SSC23-WI-02

Automating Spacecraft Analysis: The Era of Ontological Modeling & Simulation

Michael Halvorson
University of Alabama in Huntsville, Complex Systems Integration Lab
Wernher von Braun Research Hall, 301 Sparkman Drive, Huntsville, AL 35899; 334-300-8131
mch0043@uah.edu

Noah Moyers, Carlos Domani, Johny Corona, Matthew Gillis, Sean Price, Aiden Raley, Andrew Crudup, Andrew Walter, Coleman Deschepper, Grant Gravitt, Jacob Wilkins, Nik Dreicer, Seth Porter, Tyler Babinec, Tyler Johnson, Nicolas Tsolas
Auburn University, 354 War Eagle Way, Auburn, AL 36849
ncm0034@auburn.edu, cad0087@auburn.edu, jcc0126@auburn.edu, mwg0017@auburn.edu, sep0049@auburn.edu, acr0068@auburn.edu, agc0049@auburn.edu, amw0158@auburn.edu, cjd0050@auburn.edu, gmg0017@auburn.edu, jlw0102@auburn.edu, nsd0013@auburn.edu, scp0055@auburn.edu, tdb0040@auburn.edu, tsj0010@auburn.edu, ntsolas@auburn.edu

L. Dale Thomas
University of Alabama in Huntsville, Complex Systems Integration Lab
Wernher von Braun Research Hall, 301 Sparkman Drive, Huntsville, AL 35899
ldt0001@uah.edu

ABSTRACT

Verification by analysis is a predicted compliance of a design to imposed requirements. The levels of performance specified by capability requirements can be related to Technical Performance Measures (TPM) in a Model-Based Systems Engineering (MBSE) environment, but discipline engineers performing verification by analysis are not commonly versed in professional Systems Engineering (SE) techniques or modeling languages such as SysML. As the formal application of Systems Engineering (SE) results in a diminution of time, effort, and money for large-scale projects, the enablement of discipline engineers to contribute MBSE-supported content while performing verification by analysis is financially incentivized. Ontologies applied to technical analysis methodologies are shown to improve the quality of verification by analysis activities while adhering to professional organization standards such as the International Council on Systems Engineering (INCOSE) SE Handbook and the National Aeronautics and Space Administration (NASA) standard 7009A: Standard for Models and Simulations. Organization strategies for information pertinent to verification by analysis are provided using object-oriented methods in both natural and formal languages, and software capabilities for creating the object-oriented relationships in diagram format are described specific to the Engineering Management Platform for Integration, Realization, and Execution (EMPIRE).

NOMENCLATURE

Variables

A	Area	[m^2]	$\bar{x}_{e-cs}$	Earth to Satellite Vector	[km]
m	Mass	[kg]	$\bar{x}_{s-cs}$	Sun to Satellite Vector	[km]
P	Power	[W]	ε	Emissivity	[-]
$\dot{Q}$	Heat	[W]	θ	Angle	[rad]
$\dot{Q}''$	Heat Flux	[W/m^2]	ξ	Solar Zenith angle	[rad]
t	Time	[s]	ρ	Reflectivity	[-]
$\hat{\bar{x}}_{cs,X/Y/Z+/-}$	Unit Vector Orthogonal to Spacecraft Face				

Recurring Subscripts

alb	Denotes Radiative Albedo
AOI	Denotes per Angle of Incidence
cs	Denotes Whole-Satellite (CubeSat)
CPH	Denotes Charged Particle Heating
Earth	Denotes Earth as Source
ems	Denotes Radiative Emission
f, #f	Denotes per Face
FMH	Denotes Free Molecular Heating
GCR	Denotes Galactic Cosmic Rays
in	Denotes Inward Directionality
proj	Denotes Projection
SEP	Denotes Solar Energetic Particles
solar	Denotes Sun as Source
#t	Denotes Parameter Size per Timestep
TE	Denotes Trapped Electrons
TES	Denotes per Thermal Environment State
TP	Denotes Trapped Protons

INTRODUCTION

Analysis, demonstration, inspection, and test are the four verification techniques used to determine the compliance of a small satellite's design to imposed requirements[1]. Verification by analysis, a predicted compliance to requirements via Modeling & Simulation (M&S)[2], is advantageous when expected outcome precision is known, the risk of undiscovered problems can be tolerated, and testing cannot be done credibly or feasibly[3]. Because analysis is performed when desired requirements compliance data is not made available by vendors, cannot be visually acquired, and does not present during use of the system element, analysis is selected if all other verification methods are unviable[3]. Despite analysis being a last resort for verification, it is often the earliest task performed by discipline engineers, particularly students in university satellite programs[4]. Students are commonly asked to develop models, meaning physical, mathematical, or otherwise logical representations of a system, entity, phenomenon, or process[5]. When considered in the context of verification by analysis, models are either misleading, defined at an inappropriate level of abstraction, or misrepresentative of the system or environment, but some are useful[6]. If engineers develop a model that lacks utility, they have both wasted time and possibly delayed the project launch schedule. As M&S is often the earliest technical development responsibility and inexperienced engineers commonly perform M&S within the university-class satellite community, the automation of M&S should be a collective goal resulting in a diminution of time, effort, and money spent during early spacecraft development[7]. Methods for automating M&S using advanced Model-Based Systems Engineering (MBSE) techniques are applied here to an updated version of the spacecraft thermal control strategy presented in Halvorson et al[8]. Ontologies are described first, followed by an overview of ontological parameter evaluation. Parameter evaluation techniques are visualized using natural language and ontological methods. Method realization in an open source software platform is then considered.

Relevant MBSE Theory

The four pillars of applied MBSE are architecture frameworks, process frameworks, modeling languages, and ontologies[9,10]. Applying architecture and process frameworks to verification by analysis is outside the scope of this work, but applying modeling languages and ontologies is fundamental to M&S automation. Any professional M&S automation paradigm must align with and logically build upon existing theory to provide utility without a need for systematic reinvention; relevant M&S theory is recently described by the Hatley-Pirbhai (H-P) method[11,12,13], the International Council for Systems Engineering (INCOSE) Systems Engineering (SE) Handbook[14], National Aeronautics and Space Administration (NASA) standard 7009A: Standard for Models and Simulations[15], and International Organization for Standardization (ISO) 42010:2022, Software, Systems, and Enterprise – Architecture Description[16]. MBSE theory is contextualized here for M&S prior to ontological M&S exemplification to ensure automated verification by analysis relates performance measures to requirements verified by analysis, the validated purpose of M&S[1].

ONTOLOGY PURPOSE AND STRUCTURE

Ontologies, one of the four pillars of MBSE[9,10], are models of reality that taxonomically describe concepts and relations with increasing levels of specificity[17,18,19]. Top-Level Ontologies (TLO) are domain neutral in that they describe a maximally broad range of concepts; they facilitate interoperability of lower-level ontologies specific to a given subject or domain[17,18,19]. Mid-Level Ontologies (MLO) are supported by a TLO but are not domain-neutral. MLOs describe physics-based, mathematical, or equally generic concepts applicable to a broad range of domains to facilitate interoperability of Domain Ontologies (DO)[18,19]. DOs describe concepts specific to a given domain and are supported by one or more MLOs.

The use of ontologies to model object-oriented data has exploded in recent years, garnering interest from academic, industrial, and government entities to organize and populate relational databases, automate data processing, and train artificial intelligence models such as large language models[20]. While there is little agreement in the MBSE community on the best way to build and use ontologies, the combination of advanced M&S techniques, ontologies, and the burgeoning field of Model-Based Project Management (MBPM) offers a promising means to fast-track technology development.

The application of ontological methodologies to verification by analysis is contingent on rigorously defined data structures and standardized rule sets that enable determination of the ontology's logical consistency, inference of implicit relationships among taxonomized concepts, and methodical querying of the knowledge represented in an ontology. Inferences and detection of logical inconsistencies are made possible via the use of semantic reasoners that evaluate the assertions made in an ontology[21]. Querying for information stored in an ontology can be performed by using a compatible query engine with a corresponding query language[22,23] or by directly accessing the contents of an ontology using an Application Programming Interface (API) designed for Object-Oriented Programming (OOP)[24]. Ontology APIs designed for OOP allow for automated execution of complex interdependent reasoning tasks and querying of information, capabilities critical to the automation of M&S.

Formal Languages

Ontologies bear a machine-readable structure authored using an ontology language, a formal language designed for ontological modeling. The machine-readability of concepts described using a formal language is dependent upon the language syntax, the structure of an expression in a given language[25], and semantics, the ascription of meaning to syntactic elements of a language[25,26], but syntax is colloquially a conflation of the terms abstract syntax, concrete syntax, and grammar, related terms that carry distinct implications for language use[25]. Abstract syntax refers to a metamodel representation of the conceptual components of a language in terms of a vocabulary[27] independent of any encoding scheme[26]. Concrete syntax refers to a set of terminal symbols, the most primitive elements of a language, and non-terminal symbols, primitive constructs of a language[26,27]. Grammar refers to a set of rules for a language that recursively define all valid combinations of terminal and non-terminal symbols[27]. Formal languages may have more than one concrete syntax mapped to its abstract syntax[25,26].

Foundational Data Model

The Resource Description Framework (RDF) facilitates the relation and machine-readable representation of abstract and physical concepts, declared as resources in RDF parlance[28,29]. The RDF abstract syntax defines a data structure called a RDF triple or a semantic triple that allows for the assertion of a relationship from a subject to an object via a predicate. A visualization of the structure of a RDF graph containing two nodes is shown in Figure 1.

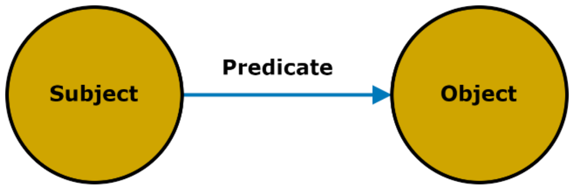


Figure 1: RDF Graph Structure

Resources cannot directly assume any position in a RDF graph and must be denoted by one of the following: an Internationalized Resource Identifier (IRI), a RDF literal, or a blank node[29]. IRIs can assume any position, RDF literals can only assume the object position, and blank nodes can only assume the subject or object positions.

An IRI is a uniquely identifying string in a global context known as a global identifier that denotes a specific resource. In contrast, a blank node is a local identifier that RDF enforces no internal representation of[29], merely denoting the existence of an unnamed resource. A RDF literal is a RDF graph that relates a lexical representation of a value to a data type denoted by an IRI and, if required by the data type, a language tag[29]. A RDF data type consists of a lexical space mapped to a value space such that one or more elements in the lexical space map to a single element in the value space as defined by the Extensible Markup Language (XML) schema[30]. To elaborate, consider the literals "+1.0" and "1.00" defined with a data type of decimal, a concrete numeric data type in XML with a lexical space consisting of any real number representable in base-10[30]. Despite the differing representation of these literals in the lexical space, both map to an identical value of "1" in the associated value space.

Ontology Languages

The RDF abstract syntax is combined with a data-modeling vocabulary and semantics framework from the RDF Schema (RDFS) to serve as the foundation for the Web Ontology Language (OWL)[31]. RDFS is a general-purpose language that can be used as an ontology language to facilitate the representation of RDF concepts[32]. RDFS expands on the concept of a RDF resource by defining a class as an instantiable and potentially hierarchical categorization of RDF resources such that the members of a class are its subclasses, instances, and instances of its subclasses. RDFS also allows for the attribution of domain restrictions, a specification of allowed classes of subjects, and range restrictions, a specification of allowed classes of objects, directly to a RDF property to enforce the intended usage of a given property. RDFS can be used to define the structure of an ontology but on its own lacks sufficiently rigorous semantics necessary to support complex reasoning tasks[31,32].

OWL is an ontology language developed as an extension of RDFS that provides a richer data-modeling vocabulary and corresponding semantics necessary to facilitate enhanced machine-readability for complex reasoning tasks[31]. OWL expansively extends the expressivity of RDFS and is subdivided into three increasingly expressive profiles: OWL Lite, OWL Description Logic (DL), and OWL Full. OWL Lite defines the core vocabulary and semantic extensions for both OWL DL and OWL Full. OWL Full allows for augmentations to the predefined RDFS and OWL vocabulary thereby creating potential for reasoning tasks to produce indeterminate results. OWL DL disallows modifications to the predefined RDFS and OWL vocabulary, among other less notable restrictions, to support deterministic reasoning of ontological structures[33,34]. OWL is superseded by OWL 2, a further extension of the data-modeling vocabulary and associated semantics of RDFS that remains backward compatible with the original version of OWL[33,35]. Despite OWL 2 remaining backward compatible, certain features introduced by OWL have been deprecated and replaced in OWL 2[36]; only relevant, non-deprecated OWL 2 DL features are discussed for the sake of brevity.

OWL 2 expands upon the concept of a RDFS property by adding annotation property, object property, and data property as specializations. An annotation property is used to ascribe metadata, meaning data about data, to a class, individual, or property to facilitate human-readability rather than machine-readability such that metadata is to be ignored by a semantic reasoner[36]. An object property is used to assert a semantic relationship between two individuals in the form of an object property expression, and a data property is used to assert a semantic relationship between an individual and a RDF literal in the form of a data property expression. A class expression must be used to assert a semantic relationship at the class level rather than at the individual level such that all individuals of involved classes are bound by said assertion[36]. Usage of a class expression allows for a restriction to be placed on an object property expression or a data property expression relating to existential quantification, universal quantification, or cardinality of the object in a semantic triple. An existential class expression is used to relate a class of subjects “A” to a class of objects “B” and assert that an instance of “A” must be related via a specific object property to a minimum of one instance of “B”. An existential class expression is used to relate a class of subjects “A” to a class of objects “B” and assert that an instance of “A” cannot be related via a specific object property to an instance of a class other than “B”. A class expression containing a minimum, exact, or maximum cardinality restriction is used to relate a class of subjects “A” to a class of objects “B” and assert that an instance of “A” must be related via a specific object property to greater than or equal to, exactly, or less than equal to a specific number of instances of “B” respectively. A more complex form of class expression is often necessary to enhance inference capabilities and can be formulated using the herein described concepts and the Boolean connectives “and”, “or”, and “not”, referred to as conjunction, disjunction, and negation respectively in the parlance of logisticians[35]. Specific examples of class expressions are provided later in this work.

The capabilities for deterministic semantic inference enabled by OWL 2 DL are extensive but limited in that minimum cardinality restrictions, including those explicitly asserted in a class expression or implicitly asserted in an existential class expression, are ignored by reasoners compliant with the open-world semantics of language[36]. During the reasoning process, the application of open-world semantics results in the assumption that information not declared is simply considered to be unknown rather than indicative of a logical inconsistency[36]. If asserted minimum cardinality restrictions are to be relied upon for detecting logical inconsistencies due to missing information in ontological structures pertinent to M&S, a reasoner that applies closed-world semantics must be used.

The Problem with Purposeful Ontologies

Ontologies and analysis models have a similar, ironic problem. Despite ontologies being designed for reuse, there are few reported examples of existing ontologies being used by groups who did not build them[37,38]. A diversity of domain-specific content representation styles with incompatible levels of abstraction and narrowly prescribed purposes limit the utility of both models and ontologies. Maier notes a substantial challenge to modern Digital Engineering (DE) is, “the sheer diversity of models, notations, tools, and methods, many of them overlapping and redundant[13].” As an example, a hierarchy of ontologies including the Basic Formal Ontology (BFO) as the TLO[39] and the Common Core Ontologies (CCO) as a suite of MLOs[40] were used as the basis for a set of Space Domain Ontologies

(SDO)[41] describing outer space, space events, space objects, spacecrafts[41], and spacecraft missions. The Marshall Space Flight Center (MSFC) Advanced Concepts Office (ACO) evaluated the BFO, CCO, and SDO for inclusion in an upcoming set of space-focused, NASA-centric DOs called the Common Space Systems Ontology (CSSO) and determined the relation formality and level of abstraction in the BFO, CCO, and SDO did not fully meet the ACO's needs, though this work is ongoing[43]. Separately, the BFO and Information Artifact Ontology (IAO) were evaluated as a basis for a DO concerning Prognostic Health Management (PHM) in spacecraft avionics by Halvorson, Moyers, and Thomas[42], but the BFO independent continuant division of material versus immaterial entities does not coincide with how the standard model of physics organizes relations between matter and energy, making it difficult to represent interactions of spacecraft electronics with subatomic particles. The BFO 2.0 User's Guide[44] states,

> Matter is intended to encompass both mass and energy (we will address the ontological treatments of portions of energy in a later version of the BFO).]

Informally, the BFO has difficulty describing physics in a realistic, machine-readable way, and changing the BFO to suit MSFC ACO's needs is akin to defining a completely new TLO, perpetuating the problem of ontologies designed for reuse not being re-used. The IAO fails to distinguish between data, entities that individually carry no intrinsic meaning, and information, entities that carry intrinsic meaning after being contextualized such that information is dependent upon data but data is not dependent upon information[45]. The distinction between data and information is necessary to support an accurate representation of knowledge in the software domain specific to M&S or otherwise.

If ontologies are to be used as the foundation for automated verification by analysis, both challenges of model reuse and ontology reuse must be addressed. A consistently reusable ontology must feature a broadly applicable content representation style, a robustness of purpose, and a level of abstraction highly aligned to engineering reality wherein both meticulously detailed and intentionally vague representations are possible. Engineers using the ontology to create models must have the ability to represent domain-specific model content and execute orders of operations using ontologically supported M&S techniques. To this end, the Alabama CubeSat Initiative (ACSI) is developing the Unified Ontological Foundry (UOF) as a basis for ontologically foundational work in spacecraft engineering, including the NASA ACO CSSO. A software platform for ontologically supported MBSE, MBPM, and M&S is being developed in lockstep with the UOF. This platform is called the Engineering Management Platform for Integration, Realization, and Execution (EMPIRE); EMPIRE version alpha will be released open source by 2024. Additional discussion of the UOF and EMPIRE is provided later in this work.

ONTOLOGICAL VERIFICATION

Requirements development is described with historical context as foundational material concerning M&S validation. Requirements verified by analysis are coupled with object-oriented technical performance measurement methods in preparation for ontological exemplification. Readers may find certain phrasing unnatural, such as "numeric parameters may assume the role of Technical Performance Measures (TPM)" instead of "numeric parameters can be TPMs," but this phrasing is intentional and supports machine-readability in an ontological context. Without delving deeply into BFO theory and justification, the attribution and loss of a BFO role such as TPM does not fundamentally change the structure of a numeric parameter. The attribution instead sets an expectation for how the role of TPM must be acquired and subsequently used[44].

Hatley-Pirbhai: Requirements and Functions

The H-P method is a MBSE method originating in the 1980s that separated any system specification into two models: requirements and architecture[13,46]. The first stage of the H-P method built the requirements model as a functional decomposition of the system's purpose; context diagrams translated stakeholder needs into customer and system specifications[13]. Parallel to the functional requirements model was a physical decomposition model with system architecture flow and connection diagrams. Viewpoints, a system organization tool used in SE methods such as NASA-STD-1005: the Space Mission Architecture Framework[47], were first conceptualized within the H-P method, though the authors did not refer to requirements and architecture models as viewpoints outright. The requirements and architecture models later became known as the logical or functional viewpoint and the physical viewpoint, respectively. There are needlessly restrictive ways to build viewpoints into applied SE methods[10], but relating system requirements to system function using diagrams was established as a useful MBSE practice by the H-P method. Maier provides a thorough overview of the H-P method and its impact on modern digital engineering[13], an especially useful perspective considering Maier also wrote parts of ISO 42010. Modern system modeling languages such as SysML[48] offer additional diagram types such as parametric and use case diagrams for system characterization outside requirements and architecture.

Verifying Requirements Compliance

Building upon the H-P method, a Requirements Verification and Traceability Matrix (RVTM)[1,3] is developed as part of a requirements specification. SysML or bespoke requirement development tools are useful when allocating verification methods to requirements and tracing requirement verification across multiple requirements levels[48]. Requirements can be verified using a verification case, meaning the application of analysis, demonstration, inspection, or test, by tracing an upper-level requirement to one or more lower-level requirements and verifying the lower-level requirements[49], or a combination of both techniques. The allocation of verification methods and lower-level requirements to upper-level requirements is tabulated in the RVTM and graphically depicted in an Operational Compliance Assessment (OCA). Upper-level requirements are often verified by the compliance of several lower-level requirements, and lower-level requirements, termed specifications, are often verified by a verification case.

Parameter Evaluation

When defining machine-readable syntax and semantics for automated M&S, ambiguity must be removed from terms often used colloquially in engineering settings. Rigorous definition specificity facilitates the translation of definitions to mathematical axioms, which enable a computer to infer what quantities, equations, and evaluation units are related to a model developed for verification by analysis, ontologically relating verification methods in the RVTM to the evaluation of parameters. Ontological class definitions concerning parameter evaluation are provided here as preface materials for automated technical performance measurement. These definitions are part of the UOF and were created specifically to assist engineers in automating spacecraft M&S. While the BFO[50] is not perfect, some classes in the BFO hierarchy are well-formed, included in the UOF TLO, and referenced here. Mathematical axiom development is ongoing and will be published as future work relating to the present work, but semantics are defined for maximal flexibility in all cases to support domain interoperability. Parameter evaluation begins by distinguishing data versus information, a distinction the IAO fails to make.

Data Entity: A data entity is a generically dependent continuant[50] that carries no intrinsic meaning until contextualized as information.

Information Entity: An information entity is a generically dependent continuant[50] that carries intrinsic meaning for a period of time by virtue of contextualized data.

Reasonable minds may disagree on the distinctions between a parameter, quantity, and value, but the semantics of an expression containing these terms must be strictly defined for automatable M&S. Values are first split into numeric and non-numeric values.

Value: A value is a data entity that is ascribed a literal from the lexical space of a data type.

Numeric Value: A numeric value is a value that is ascribed a literal from the lexical space of a numeric data type.

Non-Numeric Value: A non-numeric value is a value that is ascribed a literal from the lexical space of a non-numeric data type.

Formally defining value as a class may appear redundant due to the existence of RDF literals, a class already capable of representing a value in its lexical form[29], but as RDF literals can appear only in the object position of a semantic triple and are not denoted by an IRI[29], assertions cannot be made that associate an attribute or a quality directly to a RDF literal, limiting the inference capabilities. Value is an adapter class that extends the functionality of RDF to allow for a direct association of attributes enabling inference capabilities on the representation of data structures, namely arrays and matrices. The distinction made by the XML schema type system regarding the lexical space and value space of a data type is reflected in the definition of value and all descendent entities. Boolean values are considered a subclass of non-numeric values; numeric values do not exist in Boolean algebra.

There is a pervasive, ingrained misconception concerning the difference between measurement and evaluation that must be rectified for a computer to parse the terms with appropriate inferences. Measurement and evaluation are often used interchangeably in natural language, and units associated with numeric values are commonly referred to as units of measure. The core problem is that measurement results in the creation of a quantity through instrumentation whereas evaluation results in the creation of a quantity through any evaluation process. The concept of units, exemplified by watt, kilogram, or meter, are here termed evaluation units. The distinction between evaluation and measurement becomes increasingly important when relating verification methods to requirements, described later in this section.

Evaluation Unit: An evaluation unit is a quality[50] ascribed a numeric value that allows for comparisons of quantities of the same type.

Certain evaluation units beget additional description information, e.g., a degree or radian must be defined relative to a coordinate frame and an axis within that coordinate frame.

Quantities are ontologically related to a quantity value and an evaluation unit. The BFO does not include the concept of quantities, making the utility these definitions and relationships provide specific to the UOF.

Quantity: A quantity is a specifically dependent continuant[50] that consists of a quantity value and a category of evaluation unit.

Quantities represent the actual amount of something. Quantity values represent the data describing that amount. Quantity values are described ontologically using a similar definition to that offered by the International Vocabulary of Metrology[51] (IVM), though not all definitions offered here are consistent with IVM definitions. The IVM makes improper use of the word "magnitude", implying that magnitude could represent a negative value.

Quantity Value: A quantity value is a numeric value that is part of a quantity and is associated with an evaluation unit of some type compatible with that of the evaluation unit of the owning quantity.

Engineers frequently ascribe symbolic or alphanumeric identifiers to parameters, requirements, and other useful, tabulated entities. This could be a requirement identifier such as "ACSI.L4.C&DH.6" or a parameter name such as "Earth Radiative Surface Emission."

Identifier: An identifier is an information entity[50] that is associated with a value or a symbol and is used referentially to denote an entity.

A parameter can have a quantity value by virtue of a quantity value being a subclass of numeric value. If a quantity value is used in some data or evaluation process, it is concretized in a parameter for that process.

Parameter: A parameter is an information entity[50] that consists of an identifier and a value such that it can be input to or output from a process.

Numeric Parameter: A numeric parameter is a parameter with a value part associated with a numeric data type.

Non-Numeric Parameter: A non-numeric parameter is a parameter with a value part associated with a non-numeric data type.

The distinction between evaluation and measurement logically places evaluation process as a superclass of measurement process. When considering other useful means of evaluating a quantity, both calculation and simulation are considered wherein calculation results in the ascription of a literal to a quantity value via the execution of an equation and simulation results in the ascription of a literal to a quantity value via the execution of a model. When these evaluation processes are ontologically defined and restrictions are placed on the types of evaluation processes associated with verification methods, the computational rigor required to automate M&S is architected.

Evaluation Process: An evaluation process is a process[50] that has at least one parameter as output and results in the ascription of a literal to one or more values contained in each output parameter.

Measurement Process: A measurement process is an evaluation process that results in the ascription of a literal to one or more quantity values as part of a numeric parameter.

Measurement devices and units under test are ontologically associated with measurement processes in lieu of associating measurement processes with sensors and units under test in the measurement process definition because measurement processes can occur outside of test processes.

Calculation Process: A calculation process is an evaluation process that has at least one numeric parameter as input, at least one numeric parameter as output, at least one equation that may be ordered, and results in the ascription of a literal to one or more numeric values as part of one or more numeric parameters.

Simulation Process: A simulation process is an evaluation process that has at least one numeric parameter as input, at least one numeric parameter as output, at least one mathematical model that may be ordered, and results in the ascription of a literal to one or more values as part of one or more parameters.

With these logically consistent definitions established, mathematical axioms can be created further supporting machine-parseability. Axioms supporting UOF definitions are the subject of ongoing work and future publications.

USING ONTOLOGICAL PARAMETERS

INCOSE defines three fundamental requirement types as functional, performance, and non-functional[14], but these can be ontologically simplified to functional and non-functional when considering the syntax of a machine-readable requirement statement. Functions are unitless;

they do not have an associated value or level of performance. A system either does or does not execute a function. Like a vector with a direction and magnitude, a capability is a function executed to a given level of performance[1]. The only differences between a performance requirement statement and a functional requirement statement are therefore a relational operator such as, “less than or equal to”, a numeric value such as “5”, and an evaluation unit such as “kg”. If a performance requirement statement is a functional requirement statement with the addition of a relational operator and a quantity value, a performance requirement is a subclass of functional requirement. Because functional requirements describe functions and performance requirements describe capabilities, it is less ambiguous to instead refer to performance requirements as capability requirements, a conclusion adopted in both the abstract and remainder of this work.

Ontological restrictions are levied concerning which types of verification methods can be applied to which types of requirements; the restrictions are predicated on the type of evaluation process associated with a given verification method. Verification by analysis can be achieved via calculation if an equation is executed to evaluate a parameter or simulation if a model is executed to evaluate a parameter. There are some instances when a simulation results in a Boolean value, meaning true or false. A Boolean value in this framework would be represented by the ascription of a Boolean literal to an instance of non-numeric value that is bounded by a Boolean data type. Inspection and demonstration by nature do not have an associated numeric value; their results are always associated with a Boolean value, a type of non-numeric value. Test is a verification method associated with a measurement process, an evaluation process resulting from the application of instrumentation to a unit under test. Because instrumented sensors output a numeric value associated with some evaluation unit, testing cannot result in a Boolean value. To summarize, the verification methods analysis, demonstration, and inspection can result in Boolean values. Analysis and test can result in numeric values. Functional requirements corresponding to unitless functions can be verified by analysis or demonstration because functional requirements require Boolean values as verification criteria. Capability requirements corresponding to functions executed at a given level of performance can be verified by analysis or test because capability requirements require numeric values as verification criteria. Non-functional requirements have Boolean values as verification criteria and can be verified by inspection only. The allocation of acceptable primary verification methods to fundamental requirement types is organized in Table 1.

Table 1: Primary Verification Methods for Fundamental Requirement Types

Requirement Type	Possible Verification Method
Functional	Analysis, Demonstration
Capability (Performance)	Analysis, Test
Non-Functional	Inspection

Technical Performance Measurement

Validation ensures a system, product, service, or model fulfills its users’ operational needs[1], and relating models to system requirements ensures time spent developing, automating, and executing models is validated to the purpose of model users – to verify the system is developed in compliance to imposed requirements. To connect models to capability requirements and thereby validate M&S effort, engineers describe levels of performance using TPMs. A numeric parameter assumes the role of TPM if the numeric parameter is used for comparison of modeled or actual performance against that anticipated at the current time and on future dates[52]. As Boolean values are non-numeric values, this implies Boolean results from analysis, inspection, or demonstration cannot assume the role of TPM. TPMs exist for components, subsystems, and systems and are used to confirm progress or identify deficiencies that might jeopardize meeting a system requirement. If a requirement dictates a component can only use 5 W of power and the system is verified by analysis to use only 4 W of power, the TPM would be a component power margin with a value of 1 W. TPMs should[53],

- be relevant to the entity and tailored to the mission
- be relatively easy to measure or evaluate
- be controllable, i.e. tradeable with cost, schedule, and performance
- have a value or uncertainty that is expected to improve with time
- have a target or threshold value
- have known corrective action if the target or threshold value is exceeded

NASA decomposes stakeholder requirements into Measures of Effectiveness (MOE), translates MOEs into Measures of Performance (MOP), and selects TPMs from MOPs. The most important, system-wide TPMs relating to mission success are Key Performance Parameters (KPP), characterizing major drivers of operational performance, supportability, and interoperability26. The ACSI has found students experience significant difficulty understanding and successfully implementing MOEs and MOPs, and the NASA parameter role delineation is not ontologically supportable because it contains subjective boundaries. A simplified set of concentric numeric parameter roles is taught instead using NASA's TPM and KPP definitions with the addition of one numeric parameter role: Knowledge Points (KP). Shown in Figure 2, these numeric parameter roles are both relevant to practical verification by analysis and ontologically useful. UOF definitions for numeric parameter roles are provided here.

> Knowledge Point: A knowledge point is a role[50] a numeric parameter may assume if it is evaluated through some evaluation process.
>
> Technical Performance Measure: A technical performance measure is a role[50] a numeric parameter may assume if it is controllable, has a quantity value that is expected to improve with time, has a target or threshold quantity value, and corrective action is known if the target or threshold quantity value is exceeded or not met.
>
> Key Performance Parameter: A key performance parameter is a role[50] a numeric parameter may assume if it meets the criteria to assume the role of TPM and relates directly to system-level stakeholder needs.

Building on existing NASA definitions[26], all KPPs are TPMs and all TPMs are KPs, ergo all KPPs are also KPs. The evaluation of a KP through an evaluation process is a central feature; KPs are not sourced from a textbook or provided in a vendor's component datasheet. KPs are always calculated, simulated, or measured, else parameters cannot assume the role of KP, but not all evaluated parameters assume the role of KP. KPs may also not meet the criteria to be considered TPMs. The maximum heat flux incident on the +Z face of a spacecraft is a numeric parameter that is relevant to the system and relatively easy to evaluate, but it is not controllable. There is not a target value for maximum heat flux, and there is not an expectation that the value or uncertainty inherent to parameter evaluation will improve over time. Maximum heat flux incident on the +Z face is therefore a numeric parameter assuming the role of KP, not the role of TPM. NASA prefers to attribute the role of TPM to margins because they feel it is easier to set current and future targets on margins instead of calculated values[54].

If project managers and systems engineers understand the rules and purpose of technical performance measurement, they may tailor which numeric parameters assume the role of KP or TPM at their discretion based on imposed requirements. What is important is understanding what parameter role a capability requirement is specifying as a level of performance and how a given organization plans to track the compliance of their design to that requirement. Once parameter roles are understood in relation to imposed requirements verified by analysis, one or more parameters assuming the role of KP, TPM, or KPP are allocated to models. Models are allocated to modeling environments, developed using traditional or ontological methods, and executed for some baselined system design. Systems engineers can then enjoy a holistic view of which parameters must be evaluated using which models in which modeling environments.

Visualizing Parameter Relationships

Object-oriented representations of individual KP evaluation methodologies in user-friendly natural language are referred to as Domain Knowledge Maps (DKM). A DKM example is provided in Figure 3 for the Sun to Satellite View Factor KP. Each oval in Figure 3 corresponds to a numeric parameter that has assumed the role of KP. Non-oval entities may be numeric parameters or any other entity type described in the legend.

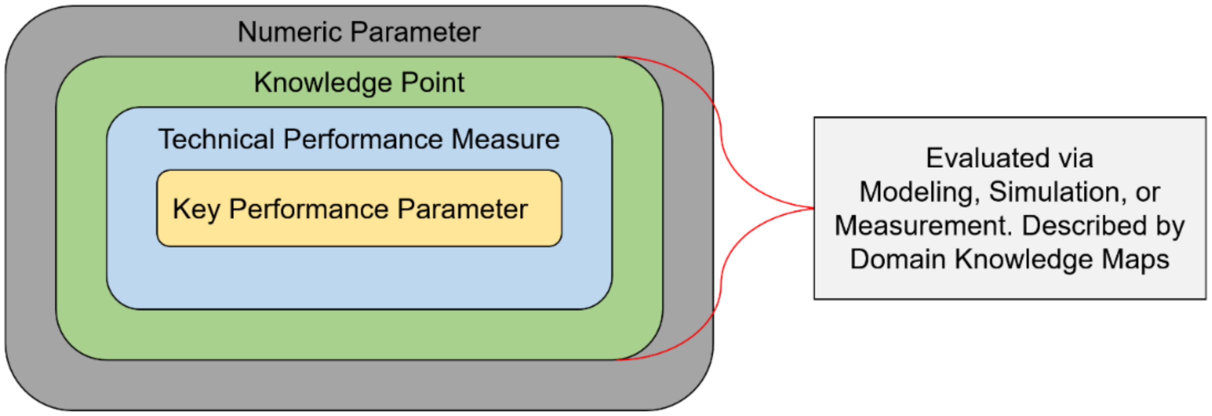

Numeric Parameter
Knowledge Point
Technical Performance Measure
Key Performance Parameter
Evaluated via Modeling, Simulation, or Measurement. Described by Domain Knowledge Maps

Figure 2: Simplified Numeric Parameter Roles Replacing Measures of Effectiveness and Measures of Performance

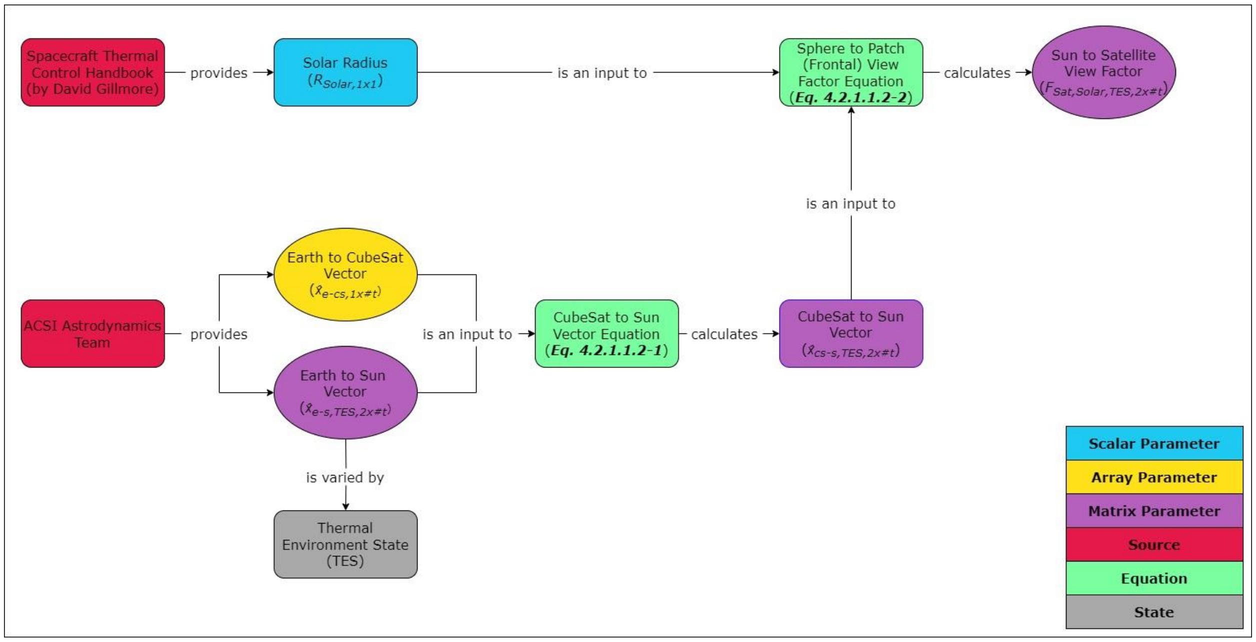

Spacecraft Thermal Control Handbook (by David Gillmore)
provides
Solar Radius ($R_{Solar,1x1}$)
is an input to
Sphere to Patch (Frontal) View Factor Equation (Eq. 4.2.1.1.2-2)
calculates
Sun to Satellite View Factor ($F_{Sat,Solar,TES,2x\neq t}$)
is an input to
Earth to CubeSat Vector ($\hat{x}_{e\text{-}cs,1x\neq t}$)
ACSI Astrodynamics Team
provides
is an input to
CubeSat to Sun Vector Equation (Eq. 4.2.1.1.2-1)
calculates
CubeSat to Sun Vector ($\hat{x}_{cs\text{-}s,TES,2x\neq t}$)
Earth to Sun Vector ($\hat{x}_{e\text{-}s,TES,2x\neq t}$)
is varied by
Thermal Environment State (TES)
Scalar Parameter
Array Parameter
Matrix Parameter
Source
Equation
State

Figure 3: Domain Knowledge Map for Sun to Satellite View Factor

Numeric parameters assuming the role of KP are often evaluated using M&S on a longer road to TPM evaluation; this necessitates KPs to be strung together in KP integration diagrams. An example is shown in Figure 4 for a subsection of the incident spacecraft heat flux calculation methodology described in Halvorson et al[8]. Each oval in Figure 4 corresponds to the same KP as the eponymous ovals of Figure 3, and KP integration diagrams form both an easily digestible process for new spacecraft engineers to use as onboarding material and an order of operations for high-level, machine-readable instruction sets. To reiterate, the Sun to Satellite View Factor KP for which the DKM in Figure 3 defines the calculation process methodology is one of the KPs present in Figure 4. The KP integration diagram in Figure 4 does not represent an exhaustive set of KPs for complete spacecraft thermal analysis, but one can readily see how this numeric parameter organization can be both extended to the desired level of detail and applied to other domains. Starting at the left, the red source block defines the team or group responsible for the evaluation of the KP. Prescribing ownership to a KP, meaning a prescription of requirement compliance responsibility related to that KP, enables project and team managers to partition verification responsibility. Moving right in Figure 4 from the source into ovals denoting KPs, each KP shown has its own DKM such as that of Figure 3. KPs, each with bespoke DKMs, are linked together until the TPM corresponding to a capability requirement's level of performance is reached. In Figure 3 are a few examples of the differences between KPs and numeric parameters not assuming the role of KP. Solar Radius is a numeric parameter that does not assume the role of a KP and therefore is also not a TPM or KPP. Non-KP parameters are shown as rectangular blocks. CubeSat to Sun Vector is a parameter that is evaluated via calculation in an equation, but it is subjectively not a KP because it is considered a steppingstone to the desired KP Sun to Satellite View Factor. The decision to keep CubeSat to Sun Vector as a parameter that does not assume the role of a KP is subjective in a different sense than MOEs and MOPs are subjective. Engineers using EMPIRE to create DKMs and KP integration diagrams will automatically link KP diagram content to other EMPIRE content such as system architecture and requirements. Any evaluated numeric parameter can be labeled as a KP to suit analysis interests. One could argue that every evaluated parameter should be a KP; that strategy is legal in EMPIRE but may not be desired by all organizations. TPMs may evolve as the architecture matures through design decisions, so parameters may assume the role of KPs or TPMs during concept definition but not during verification or integration. Because KP integration diagrams are built on DKMs and DKMs are both object-oriented and written in a natural language syntax, users can readily create machine-readable diagrams linked to a backend database using their desired parameter definition strategy. A goal in future work is for DKM and related EMPIRE database content to populate subsystem analysis documentation, avoiding the need for users to write model export code into documents using Velocity Template Language or another model export language. The equation numbers in the green equation blocks are specific to the equation numbers in ACSI documentation and do not have significance in the context of the present work.

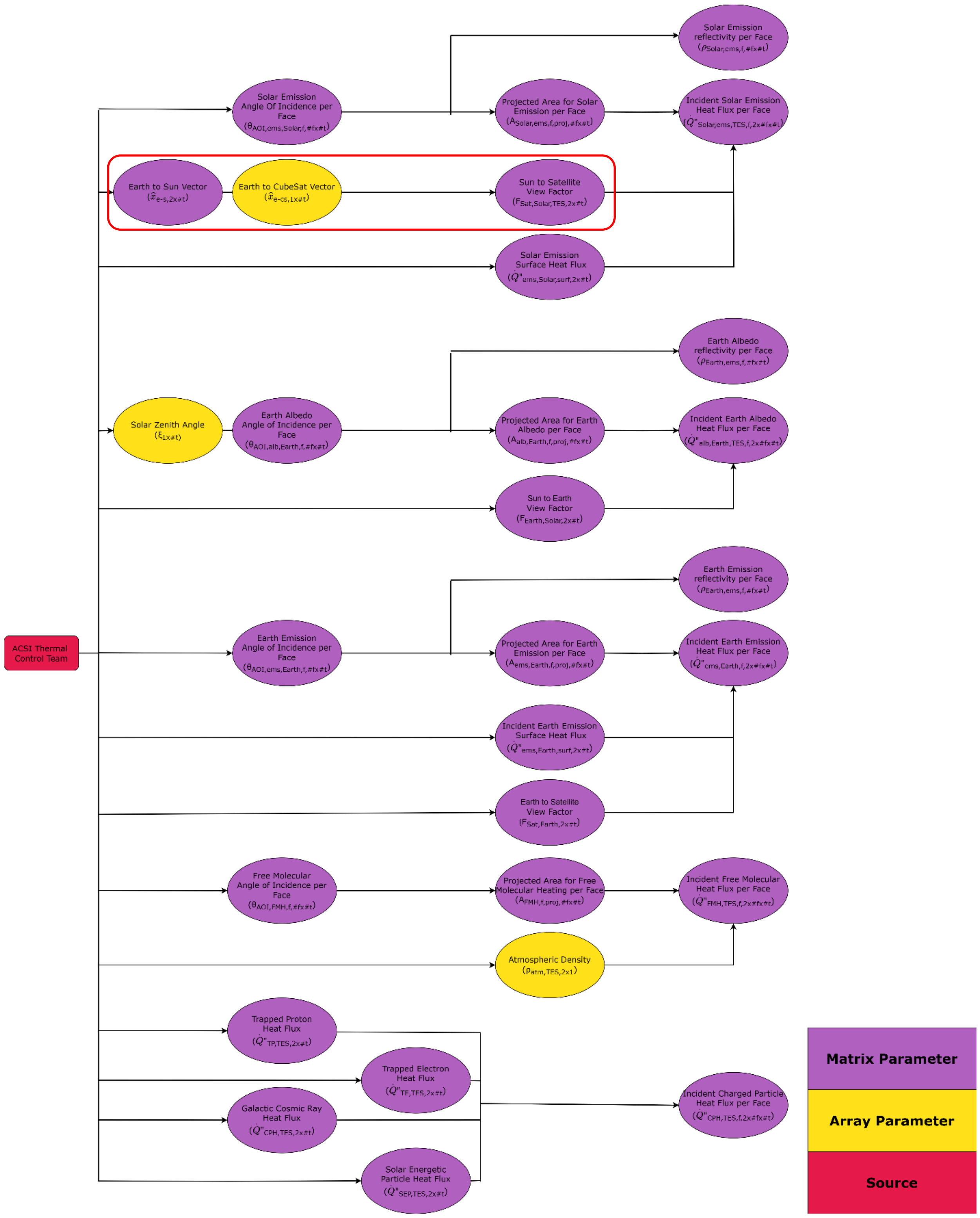

ACSI Thermal Control Team
Solar Emission reflectivity per Face ($\rho_{Solar,ems,f,\#fx\#t}$)
Solar Emission Angle Of Incidence per Face ($\theta_{AOI,ems,Solar,f,\#fx\#t}$)
Projected Area for Solar Emission per Face ($A_{Solar,ems,f,proj,\#fx\#t}$)
Incident Solar Emission Heat Flux per Face ($\dot{Q}''_{Solar,ems,TES,f,2x\#fx\#t}$)
Earth to Sun Vector ($\vec{x}_{e-s,2x\#t}$)
Earth to CubeSat Vector ($\vec{x}_{e-cs,1x\#t}$)
Sun to Satellite View Factor ($F_{Sat,Solar,TES,2x\#t}$)
Solar Emission Surface Heat Flux ($\dot{Q}''_{ems,Solar,surf,2x\#t}$)
Earth Albedo reflectivity per Face ($\rho_{Earth,ems,f,\#fx\#t}$)
Solar Zenith Angle ($\xi_{1x\#t}$)
Earth Albedo Angle of Incidence per Face ($\theta_{AOI,alb,Earth,f,\#fx\#t}$)
Projected Area for Earth Albedo per Face ($A_{alb,Earth,f,proj,\#fx\#t}$)
Incident Earth Albedo Heat Flux per Face ($\dot{Q}''_{alb,Earth,TES,f,2x\#fx\#t}$)
Sun to Earth View Factor ($F_{Earth,Solar,2x\#t}$)
Earth Emission reflectivity per Face ($\rho_{Earth,ems,f,\#fx\#t}$)
Earth Emission Angle of Incidence per Face ($\theta_{AOI,ems,Earth,f,\#fx\#t}$)
Projected Area for Earth Emission per Face ($A_{ems,Earth,f,proj,\#fx\#t}$)
Incident Earth Emission Heat Flux per Face ($\dot{Q}''_{ems,Earth,f,2x\#fx\#t}$)
Incident Earth Emission Surface Heat Flux ($\dot{Q}''_{ems,Earth,surf,2x\#t}$)
Earth to Satellite View Factor ($F_{Sat,Earth,2x\#t}$)
Free Molecular Angle of Incidence per Face ($\theta_{AOI,FMH,f,\#fx\#t}$)
Projected Area for Free Molecular Heating per Face ($A_{FMH,f,proj,\#fx\#t}$)
Incident Free Molecular Heat Flux per Face ($\dot{Q}''_{FMH,TES,f,2x\#fx\#t}$)
Atmospheric Density ($\rho_{atm,TES,2x1}$)
Trapped Proton Heat Flux ($\dot{Q}''_{TP,TES,2x\#t}$)
Trapped Electron Heat Flux ($\dot{Q}''_{TE,TES,2x\#t}$)
Galactic Cosmic Ray Heat Flux ($\dot{Q}''_{CPH,TES,2x\#t}$)
Solar Energetic Particle Heat Flux ($\dot{Q}''_{SEP,TES,2x\#t}$)
Incident Charged Particle Heat Flux per Face ($\dot{Q}''_{CPH,TES,f,2x\#fx\#t}$)
Matrix Parameter
Array Parameter
Source

Figure 4: Knowledge Point Integration Diagram for a Section of the Advanced Thermal Analysis Methodology Presented in Halvorson et al[8]. Figure 3 Content is Marked with a Red Outline.

Model Generation Versus Model Linkage

In these early days of M&S automation, engineers must still build models in modeling environments such as MATLAB, Simulink, or Thermal Desktop to match DKM organization, but the long-term goal is for object-oriented structures in DKMs to generate model content specific to the system. If the goal was to build models separately from a defined model execution hierarchy, users would be better enabled by linking externally built models to SysML parametric diagrams, though this process is time-consuming and ill-advised for understaffed projects. Generation of models corresponding to DKM content is what differentiates EMPIRE diagrams from parametric diagrams in SysML. MATLAB models can be linked to SysML diagrams, but SysML diagrams cannot generate MATLAB code.

ONTOLOGICAL AUTOMATION

The creation of a DKM in EMPIRE provides ontology-supported database population functionality. When users create DKMs, the sources, equations, parameters, and states defined also populate backend database fields. Because the natural language DKMs have an underlying, ontological structure, fidelity of database relationships is ensured during DKM creation. Natural language, however, does not enable machines to understand the relationships between DKM entities such as parameters, sources, and equations. A bridge between the user-friendly natural language and the machine-friendly formal language must be defined.

The BFO-2020 is recognized as a TLO by ISO 21838-2:2021 and is used to facilitate the representation of entities pertinent to DKMs. Definitions for BFO terminology can be found in the BFO 2.0 User's Guide[44]. The BFO 2.0 and BFO-2020 are separate iterations of the BFO, but few changes between versions are directly pertinent to this work. The BFO-2020 added an "enriched treatment of relations involving time" such that previously existing relationships such as "concretizes" were expanded to variations such as "concretizes at some time" and "concretizes at all times"[39]. The phrase "at some time" denotes that the entity in the object position of a semantic triple is impacted by the predicate to which it is attached at some temporal instant and potentially for the duration of an associated temporal interval, and "at all times" denotes said object would be impacted by the predicate to which it is attached through the entirety of the temporal interval that bounds its existence.

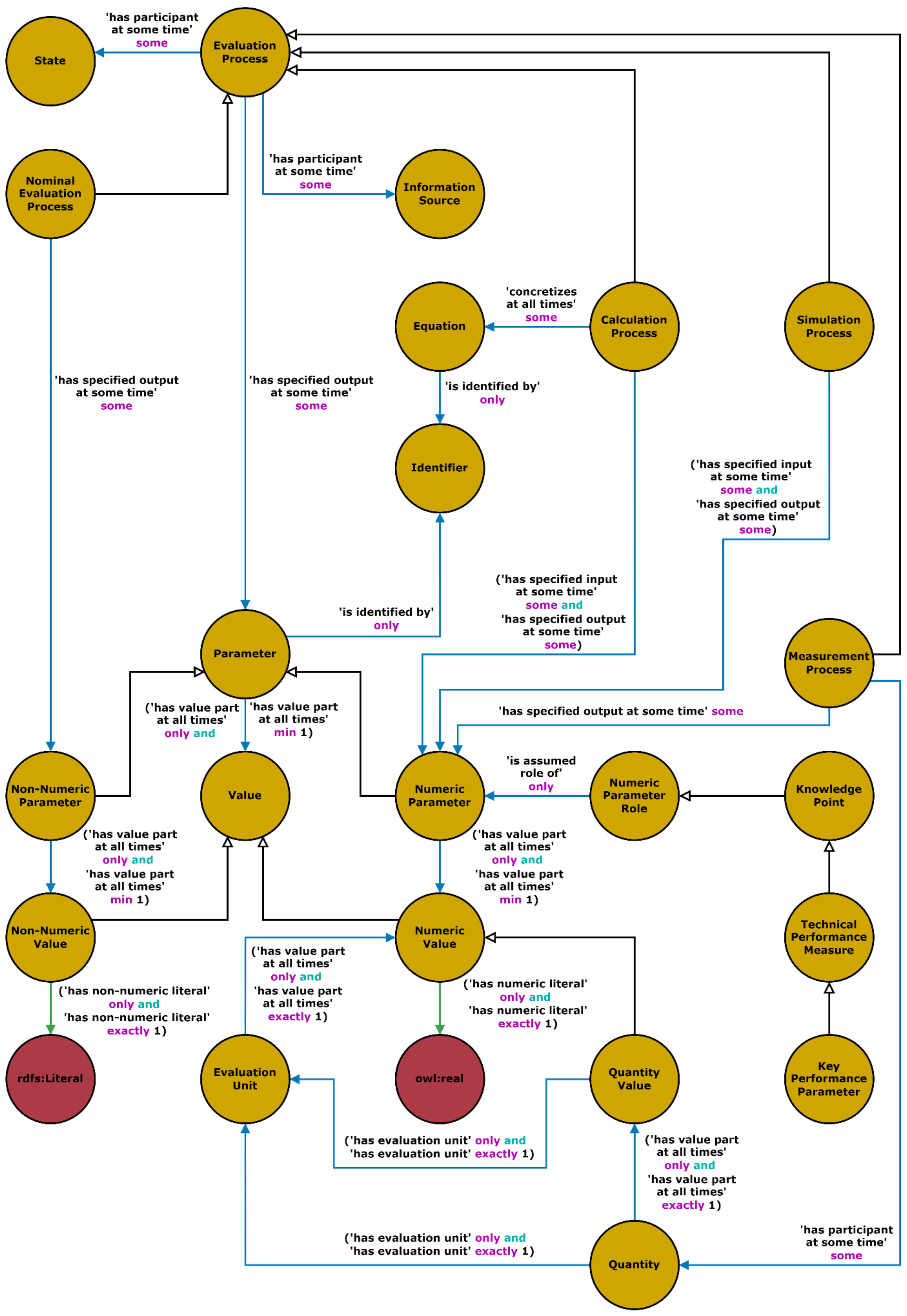
The vocabulary of a DKM such as that of Figure 3 is investigated to facilitate additional context necessary for automated M&S. The following definitions of the extrapolated vocabulary of a DKM are prescribed in ontological format:

> <u>Information Source</u>: An information source is a role[50] that an independent continuant[50] or generically dependent continuant[50] assumes if providing some information.
>
> <u>Equation</u>: An equation is a generically dependent continuant[50] that represents a mathematical statement of equality.
>
> <u>State</u>: A state is a disposition[50] that specifically depends on some system or operational environment and exists during some temporal interval.

States here are described in the context of Wasson[1] and can be system, configuration, operational, dynamic, or environmental states. The Thermal Environment State (TES) in Figure 3 varies parameters to represent their hottest and coldest possibilities. If a parameter is varied by TES, the chosen data structure must store 2 elements at a given index because there are two TES. This is an example of semantic inference applied to the data size field of a parameter. A visualization of the ontological structure of a DKM is shown in Figure 5 in Manchester OWL Syntax. The Manchester OWL Syntax is used to represent class expressions in Figure 5 in a format identical to that of Protégé 5. The Manchester OWL Syntax is an alternative abstract syntax for OWL DL that allows for development of ontologies with a "less logician like" syntax in that "special mathematical symbols such as ∃, ∀, ¬ and have been replaced by more intuitive keywords such as some, only, and not," wherein the former two symbols correspond to existential and universal property restrictions respectively[55]. Shown in Figure 5 are ontological relationships necessary to enable semantic inference, based on an asserted evaluation process, of the structure of unknown or not yet instantiated input or output parameters.

Input-Process-Output

With historical perspectives and best practices concerning requirements verification, technical performance measurement, and translation of M&S entities into machine-readable data structures understood, the next step is defining the conceptual organization of automated M&S methodology execution. DKMs and KP integration diagrams convert technical performance measurement information into machine-readable structures suitable for back-end database field population, but a common framework for providing database content to executable models and retrieving KP and TPM outputs from models must be defined. The Input-Process-Output (IPO) framework is considered as a solution to M&S automation theory.


State
'has participant at some time' some
Evaluation Process
Nominal Evaluation Process
'has participant at some time' some
Information Source
'concretizes at all times' some
Equation
Calculation Process
Simulation Process
'has specified output at some time' some
'has specified output at some time' some
'is identified by' only
Identifier
('has specified input at some time' some and 'has specified output at some time' some)
('has specified input at some time' some and 'has specified output at some time' some)
'is identified by' only
Parameter
Measurement Process
('has value part at all times' only and
'has value part at all times' min 1)
'has specified output at some time' some
'is assumed role of' only
Non-Numeric Parameter
Value
Numeric Parameter
Numeric Parameter Role
Knowledge Point
('has value part at all times' only and 'has value part at all times' min 1)
('has value part at all times' only and 'has value part at all times' min 1)
Non-Numeric Value
Numeric Value
Technical Performance Measure
('has value part at all times' only and 'has value part at all times' exactly 1)
('has non-numeric literal' only and 'has non-numeric literal' exactly 1)
('has numeric literal' only and 'has numeric literal' exactly 1)
rdfs:Literal
Evaluation Unit
owl:real
Quantity Value
Key Performance Parameter
('has evaluation unit' only and 'has evaluation unit' exactly 1)
('has value part at all times' only and 'has value part at all times' exactly 1)
('has evaluation unit' only and 'has evaluation unit' exactly 1)
Quantity
'has participant at some time' some

Legend					
Entity	Class	Data Type	Object Property	Data Property	SubClassOf
Key	●	●	⟶	⟶	⟶▷

Figure 5: Ontological Representation of DKM Structure

IPO has been used in economics, law, and software since the 70s; its earliest origins are unclear. Forsberg, Walden, Shortell, Roedler, and Hamelin organized SE processes using the IPO framework for the INCOSE SE Handbook[14], which makes IPO an attractive M&S automation option for future SE handbook inclusion. Forsberg and Mooz invented the Vee life cycle model in 1991[56], Walden developed criteria to tailor SE processes to bespoke projects[57], and Roedler published on both knowledge management and technical performance measurement[58,59], indicating INCOSE SE Handbook authors consider the IPO model useful for a variety of system architecture descriptions. Each process includes controls, activities, and enablers in addition to inputs and outputs. The INCOSE SE Handbook organizes certain chapters into technical, technical management, agreement, organizational project-enabling, and tailoring processes; the system analysis process is considered a technical process. Table 2 defines a high-level overview of the inputs, activities, and outputs for the system analysis process per the INCOSE SE Handbook.

Table 2: System Analysis Process IPO Considerations[14]

Inputs	Activities	Outputs
Life cycle concepts	Prepare for system analysis	System analysis strategy
Analysis situations	Perform system analysis	System analysis report
Life cycle constraints	Manage system analysis	System analysis record

Context descriptions are provided for each process activity, and the present assertion is that ontological, automated M&S should also adhere to theory-supported process activities. When preparing for system analysis, the scope, types, objectives, level of accuracy, evaluation criteria, methods, relevant requirements, modeling environments, order of operations, and relevant documentation must be defined. These criteria are almost all met when NASA-STD-7009A is applied, specifically Appendix E regarding M&S credibility assessments[15]. Appendix E asserts eight factors for M&S credibility in three categories: development, operations, and supporting evidence. Development defines credibility factors for data pedigree, verification, and validation. Operations define factors for input pedigree, uncertainty characterization, and results robustness. Supporting evidence defines factors for M&S history and M&S process/product management. These factors can be taxonomized in an ontology and related to technical performance measurement as database fields, though EMPIRE has not included this yet. When performing system analysis, the INCOSE SE handbook considers collection of data and assumptions, execution of the model, and application of peer-reviews to the methodology. Figure 6 represents a visualization of model process for the Isothermal First Order (IFO) model described in Halvorson et al.[8] created in the MATLAB modeling environment. While Figure 6 complements Figure 3 and Figure 4 in describing the order of operation for model execution instead of the dependence of parameters upon previously evaluated parameters, its ability to describe model assumptions falls short. Models themselves should therefore be ascribed assumptions, behaviors, and use cases in editable database fields for additional peer review. System analysis management as the final IPO activity concerns documentation, an easily automatable task for database-represented information.

A key problem in both providing database field information to and receiving it from modeling environments is that disparate modeling environments feature disparate Graphical User Interfaces (GUI) or data import methods. Modeling environments such as MATLAB can be interfaced with in an object-oriented context via the MATLAB Engine for Python API[60], making data import from and export to EMPIRE relatively easy, but modeling environments relevant to domain-specific M&S, e.g., Simulink and Thermal Desktop for spacecraft thermal analysis, require design-specific information provided to the modeling environment through graphical means such as Simscape blocks. The automation of deterministic, equation-based analysis methodologies is therefore considered for inchoate ontological automation, and ontologically driven automation using Simulink and Thermal Desktop in this thermal analysis example is considered the subject of future work.

Ontological Analysis Automation Summary

The steps to relate requirements to executable models in an ontologically supported platform such as EMPIRE are then,

1. Develop the RVTM and OCA to determine system requirements verified by analysis.
2. Define Boolean values associated with functional requirements verified by analysis.
3. Define numeric values associated with capability requirements verified by analysis.
4. For numeric parameters associated with capability requirements verified by analysis, define which numeric parameters assume the role of TPMs or KPs.
5. Allocate models to one or more KPs resulting in a TPM such that all KPs are related to a model.
6. Allocate modeling environments to models.

7. Define which KPs are inputs and outputs of which models. In EMPIRE these populate database fields.
8. Define the process for model execution using KP integration diagrams and DKMs.
9. Interface the model-based platform or SysML model with the modeling environment.
10. Export input parameters from the model-based platform or SysML model to the technical modeling environments.
11. Import evaluated KPs and Boolean values from the technical modeling environments to the model-based platform or SysML model.
12. Assess requirements compliance.

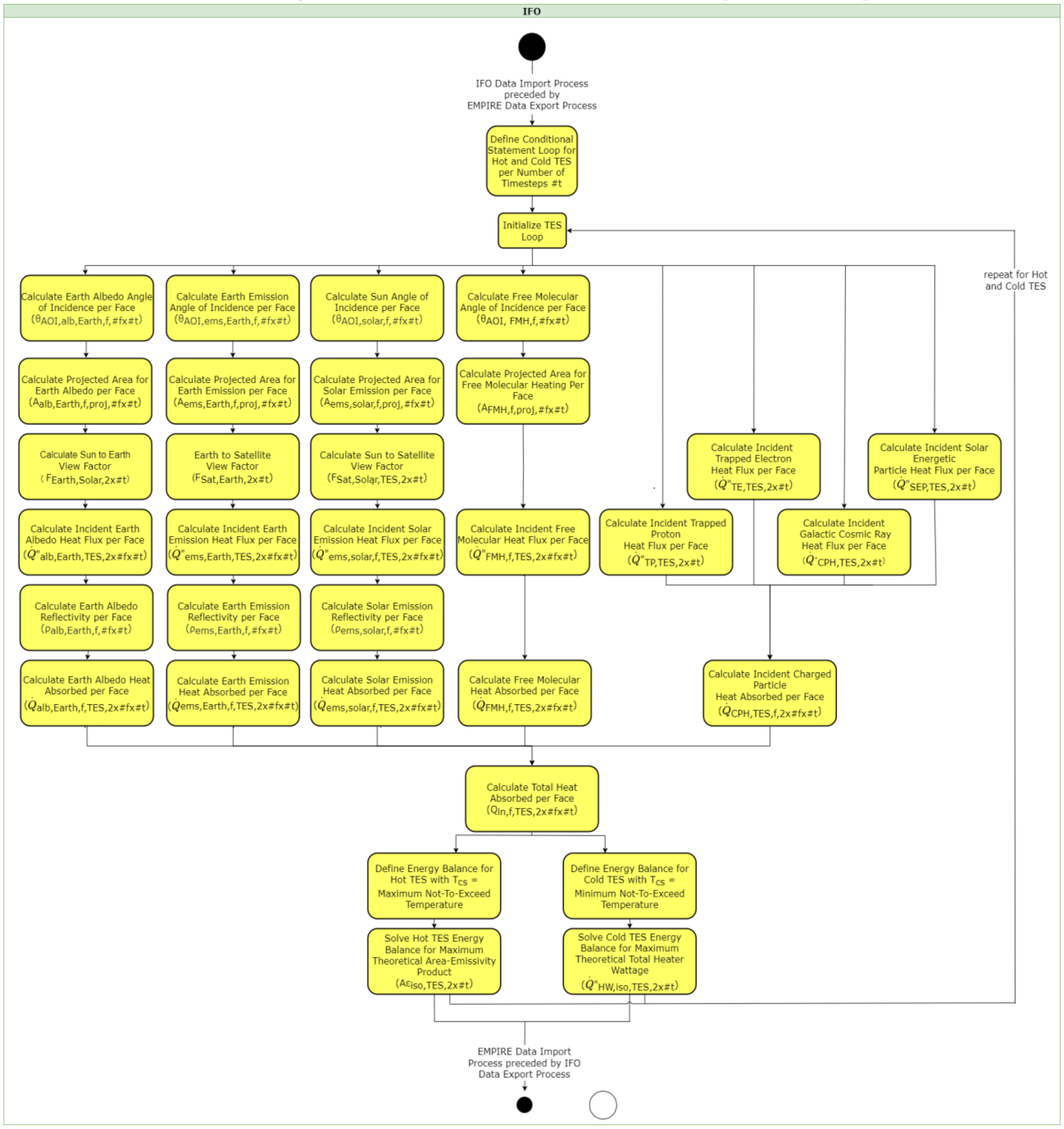

IFO
IFO Data Import Process preceded by EMPIRE Data Export Process
Define Conditional Statement Loop for Hot and Cold TES per Number of Timesteps #t
Initialize TES Loop
repeat for Hot and Cold TES
Calculate Earth Albedo Angle of Incidence per Face (θAOI,alb,Earth,f,#fx#t)
Calculate Earth Emission Angle of Incidence per Face (θAOI,ems,Earth,f,#fx#t)
Calculate Sun Angle of Incidence per Face (θAOI,solar,f,#fx#t)
Calculate Free Molecular Angle of Incidence per Face (θAOI, FMH,f,#fx#t)
Calculate Projected Area for Earth Albedo per Face (Aalb,Earth,f,proj,#fx#t)
Calculate Projected Area for Earth Emission per Face (Aems,Earth,f,proj,#fx#t)
Calculate Projected Area for Solar Emission per Face (Aems,solar,f,proj,#fx#t)
Calculate Projected Area for Free Molecular Heating Per Face (AFMH,f,proj,#fx#t)
Calculate Sun to Earth View Factor (FEarth,Solar,2x#t)
Earth to Satellite View Factor (FSat,Earth,2x#t)
Calculate Sun to Satellite View Factor (FSat,Solar,TES,2x#t)
Calculate Incident Trapped Electron Heat Flux per Face (Q̇"TE,TES,2x#t)
Calculate Incident Solar Energetic Particle Heat Flux per Face (Q̇"SEP,TES,2x#t)
Calculate Incident Earth Albedo Heat Flux per Face (Q̇"alb,Earth,TES,2x#fx#t)
Calculate Incident Earth Emission Heat Flux per Face (Q̇"ems,Earth,TES,2x#fx#t)
Calculate Incident Solar Emission Heat Flux per Face (Q̇"ems,solar,f,TES,2x#fx#t)
Calculate Incident Free Molecular Heat Flux per Face (Q̇"FMH,f,TES,2x#fx#t)
Calculate Incident Trapped Proton Heat Flux per Face (Q̇"TP,TES,2x#t)
Calculate Incident Galactic Cosmic Ray Heat Flux per Face (Q̇"CPH,TES,2x#t)
Calculate Earth Albedo Reflectivity per Face (ρalb,Earth,f,#fx#t)
Calculate Earth Emission Reflectivity per Face (ρems,Earth,f,#fx#t)
Calculate Solar Emission Reflectivity per Face (ρems,solar,f,#fx#t)
Calculate Earth Albedo Heat Absorbed per Face (Q̇alb,Earth,f,TES,2x#fx#t)
Calculate Earth Emission Heat Absorbed per Face (Q̇ems,Earth,f,TES,2x#fx#t)
Calculate Solar Emission Heat Absorbed per Face (Q̇ems,solar,f,TES,2x#fx#t)
Calculate Free Molecular Heat Absorbed per Face (Q̇FMH,f,TES,2x#fx#t)
Calculate Incident Charged Particle Heat Absorbed per Face (Q̇CPH,TES,f,2x#fx#t)
Calculate Total Heat Absorbed per Face (Q̇in,f,TES,2x#fx#t)
Define Energy Balance for Hot TES with TCS = Maximum Not-To-Exceed Temperature
Define Energy Balance for Cold TES with TCS = Minimum Not-To-Exceed Temperature
Solve Hot TES Energy Balance for Maximum Theoretical Area-Emissivity Product (Aεiso,TES,2x#t)
Solve Cold TES Energy Balance for Maximum Theoretical Total Heater Wattage (Q̇"HW,iso,TES,2x#t)
EMPIRE Data Import Process preceded by IFO Data Export Process

Figure 6: Isothermal First Order Thermal Model IPO Process Diagram[8]

EMPIRE: PURPOSE, FUNCTION, AND STATUS

EMPIRE was conceived when aerospace, mechanical, electrical, and software engineering students from multiple universities had a need to architect, design, integrate, and verify the 12U Space Transporter by ACSI (Space TACSI) satellite bus. ACSI students learn technical discipline engineering from their constituent universities, but students from all universities stated their exposure to SE principles were limited at best, a difficult foundation to build a spacecraft engineering education upon considering every engineer, no matter the technical discipline, is a systems engineer[1]. SysML offers a means to develop models representing a central source of truth for all ACSI projects, but SysML diagrams do not inherently facilitate professional SE. The soul of SE is not concerned with diagrams or documentation; SE is about making decisions that reduce the time, effort, and money required to realize a system validated to stakeholder needs[1]. Diagrams in SysML are used to represent core SE features such as architecture, interfaces, use cases, and requirements, but practitioners must still develop SE content based on their experience. When coupling a need for substantial SE experience with the steep learning curve common to SysML-based modeling environments, it is clear the existence of SysML and other SE modeling tools does not inherently provide large-scale SE capabilities.

Simultaneously, all PM efforts were organized via graduate students at the University of Alabama in Huntsville, meaning document version control and maintenance of work breakdown structures, organizational breakdown structures, integrated master plans, integrated master schedules and any team-specific documentation was a constant effort based on the experience and leadership of a few capable engineers. Online platforms suitable for geographically disperse PM such as Trello, Jira, or Wrike are useful for non-technical PM or specific technical discipline features such as software bug management, but they are not organized using core SE principles such as architecture, integration, and verification. These problems are not specific to university-class programs. When knowledge and authority is concentrated at the top of any organization, personnel turnover, an inherent feature of university programs due to frequent, expected graduations, means programs are gambling that their next leaders will be as competent and capable as the previous ones. Information silos develop and cripple an organization's ability to make and execute decisions. Many engineers enter the workforce without understanding SE principles, so a combined PM and SE platform that organizes engineering work around SE principles would relieve the need to train engineers in SE before they could work effectively. Thus, EMPIRE was born.

EMPIRE Functionality

When engineers initiate a spacecraft project, the first task is to evaluate stakeholder needs. Is the project a payload or a satellite with one or more payloads? Where is the system going? What kind of risk posture does the project assume? Model-Based Mission Planning (MBMP) was described using ontological querying in Halvorson et al[61]., and this functionality is integrated into EMPIRE. Table 3 describes the questions asked of project managers when instantiating a project and the resulting data generated based on the answers to those questions. The rationale for question genesis and order will be the topic of publication in future work for the 2024 International Astronautical Congress (IAC). This data querying relieves the need for managers and engineers to manually populate EMPIRE with desired data. Answers are selected from a list of options and are not provided as short answers. Once question responses have been ontologically translated into database fields, those fields are used to populate information in EMPIRE sections for architecture, design, integration, requirements, verification, and an executive dashboard. That database field content in part includes the parameters used as a basis for DKM creation and model development.

EMPIRE Status

EMPIRE is a work in progress. While the database backend, some of the GUI, and some of the DKM diagram creation has been baselined, significantly more work has been put into the theory of its creation than the execution. Because the theory of EMPIRE is defined by ontological relationships codified in the UOF, the ACSI is formally collaborating with the MSFC ACO on a maximally useful ontological foundry prior to developing corresponding EMPIRE functionality. Effectively, if a relationship or class does not exist in the UOF, it does not exist in EMPIRE. The ACSI is also informally collaborating with the Jet Propulsion Lab to produce functional flight software directly from diagrams in EMPIRE, which should be functional by December 2023. An overview of general EMPIRE capabilities will be provided in a presentation at the 2024 IAC, and an overview of verification by test relationships will be provided in a presentation at the 2024 Institute of Electrical and Electronics Engineers (IEEE) Aerospace Conference. The ACSI is seeking accountability in all steps of UOF and EMPIRE creation, and EMPIRE is slated to be released in part through the United Nations Office of Outer Space Affairs (UNOOSA). Software engineers have been hired, and open-source accessibility is expected by 2024.

Table 3: EMPIRE Database Instantiation Q&A

Question	Example Answer
Are you developing a full spacecraft or a payload?	Full Spacecraft
Is your spacecraft a CubeSat?	Yes
What size is your CubeSat?	12U
What is your mission class?	University
What is your mission type?	Science
What type of science?	Astrophysics
Payload Entry	High Energy Particle Detector[62]
Payload Mass	4.5 kg
Which type of celestial entity is your spacecraft primarily orbiting?	Planet
Which planet is your spacecraft primarily orbiting?	Earth
Do you have a specialty orbit?	Yes
What is your specialty orbit?	Sun-Synchronous
What is your launch type?	Rideshare
Which launch vehicle will your spacecraft be on?	Polar Satellite Launch Vehicle
What is your funding institution?	NASA CSLI
What is your funding program?	Astrophysics Research and Analysis (APRA)
Who is your lead institution?	University of Alabama in Huntsville
What is your risk posture?	Sub-Class D
What is your life cycle?	NASA Single-Mission Robotic Life Cycle
Do you have any optional subsystems?	None
Do you have any partnerships or contractors?	Auburn University
Which subsystems or components is Auburn University responsible for?	Structural Integrity, Thermal Control, Flight Software
Do you have any component vendors?	Blue Canyon Technologies
What subsystem or components is Blue Canyon Technologies providing?	Guidance, Navigation, and Control
Do you have any service providers?	KSAT
What enabling system is KSAT providing?	Ground Control Station

CONCLUSION

Every engineer is a systems engineer, but early-career engineers do not know how to perform professional PM and SE. Due to this lack of PM or SE acumen, those engineers will make decisions ultimately resulting in higher costs, longer schedules, and additional effort for the institutions they work for. Project managers can either spend valuable time and money training early-career engineers to be professional systems engineers, a worthwhile but difficult endeavor not accepted by all discipline engineers, or project managers can employ tools with professional systems engineering ingrained to support common discipline engineer tasks. Ontologies enable engineers performing M&S to accurately plan, describe, and execute M&S activities while at the same time contributing to the central source of truth for a program, but cohesive ontologies are highly complex and notoriously difficult to develop. Software platforms corresponding to rigorously developed ontologies offer strong semantics that both prevent erroneous work and infer additional system architecture descriptions from existing work, reducing the cumulative effort required to realize a project. The purpose of verification by analysis is to ensure a design is compliant to functional and capability requirements, and the cardinal sin of M&S is to make a perfect, pristine model that does not yield the TPM relevant to the requirement. Connecting system architecture descriptions to modeling environments through ontologies prevents this cardinal sin from being committed and supports M&S validation during model input, model execution, and model management. The ultimate goal of this ongoing work is to establish the ontological foundation for a platform to automate M&S, generate documentation, and provide a central source of truth for discipline engineers ill-versed in PM and SE topics to contribute technical information such that the platform semantically infers PM and SE content from technical input. The UOF will serve as the ontological foundation for EMPIRE, and EMPIRE version alpha will be released free and open-source in 2024 for community contribution.

Acknowledgments

The authors would like to thank the Alabama Space Grant Consortium for its tireless support of student space education and workforce development. Tim Canham, Robert Bocchino, and Jeff Levison of the Jet Propulsion Lab are thanked for their continued support of the Alabama CubeSat Initiative and its model-based software development.

References

1. Wasson, C. S., “Systems Engineering: Analysis, Design, & Development”, 2nd edition, ISBN 978-1-118-44226-5, Wiley, 2016.
2. Thomas, L. D., “Systems Test and Verification”, *Engineering Systems*. University of Alabama in Huntsville. Presentation. 2021.
3. Larson, W. J., Kirkpatrick, D., Sellers, J. J., Thomas, L. D., & D. Verma., “Applied Space Systems Engineering”. McGraw-Hill, 2018.
4. Berthoud, L., Swartwout, M., Cutler, J., Klumpar, D., Larsen, J. A., & J. F. D. Nielsen, (2019). “University cubesat project management for success.” *Small Satellite Conference*. 2019.
5. Modeling and Simulation (M&S) Management, DoD Instruction 5000.59, December 2003. PhilArchive copy v3: https://philarchive.org/archive/SEPGFWv3DoD
6. Box, G. E. P., "Science and statistics." *Journal of the American Statistical Association*, 71. 1976. (356): 791–799, doi:10.1080/01621459.1976.10480949.
7. Ferguson, R., Marshall, J., and L. Assadzadeh. "Automated Power Analysis of Onboard Spacecraft Electronics with Model Based Systems Engineering." *2019 IEEE Aerospace Conference.* IEEE, 2019.
8. Halvorson, M. C., Cho, J., Farkas, S., Boyd, L., Cartie,r N., Cole, K., DeAngelo, N., Edwards, T., Middleton, T., Moquin, C., May, B., Kilpatrick, W., Taylor, M., Tsai, H., Cameron, C., and N. Tsolas. “High-Fidelity Spacecraft Thermal Modeling: Synthesis of STK, SPENVIS, MATLAB, Simulink, and Thermal Desktop using Model-Based Systems Engineering.” *Small Satellite Conference*. 2022
9. Chesley, Bruce. Sellers, Jerry. *Applied Model-Based Systems Engineering*. May 2021. Copyright Teaching Science and Technology, Inc. Presentation.
10. Halvorson, M. C., & Thomas, L. D., “Architecture Framework Standardization for Satellite Software Generation Using MBSE and F Prime.” IEEE Aerospace Conference (AERO) IEEE, 2022.
11. Hatley, D. and I. Pirbhai, *Strategies for real-time system specification*. 2013: Addison-Wesley.
12. Hatley, D., Hruschka, P., and I. Pirbhai, *Process for system architecture and requirements engineering*. 2013: Addison-Wesley.
13. Maier, M. W., "Adapting the Hatley-Pirbhai Method for the Era of SysML and Digital Engineering." *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022.
14. Walden, D. D., Shortell, T. M., Roedler, G. J., Delicado, B. A., Mornas, O., Yew-Send, Y., and D. Endler., “INCOSE Systems Engineering Handbook” 4th Edition. 2015.
15. NASA Office of the Chief Engineer, NASA Technical Standard. NASA-STD-7009, “Standard for Models and Simulations”. https://standards.nasa.gov/sites/default/files/standards/NASA/w/CHANGE-1/1/nasa_std_7009a_change_1.pdf
16. ISO 42010:2022, "Software, systems and enterprise – Architecture Description” Geneva, Switzerland, 2022. https://www.iso.org/standard/74393.html
17. Information Technology – Top-Level Ontologies (TLO) – Part 1: Requirements. ISO/IEC 21838-1:2021. ISO/IEC JTC 1/SC 32. October, 2021.
18. Drobnjakovic, M., Ameri, F., Will, C., Smith, B., and Jones, A., “The Industrial Ontologies Foundry (IOF) Core Ontology.” *12th International Workshop on Formal Ontologies meet Industry*, Tarbes, France. September 12th – 15th.
19. Rudnicki, R., Smith, B., Malyuta, T., and Mandrick, W., “White Paper: Best Practices of Ontology Development.” CUBRC, Buffalo, NY, USA. October, 25. [Online]. Available: https://www.nist.gov/system/files/documents/2021/10/14/nist-ai-rfi-cubrc_inc_002.pdf
20. Hohenecker, P. and Lukasiewicz, T., 2020. Ontology reasoning with deep neural networks. *Journal of Artificial Intelligence Research*, 68, pp.503-540.
21. Antoniou, G., van Harmelen, F., “Web Ontology Language.” May, 2003. DOI: 10.1007/978-3-540-92673-3_4
22. W3C SPARQL Working Group. “SPARQL 1.1 Overview.” March, 2013. W3C Recommendation 21. [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
23. O’Connor, M., and Amar., “SQWRL: A Query Language for OWL.” *Proceedings of the 5th International Workshop on OWL: Experiences and Directions*, October 23rd – 24th, 2009.
24. Lamy, J.-B., “Owlready: Ontology-Oriented Programming in Python with Automatic Classification and High-Level Constructs for Biomedical Ontologies.” Artificial Intelligence in Medicine, 80: 28–11. July, 2017. DOI: 10.1016/j.artmed.2017.07.002.
25. Kleppe, A. G., “A Language Description is More than a Metamodel.” *4th International Workshop on Software Language Engineering,* Nashville, TN., U.S., October 1st, 2007.

26. Fondement, Frédéric., "Concrete Syntax Definition for Modeling Languages." November, 2007. Swiss Federal Institute of Technology in Lausanne. DOI: 10.5075/epfl-thesis-3927.
27. Chomsky, N., and Schützenberger, M. P. "The Algebraic Theory of Context-Free Languages." *Studies in Logic and Foundations of Mathematics*, vol. 35, 118-161, 1963. DOI: https://doi.org/10.1016/S0049-237X(08)72023-8
28. Raimond, Y., and Schreiber, G., "RDF 1.1 Primer." June, 2014. W3C Working Group Note 24. [Online]. Available: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
29. Cyganiak, R., and Wood, D., "RDF 1.1 Concepts and Abstract Syntax." February, 2014. W3C Recommendation 25. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
30. Biron, P. V. and Malhotra, A., eds., "XML Schema Part 2: Datatypes Second Edition." October, 2004. W3C Recommendation 28. [Online]. Available: https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
31. McGuinness, D. L., and van Harmelen, F., "OWL Web Ontology Language Overview." February, 2004. W3C Recommendation 10. [Online]. Available: https://www.w3.org/TR/2004/REC-owl-features-20040210/
32. Brickley, D., and Guha, R. V., "RDF Schema 1.1." February, 2014. W3C Recommendation 25. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/
33. W3C OWL Working Group. "OWL 2 Web Ontology Language Overview (Second Edition)." December, 2012. W3C Recommendation 11. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/
34. W3C OWL Working Group. "OWL Web Ontology Language Guide." February, 2004. W3C Recommendation 10. [Online]. Available: https://www.w3.org/TR/2004/REC-owl-guide-20040210/
35. Golbreich, C., Wallace, E. K., eds. "OWL 2 Web Ontology Language: New Features and Rationale (Second Edition)." December, 2012. W3C Recommendation 11. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-new-features-20121211/
36. Motik, B., Patel-Schneider, P. F., Parsia, B., eds. "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)." December, 2012. W3C Recommendation 11. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
37. Uschold, Mike, and Tate, Austin., "Putting Ontologies to Use." *The Knowledge Engineering Review* 13, no. 1. March 1998: 1–3. https://doi.org/10.1017/S0269888998001027.
38. Smith, B., (2018). Applied Ontology: Lecture 1. Introduction to Ontology. University of Buffalo.
39. Information Technology – Top-Level Ontologies (TLO) – Part 2: Basic Formal Ontology (BFO). ISO/IEC 21838-2:2021. ISO/IEC JTC 1/SC 32. November, 2021.
40. Calspan-University of Buffalo Research Center. "Common core ontologies for data integration." https://www.cubrc.org/index.php/data-science-and-information-fusion/ontology
41. Cox, A.P., Nebelecky, C.K., Rudnicki, R., Tagliaferri, W.A., Crassidis, J.L. and B. Smith, "The Space Domain Ontologies." 2021. Calspan-University of Buffalo Research Center.
42. Halvorson, M.C., Moyers, N. and Thomas, L.D., "An Ontology for Prognostic Health Management in Spacecraft Avionics." Annual Conference of the PHM Society. October, 2022. (Vol. 14, No. 1).
43. Johnson, Hamilton E., L. Dale Thomas, and Manuel J. Diaz. "Developing and Testing a Common Space Systems Ontology using the Ontological Modeling Language." *2023 IEEE Aerospace Conference*. IEEE, 2023.
44. Smith, B., "Basic Formal Ontology 2.0: Specification and User's Guide." June, 2015. [Online]. Available: https://purl.obolibrary.org/obo/bfo/reference
45. Aamodt, A., Nygard, M., "Different Roles and Mutual Dependencies of Data, Information, and Knowledge: An AI Perspective on their Integration." *Data & Knowledge Engineering*, vol. 16, no. 3, 191-222. May, 1995. DOI: 10.1016/0169-023X(95)00017-M
46. Rader, J. and Haggerty, L., "Supporting systems engineering with methods and tools: a case study." *28th Asilomar Conference on Signals, Systems and Computers* (Vol. 2, pp. 1330-1334). IEEE, 1994.
47. NASA Office of the Chief Engineer, NASA Technical Handbook. NASA-HDBK-1005, "NASA Space Mission Architecture Framework (SMAF) Handbook for Uncrewed Space Missions". https://standards.nasa.gov/standard/nasa/nasa-hdbk-1005-0. Approved 2021-03-11.

48. Friedenthal, S., Moore, A., and R. Steiner., "A Practical Guide to SysML", 3rd Edition, ISBN: 978-0-12-800202-5, Morgan Kaufmann, 2014.
49. Wolfgang, R., Katz, T., and L. Wheatcraft., "Guide to Verification and Validation" INCOSE-TP-2021-004-01, Rev 1.0. May, 2022.
50. Smith, B., "Basic Formal Ontology 2.0: Specification and User's Guide." University of Buffalo, 2015.
51. Mari, Luca, Anna Chunovkina, and Charles Ehrlich. "The complex concept of quantity in the past and (possibly) the future of the International Vocabulary of Metrology." *Journal of Physics: Conference Series*. Vol. 1379. No. 1. IOP publishing, 2019.
52. NASA Office of the Chief Engineer, NASA Technical Handbook. NASA-SP-2016-6105 Rev2, "NASA Systems Engineering Handbook". https://www.nasa.gov/sites/default/files/atoms/files/nasa_sy stems_engineering_handbook_0.pdf.
53. Oakes, J., R. Bottaand T. Bahill, "Technical Performance Measures," BAE Systems, San Diego, CA, 2004.
54. Standard: Mass Properties Control for Space Systems, AIAA Standards. *American Institute of Aeronautics and Astronautics*, Inc. 2015. https://doi.org/10.2514/4.103858.001
55. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H. H., "The Manchester OWL Syntax." *Proceedings of the 2006 OWL Experiences and Directions Workshop*, November 10th – 11th, 2006.
56. Forsberg, K. and H. Mooz., "The Relationship of System Engineering to the Project Cycle." *Proceedings of the National Council for Systems Engineering* (*NCOSE) Conference*, Chattanooga, TN. Pp 57-65. October, 1991
57. Walden, D.D., "YADSES: Yet Another Darn Systems Engineering Standard." *Proceedings of the 17th Annual INCOSE International Symposium*. San Diego, California. International Council on Systems Engineering.
58. Roedler, G. "Knowledge Management Position." *Proceedings of the 20th Annual INCOSE International Symposium.* Chicago, IL. International Council on Systems Engineering. 2010.
59. Roedler, G. and C. Jones. "Technical Measurement: A Collaborative Project of PSM, INCOSE, and Industry." INCOSE Measurement Working Group. INCOSE TP-2003-020-01.
60. MathWorks., "MATLAB Engine API for Python." [Online]. Available: https://www.mathworks.com/help/matlab/matlab-engine-for-python.html
61. Robinson, J.A., Waid, M.C., Korth, D., Rucker, M. and Renfrew, R., 2019, October. Innovative approaches to using the International Space Station as a Mars transit analog. In International Astronautical Congress (No. HQ-E-DAA-TN74078).
62. Brown, Nichols F. "Space Scientific Instrument Taxonomy (SSIT) Version 2.0." *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022.