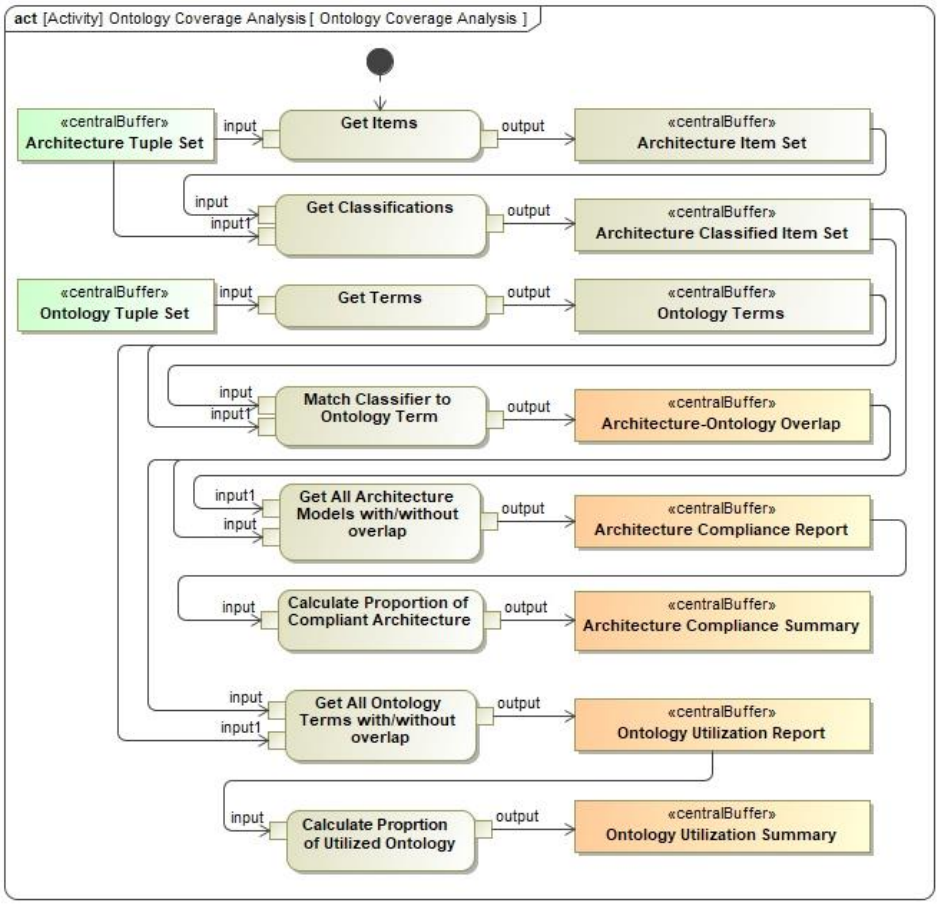


Figure 6. A UML Activity Diagram specifying the MOCA Report Calculation Algorithm. The two input sets at the top left are the GDS tuple sets. The five outputs at the bottom right are the MOCA results.

Phase II: Applying MOCA to a Mission Architecture

Phase II consists of the following steps:

a. Build an ontology-agnostic architecture model.

b. Apply the ontology profile to the architecture model and apply some stereotypes to archifacts.

c. Verify that the ontology-architecture overlap now includes a set of compliant architecture model elements, and that the scores for architecture compliance and ontology utilization are growing.

d. Review the ontology coverage reports. Determine the best way to revise or enhance the architecture.

e. Consider an extension or modification of the ontology to account for absent general concepts [this step is part of the ontological process, rather than the architecting process, but it has potential impact on the architecture].

New concepts may emerge bottom-up from work on a specific architecture. However, it may be impossible to update a given ontology with more concepts that the profile designers did not think of, or decided to exclude from the profile. We may wish to define a separate profile with important emerging concepts as stereotypes that we will be able to apply to our architecture. This will allow for classifying archifacts as part of some terminology, even if only temporarily. We can also use the emerging profile to generalize concepts for other architectures at present or in the future. This step substantiates the ontological aspect of the architect's work and makes it part of a knowledge base building activity that benefits the enterprise. The new stereotypes should not be included in the architecture model, but in a separate profile model that will be available for both the current architecture and other architectures.

The implementation of MOCA is explained and discussed in the context of a specific case model in section 4.

4. CASE STUDY: HUMANITARIAN AIRLIFT

We study a hypothetical humanitarian airlift mission, in which the enterprise is tasked with providing airlift services to a threatened community. Humanitarian airlifts, although not common, have been challenging missions with many operational complexities, risks, military and diplomatic aspects, and a significant operational planning and execution portion, but also a significant technological dimension. Model-based mission engineering can be a key success factor

in the preparation for a humanitarian airlift, particularly regarding those aspects that require the design, development, or deployment of technological solutions to support the mission, when such solutions may not be immediately available, appropriate, or interoperable.

Operation Solomon, May 24-25, 1991, is a famous example of a humanitarian airlift, in which the State of Israel, in coordination with the United States and Russia, airlifted over 14,300 Ethiopian Jews from Addis Ababa to Israel, as Ethiopia was on the brink of civil war (see Figure 7). As we were working on this research and studying the operational and technological characteristics of humanitarian airlifts, in Summer 2021 the United States and its allies terminated their 20 years of presence in Afghanistan and evacuated over 82,000 American nationals, allies, and collaborators amid the Taliban's takeover of the country. The events in Afghanistan suggested a reality check and brought the humanitarian airlift mission back to the front of the stage.

Building an ontology-agnostic architecture model.

We have built an ontology-agnostic SysML model of Humanitarian Airlift as a Use Case Diagram (UCD) where the primary mission and capability includes multiple operational capabilities, as discussed above. The initial, ontology-agnostic model is shown in Figure 8**Error! Reference source not found.**. This model contains no ontological stereotyping, but only syntactical classification by built-in types (block, actor, etc.).

Figure 7. Ethiopian Jews disembarking C-130 Hercules carriers at Ben Gurion International Airport after being evacuated from Ethiopia, May 24-25, 1991 (credit: Israel Government Press Office)

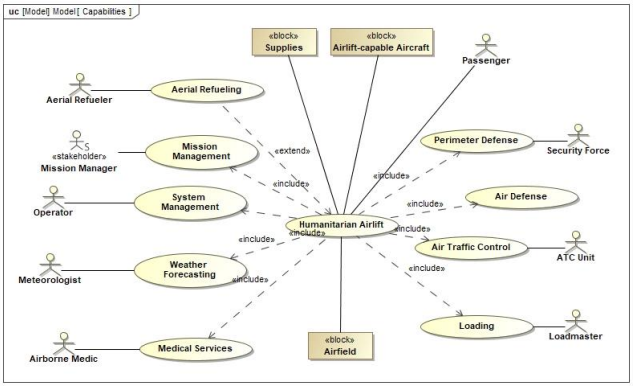


Figure 8. An Ontology-Agnostic UML Use Case Diagram of a Humanitarian Airlift Mission.

The absence of stereotyping in the model causes ambiguity and vagueness about the meaning of model artifacts. For example – we cannot be sure if the UCD defines missions or capabilities, since a use case can describe both, and in fact, to some extent, they are interchangeable concepts. The distinction is in referring to the mission as the process of achieving goals and objectives, while the capability is considered a "black-box" service provided to or by the mission – or more precisely an outcome or effect of a mission that generates the capability. From the capability provider's perspective, performing the necessary activities to generate the outcomes that enable the mission is their mission (e.g., aerial refueling, medical treatment, or perimeter defense).

Humanitarian airlift is intended to transport people and equipment out of a place – not to provide perimeter or air defense to the airfield. This distinction sharpens the understanding of the role of the mission and the clear difference between supporting capabilities and added-value ones. We are already seeing how an ontological discussion shapes our perception of the mission even before we have applied ontological scaffolding to our model.

Applying profile stereotypes to architecture elements.

We now add stereotyping to the SysML model. The profile model must be imported into the architecture model. This step allows for stereotyping archifacts according to profile stereotypes that are valid for the syntactic type of the archifact. For example – package stereotypes can be applied to packages, use case stereotypes like «Capability» and «Mission» can be applied to use cases, and actor stereotypes like «Stakeholder» can be applied to actors. A revised and partially-stereotyped model is illustrated in Figure 9. We have also applied a layout that places enabling capabilities to the left of the mission, and enabled capabilities to the right. Furthermore, we used different association stereotypes for Enabling Capability-Mission and Mission-Enabled Capability relations.

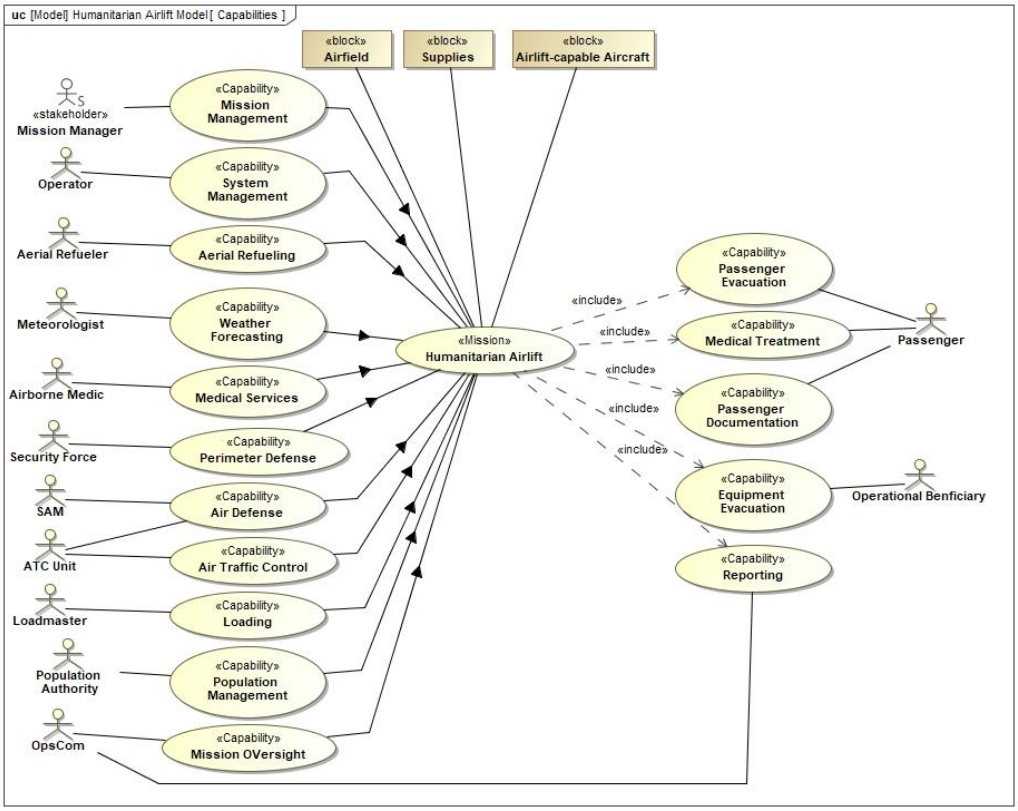


Figure 9. Structured stereotyping of the mission, supporting capabilities, and provided capabilities. Stakeholders, Solution-Neutral Operands, and Architecture packages are not yet stereotyped.

The ACR is shown in **Error! Reference source not found.**. The list of architecture items consists of 35 items in total, using 4 syntactical types (i.e., UML primitive types): «Actor», «Class», «Package», and «Use Case». The syntactical types aid the distinction between architecture resources that may have ambiguous semantics. For example, *Air Traffic Control* is both an organization and a process. In this case, we have clearly defined the *ATC Unit* as an «Actor» and *Air Traffic Control* as a «Use Case». The ACS is the number of stereotyped items divided by the number of items in the architecture: 17/35=~48.5%. The ontology-architecture overlap (OAO), is summarized in the bottom of the ACR. The overlap, architecture compliance, and ontology utilization have started growing. The complete list of ontological concepts, captured as profile stereotypes, can be extracted from the profile model as the set of elements typed as «uml:stereotype». In this example, as we have only utilized 2 out of 21 concepts, the utilization rate is only 9.5%.

5. DISCUSSION

Ontological compliance impacts architectural decisions

MOCA's ACR maps the architecture's compliance with the ontology, and measures compliance as the proportion of stereotyped items out of the architecture. Missing stereotypes are clearly visible in the ACR. The compliance map and score guide the architect to complete, reconsider, or validate the stereotyping, with a quantitative estimate of progress made.

Considering and validating applied stereotypes is based on the bundling and grouping of items under stereotypes and syntactical types. If some items are stereotyped in a way that is inconsistent with the stereotyping of other items with the same syntactical type, it alerts the architect to reconsider the stereotyping. For instance, if an item that is syntactically defined as an «Actor» is stereotyped as a «Capability», while all other actors are stereotyped as «Stakeholder» and all other capabilities are based on «Use Case», this is clearly inconsistent with the modeling convention.

The architect may use the ACR to determine how to iteratively enhance the architecture. For example, the architect may consider stereotyping another syntactical group of artifacts – actors or packages. Another step could be adding items to the architecture, like additional capabilities or resources, while carefully classifying the new items as appropriate. This will prevent the accrual of an *ontological debt*, i.e., a growing number of untyped items that will need to be stereotyped later. Another approach could be to let a colleague analyze the architecture and classify artifacts, as a quality assurance and concept validation step. The colleague may be more proficient with ontological classification and analysis and may use this process to highlight discrepancies in perception and concept specification. We suggest comparing these two approaches in a future study.

In our example, we have gone from zero compliance to almost 50% compliance by stereotyping the use case artifacts,

Table 3. Architecture Compliance Report (ACR) for Humanitarian Airlift Mission Model

Syntactical Type	Profile	Stereotype Name	Archifact	F	T	Total
uml:Actor	(blank)	(blank)	Aerial Refueler	1		1
			Airborne Medic	1		1
			ATC Unit	1		1
			Loadmaster	1		1
			Meteorologist	1		1
			Mission Manager	1		1
			Operational Benficiary	1		1
			Operator	1		1
			OpsCom	1		1
			Passenger	1		1
			SAM	1		1
			Security Force	1		1
uml:Class	(blank)	(blank)	Airfield	1		1
			Airlift-capable Aircraft	1		1
			Mission Commands	1		1
			Supplies	1		1
uml:Package	(blank)	(blank)	Capabilities	1		1
			Stakeholders	1		1
uml:UseCase	Mission Architecture Profile	Capability	Aerial Refueling		1	1
			Air Defense		1	1
			Air Traffic Control		1	1
			Equipment Evacuation		1	1
			Loading		1	1
			Medical Services		1	1
			Medical Treatment		1	1
			Mission Management		1	1
			Mission OVersight		1	1
			Passenger Documentation		1	1
			Passenger Evacuation		1	1
			Perimeter Defense		1	1
			Population Management		1	1
			Reporting		1	1
			System Management		1	1
			Weather Forecasting		1	1
		Mission	Humanitarian Airlift		1	1
Grand Total				**18**	**17**	**35**

distinguishing the «Mission» artifact from the «Capability» ones, and enabling further analysis such as the distinction between enabling and enabled capabilities. This analysis has architectural ramifications as it affects the perception, consideration, and prioritization of mission ingredients according to their impact and criticality to mission success. For example, by asserting that activities like Population Management are critical capabilities delivered to the mission by the appropriate specialists, we shape the mission.

Ontological mindset facilitates abstraction and emergence

The association of multiple enabling capabilities with the mission UC and the decomposition of the mission itself into a set of added-value capabilities highlight the notion that the mission is an abstract concept, which represents the emergence of value from the individual capabilities that compose it. The mission UC provides a compact way of describing how multiple enabling capabilities support multiple added-value capabilities. This abstraction layer means that the «capability»→«mission»→«capability» pattern can translate to an indirect relation «capability» → «capability». We can then derive a capability mapping matrix that shows how each capability enables other capabilities. We can define an interface between two capabilities directly using «directed association» between use cases. We must make sure, however, that mission and capability UCs are different from each other through stereotyping.

Enriching both the architecture and the ontology

The MOCA utilization report shows utilization of ontological concepts (represented as stereotypes) by architectural artifacts. This utilization map measures the proportion of instantiated stereotypes out of the ontology. The ontology utilization report helps both mission architects and ontology architects. Mission architects learn about unutilized and underutilized concepts and can reconsider the conceptual scaffolding of the architecture. Ontology experts can evaluate underutilized concept usability, relevance, validity and understandability, and revise the ontology to better match the needs of the mission architects who are expected to use it.

In our example, the absence of, say, Solution-Neutral Operands from the architecture, calls the architect to consider referring to such transferrable and transformable items in the context of the mission and enrich the architecture by clarifying what inputs are provided to the mission and what outputs are generated by the mission. For instance, considering the steady flow of medication into the airlift perimeter – regardless of the actual means of transporting the medication – may funnel into a discussion on the type of

relevant medication, the storing, issuance, and administering of medication, the disposal of medical waste, and the assessment of medical treatment as part of the humanitarian aspects of the mission. In Operation Solomon, for instance, medical crews delivered newborns on the ground and in midair, and such an activity requires a variety of medical devices and medications for both the mother and the child.

The adapted conceptual architecture ontology did not distinguish enabling from enabled capabilities. We can reduce ambiguity in the model by providing suitable specialized sub-stereotypes of «Capability». We can add these specialized concepts to the base profile or to a dedicated profile. However, we must maintain backward-compatibility and ensure that the new stereotypes are not misused: we should prevent the specifying of enabled capabilities as use cases that feed the mission use case and enabling capabilities as use cases that are included in the mission. This can be achieved with design rules – a topic for additional research.

6. CONCLUSION

Ontology coverage is inherently a dual concept: the ontology is the covering and covered artifact. In this paper we dismantled this duality through the coverage analysis of the ontology by the architecture, and the architecture by the ontology. The coverage of the architecture by the ontology has *compliance* semantics: the portion of the architecture that is covered by a set of concepts indicates the extent of adherence of the architecture to the conceptualization that the ontology suggests. The coverage of the ontology by the architecture has *utilization* semantics: the portion of the ontology that is covered by the architecture indicates the extent of adoption of the suggested conceptualization.

We have discussed the importance of an ontological mindset in MBSE, and particularly in mission models. The scope of generic modeling languages is intentionally minimal and meant to capture the largest possible space of specifiable systems and concepts. Terminological extensibility is inherent in MBSE: SysML is an engineering systems ontology that is implemented as a UML profile. The extended SysML vocabulary is still insufficient for describing basic aspects of complex systems, such as missions and operations, safety and security, or risk and uncertainty. Additional conceptual scaffolding is needed to reduce ambiguity, increase understandability, and facilitate automated analysis, verification, and validation of system models. The appreciation for model-based ontologies in supporting digitalization and scaling-out of MBSE is growing. However, suggesting ontologies is insufficient – we must enforce them.

In this paper, we have integrated several model analytic thrusts into one coherent theme and practical outcome that is meant to facilitate digital systems engineering: ontological analysis, category-theoretic graph-based analysis, and mission-oriented analysis. We have founded our discussion scientifically on the formality of models on the one hand and ontologies on the other hand. We have harnessed robust and rigorous model transformation and analysis methods. We have also weighed in on mission architecting challenges, particularly in a model-based mission engineering mindset.

The convergence of model analytical approaches culminated in the creation of a model-based ontology coverage analysis (MOCA) framework. As shown, running and utilizing MOCA during the system/mission architecting process has significant impacts on the quality, scope, content, and validity of the architecture, as well as critical validation of the ontology as a reference vocabulary of concepts and patterns.

MOCA provides the critical insight that informs stakeholders and decision makers about the architecture's ontological merit. This approach is far more reliable and scalable than visual reviews or assessment schemes that exist next to the model. In turn, MOCA reinforces the model's status as an Authoritative Source of Truth. MOCA is therefore a critical enabler and facilitator of MBSE, MBME, and DE.

This paper discusses one aspect of a broader program for the facilitation of digital mission engineering capabilities. Additional aspects include information visualization through an architect dashboard, integration of outcomes with other forms of analysis, and application of machine learning algorithms to the datasets generated by MOCA and other analysis methods. Future research in the context of MOCA concerns the scalability and performance of MOCA on large-scale mission models and model ensembles in an enterprise environment, the extension of MOCA to the analysis of complex conceptual constructs, starting with concept relations, integration of MOCA as an online design aid, whereas at the moment it is an offline tool, and the empirical study of MOCA's impact on architects and stakeholders' managerial, engineering, and operational decisions.

The set of performance attributes or Figures of Merit (FoMs) for solution architectures: capacity, precision, mass, response time, energy consumption, cost, reliability, etc. – can be encoded as an ontology. Solution attributes must be appropriately matched with stakeholder-defined FoMs and solution FoM values must be equal to or better than stakeholder-required FoM values. FoM value comparison can be done solution-wise or attribute-wise. In future research, we plan to use MOCA for architecture assessment, by comparing ontologically-classifiable architecture FoMs to their reference performance assessment ontologies.

ITAR Compliance.

The authors state that this paper does not violate the International Traffic in Arms Regulations (ITAR).

Organizational Approval

The MITRE Corporation has approved the publication of this article in the IEEE Aerospace Conference 2022 and in the IEEEXplore Conference Proceedings.
The MIT-affiliated authors are qualified to determine that this paper is suitable for publication.

ACKNOWLEDGEMENTS

This research was sponsored by the MITRE Corporation under Grant Agreement No. 136887. The authors thank Michael L. Curry from MITRE for his useful comments.

REFERENCES

[1] Object Management Group (OMG), *OMG Unified Modeling Language (OMG UML)*, Version 2. Needham, MA: OMG, 2015.

[2] Object Management Group, "OMG Systems Modeling Language Version 1.6," Needham, MA, USA, 2019.

[3] S. A. White, "Introduction to BPMN," 2004.

[4] D. Dori, *Model-Based Systems Engineering with OPM and SysML*. New York: Springer, 2016.

[5] D. Dori, *Object-Process Methodology: A Holistic Systems Approach*. Berlin, Heidelberg: Springer-Verlag, 2002.

[6] J. L. Peterson, "Petri Nets," *Comput. Surv.*, vol. 9, no. 3, pp. 223–252, 1977.

[7] T. Murata, "Petri Nets : Properties , Analysis and Applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[8] P. Beery and E. Paulo, "Application of model-based systems engineering concepts to support mission engineering," *Systems*, vol. 7, no. 3, 2019.

[9] D. Zwemer, "Integrated Model-Based Mission Engineering | Part 1," 2021. [Online]. Available: https://intercax.com/2021/01/07/integrated-model-based-mission-engineering-part-2/. [Accessed: 29-Apr-2021].

[10] J. Dahmann, "Mission Engineering: An Implementation Approach," in *22nd Annual National Defense Industrial Association Systems and Mission Engineering Conference*, 2019.

[11] P. Zimmerman and J. Dahmann, "Digital Engineering Support to Mission Engineering," in *21st Annual National Defense Industrial Association Systems and Mission Engineering Conference*, 2018.

[12] J. P. Hale *et al.*, "Digital Model-based Engineering: Expectations, Prerequisites, and Challenges of Infusion," NASA, Hampton, VA, NASA/TM—2017–219633, 2017.

[13] P. Zimmerman, T. Gilbert, and J. Dahmann, "Extending the DoD Digital Engineering Strategy to Missions, Systems of Systems, and Portfolios," in *22nd Annual National Defense Industrial Association Systems and Mission Engineering Conference*, 2019.

[14] United States Department of Defense (DoD), "The DoDAF Architecture Framework Version 2.02," 2010. [Online]. Available: https://dodcio.defense.gov/Library/DoD-Architecture-Framework/. [Accessed: 03-Dec-2020].

[15] J. Colombi, John M.; Miller, Michael E.; Schneider, Michael; McGrogan, Jason; Long, David S.; Plaga and C. Piaszczyk, "Model Based Systems Engineering with Department of Defense Architectural Framework," *Syst. Eng.*, vol. 14, no. 3, pp. 305–326, 2011.

[16] Office of the Under Secretary of Defense for Research and Engineering, *Mission Engineering Guide*. Washington, DC, USA, 2020.

[17] B. E. Strom, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK (R): Design and Philosophy," 2018.

[18] N. G. Leveson, *Engineering a Safer World*. Cambridge, MA, USA: MIT Press, 2011.

[19] D. P. Pereira, C. Hirata, and S. Nadjm-Tehrani, "A STAMP-based ontology approach to support safety and security analyses," *J. Inf. Secur. Appl.*, vol. 47, pp. 302–319, 2019.

[20] Y. Menshenin, Y. Mordecai, E. F. Crawley, and B. G. Cameron, "Model-Based System Architecting and Decision-Making," in *Handbook of Model-Based Systems Engineering*, A. M. Madni and N. Augustine, Eds. Springer, 2021.

[21] D. Orellana and W. Mandrick, "The Ontology of Systems Engineering: Towards a Computational Digital Engineering Semantic Framework," *Procedia Comput. Sci.*, vol. 153, pp. 268–276, 2019.

[22] L. Yang, K. Cormican, and M. Yu, "Ontology-based systems engineering: A state-of-the-art review," *Comput. Ind.*, vol. 111, no. October, pp. 148–171, 2019.

[23] Y. Menshenin and E. Crawley, "A system concept representation framework and its testing on patents, urban architectural patterns, and software patterns," *Syst. Eng.*, vol. 23, no. 4, pp. 492–515, 2020.

[24] A. Renault, "A Model for Assessing UAV System Architectures," *Procedia Comput. Sci.*, vol. 61, pp. 160–167, 2015.

[25] A. M. Ross, D. E. Hastings, J. M. Warmkessel, and N. P. Diller, "Multi-Attribute Tradespace Exploration as Front End for Effective Space System Design," *J. Spacecr. Rockets*, vol. 41, no. 1, pp. 20–28, 2004.

[26] I. Hadar and P. Soffer, "Variations in Conceptual Modeling: Classification and Ontological Analysis," *J. Assoc. Inf. Syst.*, vol. 7, no. 8, pp. 568–592, 2006.

[27] P. Soffer and I. Hadar, "Applying ontology-based rules to conceptual modeling: A reflection on modeling decision making," *Eur. J. Inf. Syst.*, vol. 16, no. 5, pp. 599–611, 2007.

[28] N. Guarino, "Ontology-Driven Conceptual Modelling." Italian National Research Council, Laboratory of Applied Ontology, 2005.

[29] Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, 1993.

[30] C. Partridge, C. Gonzalez-Perez, and B. Henderson-Sellers, "Are conceptual models concept models?," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8217 LNCS, pp. 96–105, 2013.

[31] J. P. van Gigch, *System Design Modeling and Metamodeling*, vol. 1. 1991.

[32] J. P. van Gigch, "Metamodeling: The epistemology of system science," *Syst. Pract.*, vol. 6, no. 3, pp. 251–258, 1993.

[33] R. Lukyanenko, K. R. Larsen, J. Parsons, D. Gefen, and R. M. Mueller, "Toward creating a general ontology for research validity," in *CEUR Workshop Proceedings*, 2019, vol. 2469, no. November, pp. 133–137.

[34] Y. Mordecai, C. Chapman, and D. Dori, "Conceptual Modeling Semantics for the Physical-Informatical Essence Duality Problem," in *IEEE International Conference on Systems, Man, and Cybernetics - SMC*, 2013.

[35] Y. Mordecai, "Model-based protocol specification," *Syst. Eng.*, vol. 22, no. 2, 2019.

[36] D. Dori and I. Reinhartz-Berger, "An OPM-based metamodel of system development process," in *Lecture Notes in Computer Science*, vol. 2813, Springer Berlin Heidelberg, 2003, pp. 105–117.

[37] Object Management Group, *Unified Architecture Framework Profile (UAFP) Version 1.1*, no. April. Object Management Group (OMG), 2020.

[38] D. Ernadote, "An ontology mindset for system engineering," *1st IEEE Int. Symp. Syst. Eng. ISSE 2015 - Proc.*, pp. 454–460, 2015.

[39] L. Favre, *UML and the Unified Process*. IRM Press, 2003.

[40] M. Hause, "The Unified Profile for DoDAF/MODAF (UPDM) enabling systems of systems on many levels," in *2010 IEEE International Systems Conference*, 2010, pp. 426–431.

[41] F. Dandashi and M. C. Hause, "UAF for System of Systems Modeling," in *10th System of Systems Engineering Conference (SoSE)*, 2015, pp. 199–204.

[42] Y. Mordecai and D. Dori, "I5: A Model-Based Framework for Architecting System-of-Systems Interoperability, Interconnectivity, Interfacing, Integration, and Interaction," in *INCOSE International Symposium*, 2013, vol. 23, pp. 1234–1255.

[43] P. Burek, H. Herre, and F. Loebe, "Ontological analysis of functional decomposition," *Proc. 8th Int. Conf. New Trends Softw. Methodol. Tools Tech. SoMeT_09*, 2009.

[44] Y. Mordecai and D. Dori, "Model-Based Operational-Functional Unified Specification for Mission Systems," in *10th Annual IEEE International Systems Conference (SysCon)*, 2016.

[45] M. Noguera, M. V. Hurtado, M. L. Rodríguez, L. Chung, and J. L. Garrido, "Ontology-driven analysis of UML-based collaborative processes using OWL-DL and CPN," *Sci. Comput. Program.*, vol. 75, no. 8, pp. 726–760, 2010.

[46] J. Bermejo-Alonso, C. Hernandez, and R. Sanz, "Model-based engineering of autonomous systems using ontologies and metamodels," *IEEE Int. Symp. Syst. Eng.*, 2016.

[47] J. F. Castet *et al.*, "Ontology and modeling patterns for state-based behavior representation," *AIAA Infotech Aerosp.*, 2015.

[48] J. C. Day, K. Donahue, M. Ingham, A. Kadesch, A. K. Kennedy, and E. Post, "Modeling off-nominal behavior in SysML," *AIAA Infotech Aerosp. Conf. Exhib. 2012*, pp. 1–10, 2012.

[49] D. A. Wagner, M. B. Bennett, R. Karban, N. Rouquette, S. Jenkins, and M. Ingham, "An ontology for state analysis: Formalizing the mapping to SysML," *IEEE Aerosp. Conf. Proc.*, no. February 2015, 2012.

[50] S. J. Herzig, D. Velez, B. Nairouz, B. Weatherspoon, R. Tikidjian, and B. Muirhead, "A Model-based Approach to Developing the Concept of Operations of the Proposed Mars Sample Return Mission," *2018 AIAA Sp. Astronaut. Forum Expo.*, no. September, pp. 1–15, 2018.

[51] N. A. Stanton, G. Walker, D. Jenkins, P. Salmon, M. Young, and A. A. Aujla, "Models of Command and Control," in *Engineering Psychology and Cognitive Ergonomics*, vol. LNAI 4562, no. July, D. Harris, Ed. Springer-Verlag Berlin Heidelberg, 2007.

[52] S. Jenkins, "Introduction to System Modeling and Ontologies," Pasadena, CA, 2011.

[53] Y. Mordecai, N. K. James, and E. F. Crawley, "Object-Process Model-Based Operational Viewpoint Specification for Aerospace Architectures," in *IEEE Aerospace Conference Proceedings*, 2020.

[54] R. Mohamad and N. Mohd-Hamka, "Similarity algorithm for evaluating the coverage of domain ontology for semantic Web services," *2014 8th Malaysian Softw. Eng. Conf. MySEC 2014*, no. September, pp. 189–194, 2014.

[55] N. DiGiuseppe, L. C. Pouchard, and N. F. Noy, "SWEET ontology coverage for earth system sciences," *Earth Sci. Informatics*, vol. 7, no. 4, pp. 249–264, 2014.

[56] J. Lehmann, M. Shamiyeh, and S. Ziemer, "Towards integration and coverage assessment of ontologies for knowledge reuse in the aviation sector," in *SEMANTiCS Workshop Proceedings: LIDARI*, 2017.

[57] Y. Mordecai, J. P. Fairbanks, and E. F. Crawley, "Category-Theoretic Formulation of the Model-Based Systems Architecting Cognitive-Computational Cycle," *Appl. Sci.*, vol. 11, no. 4 (1945), 2021.

[58] Y. Mordecai and E. F. Crawley, "Conceptual State Analysis," in *Complex Adaptive Systems (CAS)*, 2021.

[59] U. Shani, S. Jacobs, N. Wengrowicz, and D. Dori, "Engaging ontologies to break MBSE tools boundaries through semantic mediation," in *Conference on Systems Engineering Research*.

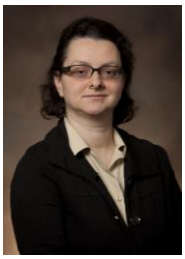
[60] D. I. Spivak and R. E. Kent, "Ologs: A categorical framework for knowledge representation," *PLoS One*, vol. 7, no. 1, 2012.

[61] Jim Webber and R. Van Bruggen, *Graph Databases*, Neo4j Spec. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2020.

[62] D. Medvedev, U. Shani, and D. Dori, "Gaining Insights into Conceptual Models: A Graph-Theoretic Querying Approach," *Appl. Sci.*, vol. 11, no. 2, p. 765, 2021.

BIOGRAPHY

Yaniv Mordecai *(SM'18) is a post-doctoral researcher at the Massachusetts Institute of Technology. He holds a Ph.D. in information systems engineering from Technion – Israel Institute of Technology, Israel (2016), and M.Sc. (2010, cum laude) and B.Sc. (2002) degrees in industrial engineering & management from Tel-Aviv University, Israel. His research interests include model-based systems engineering, model analytics, cybernetics, interoperability, and operations research. Dr. Mordecai is a senior member of IEEE and chairperson of the IEEE Systems Council's Israel Chapter. He won the IEEE Systems, Man, and Cybernetics Society Outstanding Ph.D. Diploma Award in Systems Science and Engineering (2017) and the OmegaAlpha Association's Exemplary Systems Engineering Dissertation Award.*



Aleksandra Markinda-Khusid *received the B.S. degree in physics, M.S. and Ph.D. degrees in electrical engineering, and the M.S. degree in engineering and management, all from the Massachusetts Institute of Technology, Cambridge, MA, USA 1999, 2001, 2005, and 2015 respectively. She leads the Systems and Mission Analysis Department, MITRE Corporation, McLean, VA, USA. Her research interests include analytical and quantitative systems engineering and mission engineering, including systems of systems engineering, tradespace analysis, and decision support as enabled by the modern digital engineering approaches.*

Greg Quinn *is a Principal Software Systems Engineer with more than 20 years of experience, including 13 years at MITRE. Mr. Quinn's current work is on the Standard Health Record for Oncology project, which is targeting the ability to capture structured treatment data for all cancer patients, to support a learning health system for cancer. Greg leads the Flux Notes task that demonstrates the low burden and incentivized collection of high-quality treatment data at the point of care.*

Edward F. Crawley *is the Ford Professor of Engineering, and a Professor of Aeronautics and Astronautics at MIT. He served as the founding President of the Skolkovo Institute of Science and Technology (Skoltech) in Moscow, founding Director of the MIT Gordon Engineering Leadership Program, Director of the Cambridge (UK) MIT Institute, and Head of the Department of Aeronautics and Astronautics at MIT. Dr. Crawley is a Fellow of the AIAA and the Royal Aeronautical Society (UK), a member of the International Academy of Astronautic, and a member of five national academies: Sweden, UK, China, Russia, and USA. He received an S.B. (1976) and an S.M. (1978) in aeronautics and astronautics and a Sc.D. (1981) in aerospace structures from MIT, and has been awarded two degrees of Doctor Honoris Causa. Prof. Crawley's research focuses on the architecture, design, decision support and optimization in complex technical systems subject to economic and stakeholder constraints.*