An Ontology for Prognostic Health Management in Spacecraft Avionics

Michael C. Halvorson[1], Noah Moyers[2], L. Dale Thomas[1]

[1]*University of Alabama in Huntsville, Complex Systems Integration Lab, Huntsville, AL, 35899, USA*
mch0043@uah.edu
ldt0001@uah.edu

[2]*Auburn University, Auburn, AL, 36849, USA*
ncm0034@auburn.edu

ABSTRACT

Prognostic Health Management (PHM) is the disciplined application of measurement, monitoring, and support strategies to protect structural, electrical, or data entities precluding the failure of measured systems in all phases of operation. Model-Based Systems Engineering (MBSE) can be used to formalize system structure, operations, behavior, and requirements using an Architecture Framework (AF), Process Framework (PF), modeling language, and ontology; whereas the AF, PF, and modeling language may be specific to the program or mission employing MBSE, ontologies may be developed specific to a given domain. The PHM domain considers failure modes, effects, and criticality, and ontological system analysis in this domain can inform system structure, operations, behavior, or requirements. A reference ontology for the PHM domain in spacecraft avionics is presented here including aspects of existing ontologies such as the Basic Formal Ontology (BFO), a Top-Level Ontology (TLO) newly recognized by the International Organization for Standardization (ISO), the Information Artifact Ontology (IAO), and the Space Object Ontology (SOO). A distinction is made between a full PHM domain ontology, which would include many mechanical or electrical systems with myriad purposes, and a PHM domain ontology specific to spacecraft avionics. Present ontological development originated using the parlance and format of BFO and IAO in Stanford University's Protégé software but diverged to include International Union of Pure and Applied Chemistry (IUPAC) terminology and classifications. When interacting with this ontology, engineers seeking to characterize system-specific failure modes, effects, and criticality can query the ontology with their hardware or software entities to obtain failure information specific to the operation of their system in a given operational environment. While this domain ontology is robust, the authors do not claim it to be complete or validated for all spacecraft avionics. It should be considered version one of a useful PHM tool with continual updates occurring after peer review and feedback.

1. INTRODUCTION

All models are wrong, but some are useful (Box, 1976). Ontologies are models of the concepts inherent to a given domain, how those concepts are differentiated taxonomically with increasing levels of specificity, and how those concepts categorically relate to one another. They are agreements on conceptual representation with rigorously scrutinized definitions and are only useful if they serve some accessible purpose (Smith, 2018; Seppälä, Ruttenberg, & Smith, 2017). For Prognostic Health Management (PHM) engineers, that purpose is the characterization of system failure. At the Institute of Electrical and Electronics Engineers (IEEE) Aerospace Conference in March 2022, experts discussed misconceptions of spacecraft PHM inherent to newcomers and veterans alike. Some engineers misunderstood concepts inherent to PHM whereas others debated definitions of established PHM material. It was clear an ontology for PHM in spacecraft avionics, an agreement on PHM entity taxonomy, definitions, and relations with explicit conceptual usage, would be vital as a foundation for intellectual communication, but it needed to be useful.

Not all ontologies are made equal. The term ontology was first used in the *Ogdoas Scholastica* by Jacob Lorhard in 1606, but the first ontology is considered Aristotle's *Categories* (Smith, 2022) wherein the categories of existing things were defined as substance, quantity, quality, relation, place, time, position, doing, having, and being affected. Semantically, these are problematic, but it was a start. In 1998, the Gene Ontology (GO), arguably the most successful and globally utilized ontology (Smith, 2018), was created for

rigorous genome characterization, which spawned the creation of the Open Biological and Biomedical Ontologies (OBO) Foundry. The OBO Foundry acts as a hub for other biological ontologies to be created as spokes wherein OBO entity definitions are utilized in spoke ontologies; it is still in use as of this writing (Jackson, Matentzoglu, Overton, Vita, Balhoff, Buttigieg, Carbon, Courtot, Diehl, Dooley, & Duncan). Biology and genomics received the majority of ontological attention until 2004 when the Web Ontology Language 1.0 was released and applied to Stanford University's Protégé software for ontology development (Musen, 2015). Anyone could create ontologies, so there were quickly many models that were useful only to narrowly defined applications. Nearly all those domain ontologies failed to be useful due to concept redundancy between multiple ontologies, inconsistency in concept definitions, and a lack of common development methodology (Smith, 2018). Centrally, new domain ontologies needed a common Top-Level Ontology (TLO), or common root structure, and methodology for taxonomizing concepts or entities. Thus, the Basic Formal Ontology (BFO) was born (Smith, Kumar, & Bittner, 2005).

The BFO, having received multiple updates since its inception, is primarily divided into continuants and occurrents with a root node of entity. A continuant is defined as, "an entity that exists in full at any time in which it exists at all, persists through time while maintaining its identity, and has no temporal parts." An occurrent is defined as, "an entity that has temporal parts and that happens, unfolds, or develops through time." Ontological definitions for the specialization of a given entity are best written in a specific style (Seppala et al., 2017), e.g. "an [entity] that/which" followed by the aspect of the entity that specializes it from the higher-level entity. Entity, as the root node for BFO, does not have a definition because it does not have a higher-level node to specialize from. Definition creation in an ontology for PHM in spacecraft avionics becomes problematic because most widely accepted definitions for complex engineering concepts stem from the ISO which sometimes provides multiple definitions for a concept and does not adhere to the ontological style of definitions. A visual breakdown of the BFO taxonomy is provided for continuant in Figure 1 and occurrent in Figure 2.

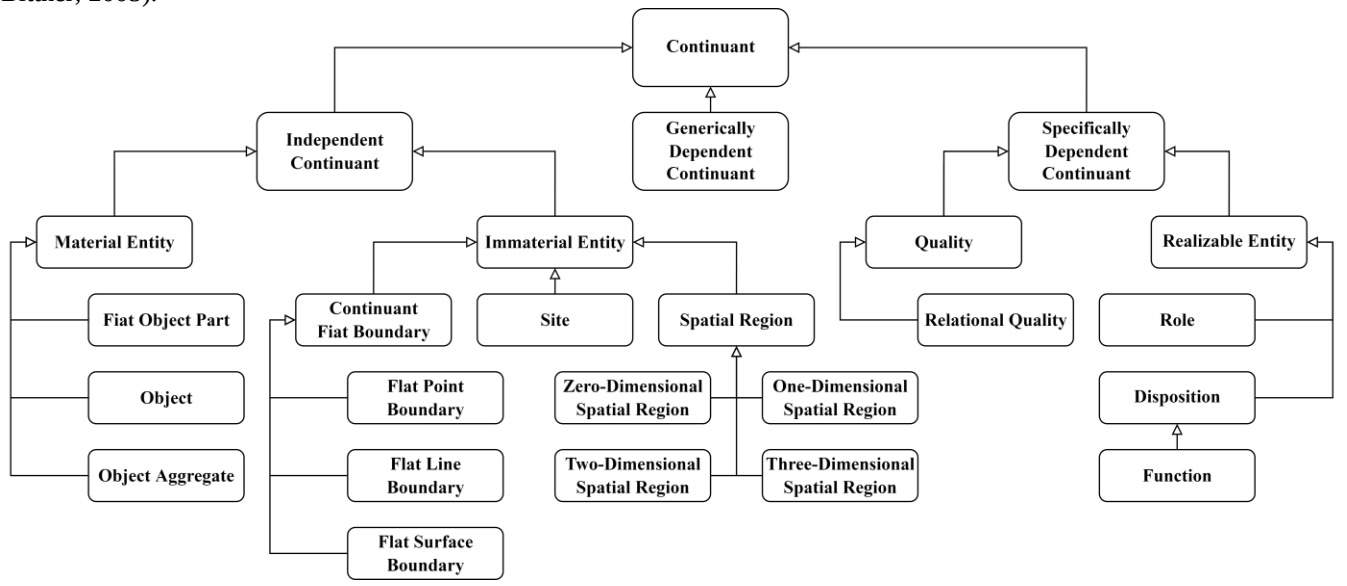


Figure 1: BFO Taxonomy of Continuant

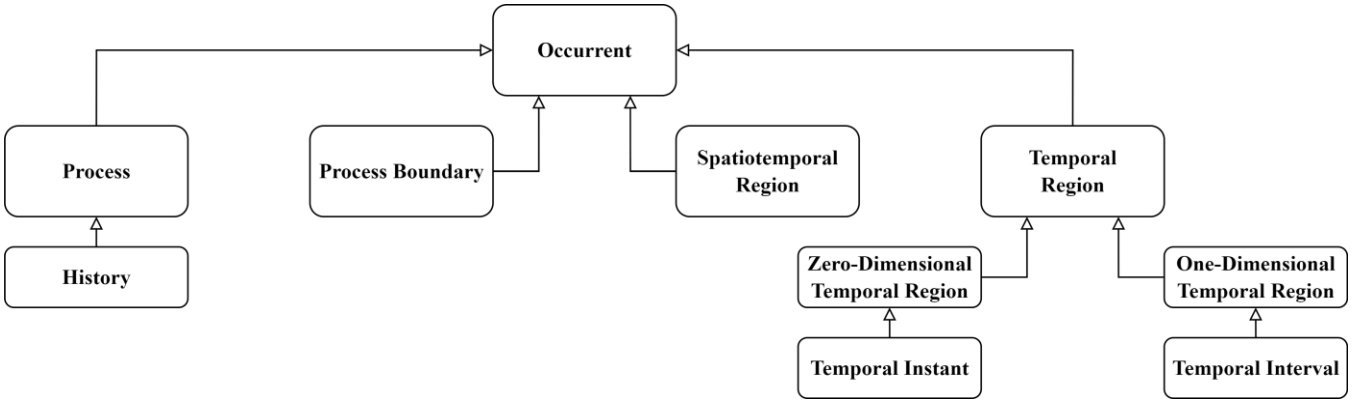


Figure 2: BFO Taxonomy of Occurrent

Continuant is specialized into generically dependent continuant, which includes data entities, independent continuant, which includes objects, and specifically dependent continuant, which includes qualities such as color. Operating systems, protons, circuits, permanence, Mean Time To Failure (MTTF), and induced faults are all examples of continuants. Occurrent is specialized into process, process boundary, spatiotemporal region, and temporal region. Data encryption, particle interactions, startup, and shutdown are all examples of occurrents. These concepts provide the basis for which to define PHM-specific concepts and to relate non-PHM domain ontologies also using the BFO as a foundational ontology.

Avionics here means electrically-interfaced hardware and controlling software, and PHM for spacecraft avionics contains many aspects that could merit domain ontologies of their own. The most important concept is reliability; engineers only perform PHM to ensure systems function as intended. The hardware engineers use to build spacecraft must be taxonomized because it is the subject of mechanical or electrical health monitoring, and spacecraft software is increasingly critical with the advent of components like software-defined radios. Taxonomizing software also allows a relation of software to physical bit representations on memory devices and corresponding software functionality, meaning a fault incurred at one or more bits may result in an error in one or more granular software functions.

The Space Radiation Environment (SRE) contains photons, protons, neutrons, electrons, and heavy ions (Nöldeke, 2015) which interact with spacecraft hardware resulting in quantifiable metrics such as Total Ionizing Dose (TID), Non-Ionizing Energy Loss (NIEL), Charged Particle Heating (CPH), Single Event Effects (SEE), and both surface and deep dielectric charging. Each SRE features different particle types and fluxes, each particle type can cause different radiation effects based on its energy and target material, and each radiation effect can result in differing faults that may or may not manifest as errors.

Ontologies have a primary use case that can be expressed in various methods, but the governing user interaction is, for this application, to assert system hardware, function, and SRE using input queries and generate mission-tailored products such as the relation of software errors to possible hardware faults. Ontologies should exist in the background of user-friendly tools, and querying occurs by using a GUI-based data input and processing wrapper around an ontology built in some ontological development tool such Protégé. Well-made ontologies can also be used as reference dictionaries because of their rigorous definition structures, another use case, but ontological development tools are not user-friendly dictionaries.

A graphical example of relations stemming from a queried ontology is provided in Figure 3 and described here using the Single Event Dielectric Rupture (SEDR) SEE on a Complementary Metal-Oxide Semiconductor (CMOS). A CMOS is a Metal-Oxide Semiconductor Field Effect Transistor (MOSFET). A MOSFET has a gate. A MOSFET gate has an oxide layer, which is a dielectric layer. A dielectric layer has a dielectric displacement field. Protons may exist in a given SRE. A proton strike ionizes a MOSFET gate oxide layer. Ionization of a MOSFET gate oxide layer breaks down the dielectric displacement field. Transistor function is dependent on the dielectric displacement field. Lack of transistor function is a fault. A fault manifests as an error when sensitized by a particular system state. The ontology taxonomy and relationships therefore establish how hardware faults can be initiated in a given SRE, and faults manifesting as software errors due to the lack of hardware functionality can be related to functions in spacecraft software. By querying the ontology in this way, an error manifesting in software can provide insight into hardware health, and PHM is performed.

.

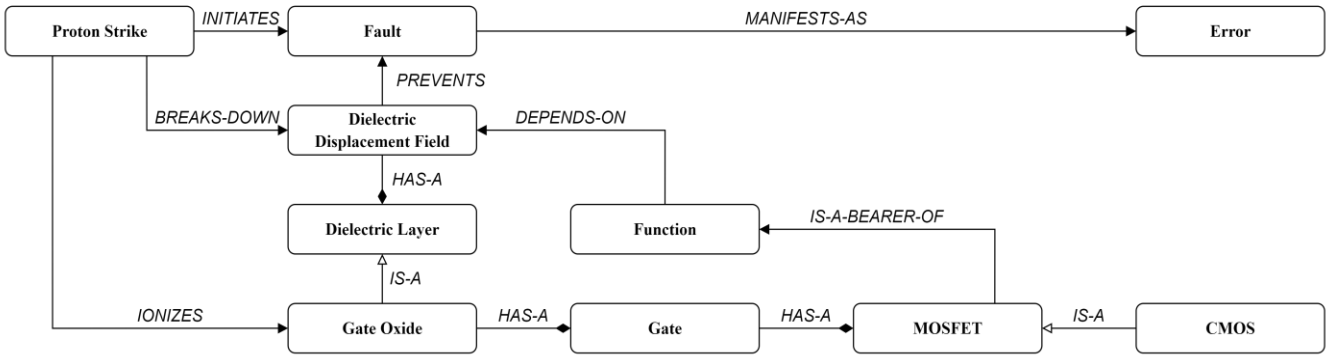


Figure 3: Query Example in the Domain Ontology for Spacecraft Avionics

2. ONTOLOGY DEVELOPMENT METHODOLOGY

There are many ways to create a domain ontology; historically, most result in failure or lack of adoption (Smith, 2018). The Calspan-University of Buffalo Research Center (CUBRC) has been researching ontological development through the Data Science and Information Fusion Group since 2008 (CUBRC, 2022) and has developed the Common Core Ontologies (CCO) method. CCO is rooted in the Basic Formal Ontology (BFO) and extends BFO through 9 other "common" ontologies intended to provide a basis for enterprises to build knowledge networks (CUBRC, 2022). Enterprises using CCO can build domain ontologies using CCO definitions and instantiate enterprise-specific content with predefined relations. Public, non-CCO linkages of domain ontologies with common Top-Level Ontology (TLO) structures are known professionally as the semantic web, a term coined by internet inventor Tim Berners-Lee, and colloquially as Web 3.0 (Smith, 2018). For large enterprises with broad scope, starting with CCO is arguably the best method for creating domain ontologies. However, CCO includes ontologies for currency units, units of measure, and geospatial regions not relevant for the scope of this work. There are outstanding philosophical issues within BFO which present challenges for engineers characterizing hardware interacting with subatomic particles. Entities such as liquids and energy do not yet have an accepted categorization within BFO (Smith, 2018), and the continuant division of material versus immaterial entities does not coincide with how the standard model of particle physics organizes relations between matter and energy.

For this work, the BFO was utilized as a TLO with minor changes. Non-CCO domain ontologies were considered for conceptual inclusion, and domain concepts relevant to PHM in spacecraft avionics were taxonomized. International Union of Pure and Applied Chemistry (IUPAC) classifications were considered, relations between entities were created using the Resource Description Framework (RDF) format, and definitions were compared to ISO standards. While ISO, International Electrotechnical Commission (IEC), and IEEE standards, specifically ISO/IEC/IEEE 24765:2017: Systems and Software Engineering – Vocabulary (ISO, 2017), were recognized as accessible sources for ontological definitions, discussed is a lack of rigor and clarity in some ISO terminology content.

The two non-CCO ontologies considered for inclusion were the Space Object Ontology (SOO) for its characterization of the relation between spacecraft parts and functions (Cox, Nebelecky, Rudnicki, Tagliaferri, Crassidis, & Smith, 2016) and the Information Artifact Ontology (IAO) for its representation of software entities (Smith, Malyuta, Rudnicki, Mandrick, Salmen, Morosoff, Duff, Schoening, & Parent, 2013). There exists an Information Entity Ontology within CCO, but without using the rest of CCO it is baseless.

2.1. Using the Basic Formal Ontology

ISO/IEC 21838-1 defines the requirements of a TLO (ISO, 2021), and ISO/IEC 21838-2 recognizes the BFO as a TLO (ISO, 2021). Both the BFO and its derivation into the present PHM domain ontology for spacecraft avionics are reference ontologies per ISO/IEC 19763-3 (ISO, 2020), meaning all child entities have at most one parent. When child entities have more than one parent or when instances of entities are mixed with ontological categories, the ontology becomes an application ontology. Reference ontologies may be reused and updated as needed for use by the general community whereas application ontologies are specific to a narrow field or problem and are generally not reusable.

Because the present PHM ontology does not strictly adhere to BFO or CCO structures, it is not conformant to the Metamodel Framework for Interoperability (MFI). Being MFI-compatible per ISO/IEC 19763-3 allows direct merging of disparate ontologies because they share the same TLO or broader CCO structure. If engineers at disparate companies create distinct ontologies using a common, MFI-compatible TLO without changing the TLO, merging ontologies in Protégé is easily performed using the "refactor" feature. A built-in reasoner function allows existing relations to be applied to new entities, and the utility of both ontologies is preserved and expanded. Becoming MFI-compatible is a goal of the present ontology, but the addition of IUPAC intensive and extensive quantities, widely accepted and useful to engineers, breaks the standard BFO structure and precludes MFI compliance. It is the intent of this work to useful first and conformant second.

2.1.1. What's the Matter with BFO?

For the purpose of taxonomizing concepts related to the PHM of spacecraft avionics, the BFO was first augmented to represent particle physics entities. Subatomic particles, a consistent and inexorable source of spacecraft software errors due to fault creation in electronic hardware, are problematic for BFO because the distinctions within independent continuant are material entity and immaterial entity. The etymology of the word "material" sources the Latin *materialis* and translates to "of or belonging to matter". It can be understood by persons of ordinary capacity that material in the BFO means composed of matter and immaterial means not composed of matter, but that is not how the standard model of particle physics is organized (Elert, 2022). Without detailing definitions of particle physics terminology outside the scope of this work, elementary particles are logically separated into fermions and bosons, shown in Figure 4. To reiterate, the particles of interest for PHM in spacecraft avionics are photons, protons, neutrons, electrons, and heavy ions.

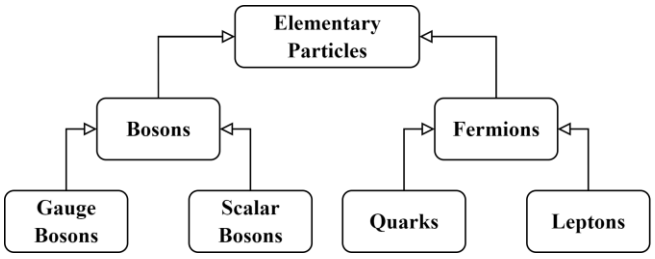


Figure 4: Particle Physics Taxonomy Overview

Bosons are divided into gauge bosons and scalar bosons. Photons and gluons, which do not have mass, are types of gauge bosons. W and Z bosons, which do have mass, are also types of gauge bosons. Whether the particle has mass or not is not the base differentiator for taxonomization in particle physics; actual differentiators such as spin and adherence to Pauli's exclusion principle are outside the scope of this work. Fermions include particles such as quarks and leptons, with electrons being a type of lepton. Neutrons and protons are considered composite fermions, and heavy ions could be composite fermions or composite bosons based on how many particles comprise them. It is clear the distinction of material versus immaterial is insufficient for the purposes of an ontology characterizing interactions of subatomic particles. To remedy this problem, taxonomies for elementary particles, e.g. photons and electrons, and composite particles, e.g. neutrons and protons, were created. The object category under material entity includes an atom as an example in its description, so heavy ions were considered part of the object category despite elemental atoms technically being composite particles. This taxonomy addition, which does not render the present ontology MFI-incompatible because it merely adds to the BFO, is visually represented in Figure 5.

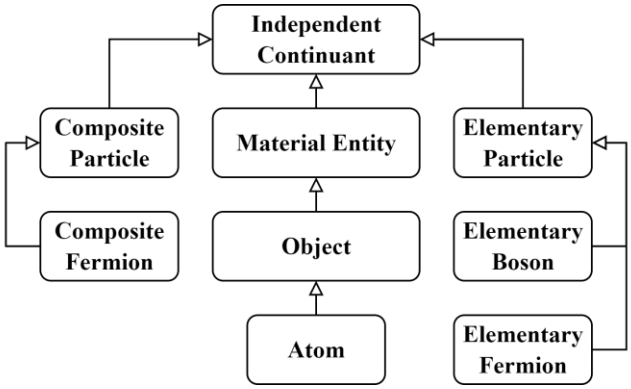


Figure 5: Particle Organization as Independent Continuants

The decision to assert an atom as an object stems from the concept of causal unity, described at length in the BFO 2.0 Specification and User's Guide (Smith, 2015), wherein an object is described as "a maximal causally unified material entity". Causal unity is characterized in the parlance of Ingarden (Smith & Brogaard, 2003; Ingarden, 1983), meaning material entities are, "both structured through a certain type of causal unity and maximal relative to this type of causal unity." Three paradigms of causal unity are offered: unity by the presence of a covering membrane, unity by existence as a solid portion matter, or unity as an engineered artifact. An atom counts as an object here via unity by existence as a solid portion of matter. Describing an atom as a "solid" is limiting when considering phases of matter, but atoms may analogously be regarded as matter by means of tangible continuum identity. Considering atoms in various phases of matter could be composite particles, taxonomizing engineering-relevant entities such as liquids, plasma, and other forms of energy may require fundamental BFO restructuring without the presence of matter as a fundamental delineator.

While causal unity influences independent continuant categorization, causality theory itself is not incorporated into BFO (Smith, 2015). Alongside object, the BFO features an independent continuant subcategory for object aggregate, e.g. The Beatles or five apples. A Printed Circuit Board (PCB), MOSFET, or any other avionics circuitry is herein considered an object via unity as an engineered artifact and not an object aggregate. This decision is intended to simplify hardware taxonomies in the present ontology, though a dissenting case could be made within reason.

2.1.2. Qualifying and Quantifying in BFO

The preceding material distinctions may be substantiated for a domain ontology on PHM in spacecraft avionics, but they do not preclude MFI compliance. Subatomic particle characterization is an addition to BFO allowed of any domain ontology, and asserting spacecraft hardware as objects in lieu of object aggregates for simplicity is within reason. A central, limiting problem regarding the application of BFO to this domain exists in the categorization of quality, and it is here asserted that this is a shortcoming of BFO in a holistic sense. Humorously, the missing piece required of this PHM ontology was described in Aristotle's *Categories*; it is the concept of quantity.

BFO defines quality as, "a specifically dependent continuant that…does not require any further process in order to be realized." Axiomatically, "if an entity is a quality at any time that it exists, then it is a quality at every time that it exists." Both the definition and axiom could be used to describe quality or quantity, but provided examples for quality include the color of a tomato, the shape of a person's nose, and the ambient temperature of a portion of air. The problem with the definition of quality is, when considering the usefulness of a model, not all qualities can be measured as properties. The Stanford Encyclopedia of Philosophy describes properties as "those entities that can be predicated of things or…attributed to them." Quality is even used as a synonym for property (Orilia, & Paoletti, 2020). Although neither BFO defines quality in a way that excludes quantity nor does Stanford define property in a way that excludes quality or quantity, the concept of measurability is directly consequential to the

utility of a practical engineering ontology. A quantity is not simply a measurable quality. A quality for a data item might be its data format, and it exists with that quality without needing any further process after it undergoes a data transformation process. A memory device might have a quality of volatility, and it is said to be the bearer of that quality. Data formats and volatility innately differ from physical properties, described by Burgin (2016) as "any property that is measurable, whose value describes a state of a physical system," in that they do not have enumerable values. Data formats could be described by a data type, but neither data types nor volatility are evaluable. When properties are formally considered to be quantities, not qualities, and are ascribed divisions of digital quantity and physical quantity, the established IUPAC concepts of intensive quantity and extensive quantity can be utilized. Per the IUPAC Compendium of Chemical Technology (McNaught, 2019), an intensive quantity is a, "physical quantity whose magnitude is independent of the extent of the system" and an extensive quantity is a, "physical quantity whose magnitude is additive for subsystems." Extensive quantities would include properties such as resistance, charge, and energy, thus solving BFO's energy allocation problem, and intensive quantities would include properties such as voltage, temperature, and density. The following definitions are prescribed in ontological format.

> Quantity: a specifically dependent continuant that does not require any further process in order to be realized and can be prescribed a numeric value.
>
> Digital Quantity: a quantity whose value describes a state of a digital system.
>
> Physical Quantity: a quantity whose value describes a state of a physical system.
>
> Extensive Quantity: a physical quantity whose magnitude is additive for subsystems.
>
> Intensive Quantity: a physical quantity whose magnitude is independent of the extent of the system.

Because entities such as temperature would not be allocated to quality as in the BFO, this domain ontology is not MFI-compliant. A visual reference is provided in Figure 6.

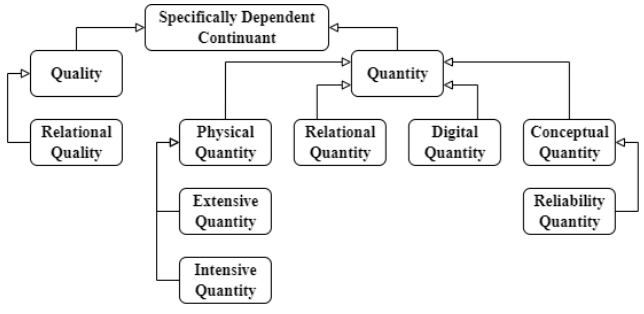


Figure 6: Specifically Dependent Continuant Entities in the PHM Domain Ontology for Spacecraft Avionics

2.1.3. BFO Continuant Usage

Continuants exist and are of three types: independent continuant, generically dependent continuant, and specifically dependent continuant. An independent continuant bears qualities, quantities, and realizable entities; other entities inhere within it. Independent continuant in the PHM domain ontology for spacecraft avionics includes composite particles, elementary particles, immaterial entities such as spatial regions, and material entities such as objects. A generically dependent continuant depends on some independent continuant to exist, but it is not specific to a single entity or instance of that entity. Data files, which may or may not exist in the exact same form on one or more computers, are examples of a generically dependent continuant. The only sub-entity for generically dependent continuant in this domain ontology is information content entity, a title reused from the IAO, which describes various data representations within spacecraft software. A dependent continuant inheres in or is borne by other entities; qualities such as data formats, quantities such as temperature, and realizable entities such as roles are examples of specifically dependent continuants in this domain.

In the BFO, functions are afforded a loose but versatile definition (Smith, 2015). These functions are, "realized in processes called *functionings*."

> Function: a disposition that exists in virtue of the bearer's physical make-up [, i.e.] something the bearer possesses because it came into being either through evolution…intentional design…[or] in order to realize processes.

From a philosophical view, this definition is functional. The BFO authors often liken function to purpose. From an engineering perspective, it is incomplete because functions are inherently unitless continuants. Many engineered artifacts have multiple functions, thereby diluting purpose lest the purpose is to be multifunctional. Wasson (2016) makes the analogy of a capability to a vector, stating, "a *capability* (vector) is characterized by a *function* (direction) and a *level of performance* (magnitude)." The distinction is made due to prevalent misconceptions between the concept of functional analysis and capability analysis in systems engineering leading to opaque requirements derivation. Wasson's juxtaposition of function versus capability is represented in the present domain ontology in that a level of performance is a realizable entity evaluating a quantity that can be allocated to a function to represent a capability. Capability is not a defined entity separate from function. Unitless functions, e.g. functions corresponding to all software processes, can still exist without a level of performance, but capabilities, e.g. executing an Error Correction Code (ECC) within five seconds, may be distinguished by allocating a level of performance to a function.

2.1.4. BFO Occurrent Usage

Occurrents happen and are of four types: process, process boundary, spatiotemporal region, and temporal region. Processes contain all software operations, among other entities, in a sub-entity termed planned process originating from IAO. Every process exists within a one-dimensional temporal region, such as startup, and is bounded by two zero-dimensional temporal regions, such as startup initialization and startup completion. Spatiotemporal regions bound a temporal existence of a spatially dimensional entity, e.g. the spatial region occupied during the mission lifespan of a spacecraft. An ion strike, a process in the present domain, would exist in a one-dimensional temporal region bounded in time by two zero-dimensional temporal regions. If striking a MOSFET, it would generate a charge funnel occupying some spatiotemporal region (Baumann, 2004). If the spatiotemporal region of the charge funnel overlaps with the spatiotemporal region of MOSFET operation, a SEE may occur. Therefore, SEEs are logically occurrents in the present domain which may or may not result in a fault, which is a continuant.

2.2. Information Artifact and Space Object Ontologies

The IAO and SOO were both considered in the creation of the present domain ontology. Definitions in the IAO were directly utilized herein where applicable, but some definitions, such as that of information content entity, "a generically dependent continuant that is about some thing," could be given more rigorous characterization future work. The IAO was not created with space applications in mind. It contains entities such as postal code, email address, and various textual components of a conference or journal publication. Those entities deemed irrelevant to the purview of PHM for spacecraft avionics were removed, and various terminology regarding "measurement datum" intended for a journal publication were replaced with quantity terminology for specifically dependent continuants and a new classification taxonomy for data items. Where the IAO displayed significant utility was its organization of directive information entities including conditional, language, objective, and plan specifications. Event and time triggers exist as conditional specifications; algorithms, software, and Error Detection and Correction (EDAC) codes exist as plan specifications. A data format specification was added.

One concept that vaguely existed in IAO and was removed was the concept of units represented as data items. When data is downlinked from a spacecraft, the Orbital Data Message (ODM) often contains values understood by mission operators to be certain units, e.g. energy measured in MeV instead of keV. Units are not included in the ODM because representing units wastes valuable bits, and message bit structures are interpreted to be certain units in the mission operations center. Sometimes, such as in the case of the Mars Climate Orbiter, not representing units in downlinked ODMs is bad practice. The failure of the Mars Climate Orbiter due to a mismatch in metric units and incessantly useless imperial units was a dramatic case of unit representation relevance (Mishap Investigation Board, 1999), but quality assurance and accurate data interpretation should prevent this problem. Units are not included in the present domain ontology.

The SOO is an extension of both BFO and CCO and is therefore not directly compatible with the present ontology. The SOO also contains entities such as spacecraft mission plan and spacecraft mission objective not required in the present domain ontology, which emulates the SOO in that spacecraft parts are considered bearers of artifact functions.

3. CONCEPTS IN PHM FOR SPACECRAFT AVIONICS

Information relevant to a PHM domain ontology when applied to spacecraft avionics includes taxonomies for avionics hardware and software, general avionics processes such as commanding and scheduling, PHM-specific avionics processes such as EDAC, SRE entities, radiation physics terminology, interactions between space radiation and spacecraft avionics, faults resulting from space radiation effects, errors induced in hardware and software, and radiation effect mitigation strategies. Examples of relationships between taxonomic entities are provided for select material in RDF format; "is a" relations are implied by the taxonomy. The ontology taxonomy is visually provided in the Appendix, and the full ontology including relationships is available upon request to the authors. Definitions in the present ontology were cross-referenced with ISO standards, but few ISO definitions were usable due to a critical lack of definition consistency within ISO standards.

3.1. Avionics Hardware Taxonomy

Spacecraft hardware is considered an object in the BFO sense and contains electronic and mechanical hardware for the present domain. Electronic hardware is subdivided into circuitry and computer hardware such that circuitry hardware contains low-complexity electronics such as analog and digital circuits whereas computer hardware contains high-complexity electronics such as controlling, processing, programmable, and storage devices. Fundamental circuitry such as transistors, diodes, and their subtypes, analog circuitry such as amplifiers, single-function digital circuitry such as encoders and latches, multi-function digital circuitry such as arithmetic logic units, and mixed-signal circuitry such as Analog-to-Digital Converters (ADC) are all considered low-complexity electronics under the entity circuitry hardware. Computer hardware elevates the level of abstraction to contain many elements of circuitry hardware; Field-Programmable Gate Arrays (FPGA) and storage devices exist in this category.

Rigorously defining taxonomical entities is paramount to ontological utility, and the common forum for fastidious engineers to obtain definitions is ISO standards. In some cases, ISO standards fail to provide adequately utilitarian clarity. The National Aeronautics and Space Administration (NASA) has the same inter-standard definition consistency problem (Halvorson & Thomas, 2022). As an example, ISO/IEC/IEEE 24765:2017 defines Read-Only Memory (ROM) as a "non-volatile semiconductor storage device, from which data cannot be removed once it is written" and Random-Access Memory (RAM) as a "volatile semiconductor storage device which allows data to be written or accessed in approximately the same amount of time, regardless of the data's physical location" (ISO, 2017). The presented definition of ROM exclusively describes Mask Read-Only Memory (MROM) and Programmable Read-Only Memory (PROM) but fails to consider modern ROM storage devices such as Erasable Programmable Read-Only Memory (EPROM) and Electronically Erasable Programmable Read-Only Memory (EEPROM). The presented definition of RAM exclusively describes types of Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM) while excluding all types of Non-Volatile Random-Access Memory (NVRAM), such as EEPROM and NOR flash memory. When considering EEPROM and NOR flash are both types of ROM and types of NVRAM, a logical flaw in the typical categorizations of storage devices is apparent from the categorically assumed equivalence of ROM and RAM. The term ROM is attempting to describe the quality of a storage device regarding its inability to be rewritten whereas the term RAM is attempting to describe the quality of a storage device access method regarding its ability to access any address inconsecutively without sacrificing efficiency. Typical categorizations often lack the term Sequential-Access Memory (SAM), hierarchically equivalent to RAM, describing the quality of a storage device access method regarding its inability to access any address inconsecutively without significant impacts to efficiency. Accounting for these discrepancies, to present a more realistic storage device taxonomy than that implied in ISO/IEC/IEEE 24765:2017, directly categorizing storage devices under RAM, SAM, and ROM has been avoided in favor of allocating qualities to particular types of storage devices. Examples of relations as ontological triples in RDF format for avionics hardware are provided in Table 1. The word “object” in RDF format refers to the target of the relation, not object in the sense of BFO material entity with an associated causal unity.

Table 1: Avionics Hardware Relation Examples

Subject	Predicate	Object
computer hardware	hasPart	circuitry hardware
data process	occursIn	processing device
memory management unit	providesAddressTo	storage device controller
storage device	participatesIn	data process

3.2. Avionics Software Taxonomy

All data and data representations are considered to be generically dependent continuants in BFO and are subcategorized under information content entity in IAO; an information content entity may be a directive information entity, data item, identifier, or symbol in the present domain.

IAO introduced conditional, language, objective, and plan specifications as subtypes of directive information entity, and both software, a set of instructional data used to operate a computer system, and algorithms, the ordered content of instructional data relevant to software, are considered types of plan specifications. Software subdivides into software application, software library, software method, software module, and software script. Software application contains fully executable software such as application software, software such as F’ (Bocchino, Canham, Watney, Reder,& Levison, 2018) designed to achieve the goals of the user, and system software, software designed to support application software such as the operating system. Software module contains independent software components that can be linked together to form a portion or the entirety of a software application, such as the operating system kernel, file system, and scheduler. Software method contains individual software subroutines such as commands and system calls that can be linked together to form a portion or the entirety of a software module. Algorithm subdivides into numerous categories described by data interactions such that the intent of a particular software application, software module, or software method can be identified. Categorical specializations of software related objective specifications, functions, and processes were created in an identical manner.

Forms of data intended for software interpretation instead of direct execution are considered specializations of data item, including singular datum, non-singular data, and aggregate data. Singular datum contains data intended for individual interpretation such as quality, quantity, and setting parameters. Non-singular data contains data intended for unified interpretation wherein individual interpretation of data segments still provides value, such as a Centrally Registered Identifier Registry (CRID), setting configuration, and state vector. Each CRID, possibly having set or variable cardinality and ordinality depending on the interpretation strategy, is linked to one or more identifiers wherein the usage of complex data structures such as stacks, queues, or linked lists allows this linkage to occur. By extension, it is not necessary for every complex data structure to have its own CRID so long as the complete data structure is accessible in some manner using an associated identifier. Aggregate data contains data intended for unified interpretation wherein individual interpretation of data segments is less meaningful, such as a data log, raster graphic, or orbital data message. Avionics software relation examples displaying ontological connectivity are provided in Table 2.

Table 2: Avionics Software Relation Examples

Subject	Predicate	Object
scheduler	hasPart	process registry
process registry	hasQuality	ordinality
command	hasResult	objective specification
data objective specification	isTargetOf	data function
data function	isRealizedIn	data process

3.3. PHM-Specific Avionics Software Taxonomy

Algorithmic implementations of EDAC are necessary to prevent the occurrence of interpretation or transmission of corrupt data. EDAC implementations consist of the addition of redundant data to a given message in the form of ECC. Error Detection Codes (EDC) allow for the detection of the presence of errors whereas ECCs allow for both the detection and correction of errors in a given piece of data. Both EDCs and ECCs are limited in capability respective to their implementations (Heidergott, 2004). Types of EDCs include the repetition code, checksum, cyclic redundancy check, longitudinal redundancy check, and transverse redundancy check. Types of ECCs include linear block codes and convolutional codes. Data scrubbing is an EDAC mechanism of invoking ECC on a particular address in a storage device that is both readable and writable regardless of volatility; data scrubbing techniques include demand scrubbing and patrol scrubbing (Heidergott, 2004). Demand scrubbing is automatically invoked by a storage device controller upon receiving a request to read from an associated ECC-capable storage device before sending the requested data. Patrol scrubbing is invoked from software as a request to a storage device controller associated with an ECC-capable storage device to iteratively perform demand scrubbing on each address without returning the data at said address. The usage of both demand scrubbing and patrol scrubbing slows the accumulation of uncorrectable multi-bit errors.

Another EDAC mechanism pertains to majority voting paired with computational redundancy implementable in either hardware or software. Majority voting involves the repetition of an identical operation wherein the results of each operation are provided to a hardware or software construct capable of determining the highest modal input and returning that input as a single output. Implementations in hardware involve the utilization of transistor-transistor logic in the form of digital circuits composed of basic logic gates and multiplexers (Choudhary, Balasubramanian, Varghese, Singh, & Maskell, 2019) such that the internal data bus is capable of supporting the bit-width of each input and output. Implementations in software involve the utilization of an appropriate algorithm such as the Boyer-Moore majority vote algorithm, an optimal approach removing the necessity for input sorting (Boyer & Moore, 1991). Table 3 details relations for PHM-specific software enabling software functional traceability.

Table 3: PHM-Specific Software Relation Examples

Subject	Predicate	Object
storage device controller	invokesFunction	detect and correct data error
detect and correct data error	hasAlgorithm	error detection and correction algorithm
error detection and correction algorithm	corrects	undesired system state

3.4. Space Radiation Environment Entities

Space is radioactive, containing Solar Energetic Particles (SEP), Galactic Cosmic Rays (GCR), Van Allen Belt (VAB) trapped particles, and locally generated neutrons from sources such as the surface of the moon (Nöldeke, 2015). All SREs can be described by particle types, including protons, neutrons, electrons, photons, and heavy ions, and the corresponding energy of those particles. Different locations will feature differing particle profiles at differing energy levels; Jupiter has a magnetosphere stronger than that of Earth and therefore traps more particles with higher energies in its VAB. Deep space, by contrast, features negligible magnetospherically trapped particles but higher GCR particle fluxes. For effect examples, a SRE with high particle flux but low particle energy, such as Earth's VAB, will cause high TID. A Coronal Mass Ejection (CME) from the Sun releases protons causing high NIEL, which manifests as solar array efficiency degradation. GCR flux increases outside of Earth's magnetosphere causing additional SEEs due to very high particle energies penetrating spacecraft shielding. An ontology detailing these concepts must define particle types, properties, microscopic interactions, macroscopic interaction effects, SRE locations, and solar events.

Particle taxonomy is described by Figure 4, representing the relevant particles of photons, protons, neutrons, electrons, and heavy ions under broad categories. Intensive radiative quantities, such as radiative wavelength, electromagnetic absorptivity, and linear energy transfer, and extensive radiative quantities, such as flux and intensity, are represented as quantities. One unusual assertion of the present domain is that radiative dose is an extensive quantity. The units of dose are J/kg, and usually when a parameter is per kg, e.g. specific internal energy or specific heat capacity, that quantity is intensive. However, adding more flux or more material would increase the dose absorbed by the material, so dose, TID, and NIEL are all considered extensive quantities. Instances of entities are not to be represented in reference ontologies, so only deep space and planetary magnetosphere were included as three-dimensional spatial regions in lieu of planet-specific locations. Magnetic and electric fields are considered realizable entities, a type of specifically dependent continuant.

A process specialization of radiative process was created to include processes such as solar event, photon interaction, particle interaction, and SEE. Solar event includes supernova, solar flare, and CME. Photon interactions are described by Lechner (2018), and particle interactions include single-particle processes such as brehmstrallung and induced processes such as dielectric breakdown. Finally, SEEs are specified into recoverable and destructive effects, with recoverable including sensor noise, Single Event Functional Interrupt, Single Event Hard Error, Single Event Transient (SET), and Single Event Upset (SEU) and destructive including Single Event Burnout, Single Event Gate Rupture, Single Event Latch-Up, and Single Event Snap-Back, and SEDR.

Relations for the SRE must relate environments to particles, particles to particle properties, particles and particle properties to microscopic interactions, microscopic interactions to macroscopic effects. Effects of interest include physical properties such as TID, NIEL, and SEEs and digital properties such as bit error rate. Examples of SRE relations allowing characterization of particle interactions and effects on hardware circuitry are detailed in Table 4.

Table 4: SRE Relation Examples

Subject	Predicate	Object
planetary magnetosphere	contains	proton
proton	hasQuantity	energy
proton	hasInteraction	ion strike
ion strike	breaksDown	dielectric displacement field
ion Strike	initiates	induced electrical fault

3.5. Faults and Errors

ISO/IEC/IEEE 24765:2017 provides five definitions for fault: (1) manifestation of an error in software (2) incorrect step, process, or data definition in a computer program (3) situation that can cause errors to occur in an object (4) defect in a hardware device or component (5) defect in a system or a representation of a system that if executed/activated could potentially result in an error (ISO, 2017). Lack of definition in ontological format aside, it is unclear from the blending of these definitions if a fault is the manifestation of an error, the error itself, or the initiating defect. One might argue if five definitions are needed for a given term when used in five contexts, the term itself may need qualifying verbiage to distinguish definitions per contexts, e.g. fault versus latent vault. ISO/IEC/IEEE 24765:2017 provides an equally inadequate characterization of errors for use in the present domain in three definitions: (1) human action that produces an incorrect result (2) difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition (3) erroneous state of the system (ISO, 2017).

An excellent characterization of faults in the context of radiation effects and soft errors in integrated circuits and electronic devices is provided by Heidergott (2004),

> A fault may be classified by its nature, duration, and extent; the duration of a fault may be transient, intermittent, or permanent...errors are the undesired system states caused by the fault…a fault in a system does not necessarily result in an error; a fault may be latent in that it exists but has not resulted in an error; the fault must be sensitized by a particular system state and input conditions to produce an error.

Building on Heidergott's characterization, fault and error are ascribed the following definitions for the present domain,

> Fault: a realizable entity initiated transiently, intermittently, or permanently by an event that may or may not manifest as an error when sensitized by a particular system state.
>
> Error: a realizable entity that manifests as an undesired system state, a loss of subsystem function, or an inability to provide a service.

The nature of a fault can be described as latent or induced. An undiscovered fault in software programming or an electrical defect in a MOSFET would be considered a latent fault. A fault initiated as a consequence of a SEE would be considered induced by radiation. Whether that radiation-induced fault is recoverable or destructive is simply a characterization of the duration of the fault as transient or permanent. If a permanent fault is induced on-orbit, functionally that fault is equivalent to a latent fault. The extent of the fault refers to the propagation of the fault across system interfaces. A SET might induce a SEU, which is functionally understood as a bit flip, e.g. a zero becoming a one. If demand scrubbing is performed on the memory device when the faulty bit is read, an ECC would prevent downstream propagation of the fault into the software, and the extent of the fault is contained. Examples of relations specific to faults and errors are provided in Table 5.

Table 5: Fault and Error Relation Examples

Subject	Predicate	Object
single event upset	hasQuality	fault extent
single event dielectric rupture	hasQuality	fault nature
induced electrical fault	manifestsAs	loss of function

3.6. Radiation Effect Mitigation

It is well-established that a fault-tolerant system makes dedicated use of fault avoidance, fault masking, detection of compromised system operation, containment of error propagation, and recovery to normal system operations (Somani & Vaidya, 1997). Each of these is enabled in the present ontology.

Fault avoidance primarily occurs through four strategies: reduction of particle energy that would induce a fault, i.e. radiation shielding, avoiding deleterious particles by means of orbital element causation, powering down sensitive electronics when encountering deleterious particles, or using electronics hardened to deleterious radiation, i.e. "space-rated" or "rad-hard" devices. Dedicated radiation shield technology such as Z-graded radiation shielding (Thomsen, Kim, & Cutler, 2015) does exist, but any material entity could be a radiation shield. A role of radiation shield was created that a material entity could become the bearer of. Qualities for powered-off and powered-on were created that can be ascribed to hardware components encountering deleterious radiation. The quality rating was added to describe derated components operating below the rated stress level in a given operational environment (Ebeling, 2004), and a relational quality for mounting was included to describe mounting hardware to an insulator as one example of a physical radiation hardness trait. In modern spacecraft engineering, many programs are using Commercial Off The Shelf (COTS) parts without the Military Specification (MIL SPEC) designation; some experts assert COTS is equally viable to MIL SPEC for many applications due to increased understanding of radiative effects on spacecraft electronics (Leitner, 2022).

Fault masking refers to EDAC, EDC, ECC, majority voting, scrubbing, and redundancy. Redundancy, under the umbrella of system reliability, can be informational, spatial, repetitively temporal, temporally delayed or any combination therein. Informational redundancy signifies the existence of the same data item multiple times on a single storage device whereas spatial redundancy signifies the existence of the same data on two or more storage devices. Spatial redundancy could also signify multiple processors or other hardware in a majority voting schema. Repetitive temporal redundancy signifies the execution of the same process on the same hardware or software three or more times and majority voting the output, and delayed temporal redundancy signifies the execution of the same process on n spatially redundant hardware devices with the execution of the process on each sequential hardware device being delayed by an incrementing number of clock edges. Delayed temporal redundancy guards against faults induced by SETs in majority voting schemes, though it at minimum triples the execution time for the hardware and is dependent upon the hardware's Window of Vulnerability (Fouillat, Pouget, Lewis, Buchner, & McMorrow, 2004). Redundancy is considered a relational quality in the present domain ontology.

Reliability constructs such as Mean Time To Failure (MTTF), Failure In Time (FIT), and minimum life are also represented in the present domain under the quantity reliability quantity. Traditionally, a Reliability Block Diagram (RBD) follows a success perspective to reliability whereas a fault tree follows a failure perspective. Fault trees determine cut sets, or a set of basic events whose simultaneous occurrence ensures that the system failure event occurs (Kwatny, 2007). All the taxonomies, functions, and relationships described herein are ontologically useful because they identify and characterize cut sets resulting in errors that can be used to prognosticate and diagnose spacecraft avionics after an error has been detected. More succinctly, when an error arises or is predicted, the causes, preventative measures, and potential solutions, even if multitudinous, are known.

Detection of compromised system operation is only half of PHM; the other half is to predict it. Errors such as loss of subsystem function or inability to provide a service are easily detected as they result in failure messages, but undesirable system states are more difficult to detect. For parameters, specifically quantity data as sensor data or setting parameters, it is possible to evaluate the data to be within prescribed upper and lower bounds, whether that data is in the correct format, if it is the correct data type, or if it the return is null. Parameters can also be monitored to predict if they will drift above or below the bounds. A useful data operation for PHM is to compare parameter values during startup and shutdown to a running log of startup and shutdown values. In the present domain, the parameter boundary is included as a conditional specification, startup and shutdown are one-dimensional temporal regions, the divisions between startup boundary, nominal operation, and shutdown are each included as a process boundary, and the log of parameters during startup and shutdown is represented by the aggregate data entity, data log.

Containment of error propagation is performed by confining the error to the component or subsystem in which the fault occurred (Heidergott, 2004), and containment is most useful at the lowest level possible. Each subsystem can define containment boundaries which are logically executed by continual verification of data value, structure, and format. Error detection should be decoupled from error correction to avoid unwanted persistence of one or more errors; halting a process signaling an error and performing a power cycle on one or more system elements is warranted. In recovery to normal system operation, recovery to a previous state thought to be correct is risky because the fault could still exist in that state. If possible, a strategy of panic, power cycle, and reset is recommended. Table 6 details examples of radiation effect mitigation relations aiding system architecture and design.

Table 6: Radiation Effect Mitigation Relation Examples

Subject	Predicate	Object
chassis hardware	bearsRole	radiation shielding
storage device	hasQuality	informational redundancy
startup	hasLog	startup health log

4. CONCLUSIONS

Provided here is an ontology for PHM in spacecraft avionics detailing taxonomic hierarchies of concepts relevant to spacecraft PHM and ontological triples relating taxonomized entities. The ontology was derived from the BFO as a TLO, but changes were made to the BFO to allow practical applications using IUPAC quantity characterizations. Instantiating a real world system using the present ontology allows for error-producing cut sets to be identified regarding system operation in a given operational environment, and error causes, preventative measures, and potential solutions, even if multitudinous, are known. Characterization of potential system errors provides engineers with substantial system analysis tools during development and operation. It is assumed and desired that the present work will undergo improvements based on PHM community feedback and use case definition. All entity definitions must be defined ontologically, and all ISO definitions must be revisited for clarity. Future work is warranted regarding the addition of roles for hardware and software, conditionally relating levels of performance with extensive and intensive physical quantities, and characterizing fault subtypes relative to types of hardware and hardware-specific particle interactions.

ACKNOWLEDGEMENT

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

REFERENCES

Baumann, R. C., (2004) "Soft errors in commercial integrated circuits." *International Journal of High Speed Electronics and Systems* 14.02: 299-309.

Bocchino, R., Canham, T., Watney, G., Reder, L.,& Levison, J. (2018) "F Prime: an open-source framework for small-scale flight software systems."

Box, George E. P. (1976), "Science and statistics" (PDF), Journal of the American Statistical Association, 71 (356): 791–799, doi:10.1080/01621459.1976.10480949.

Boyer, R.S., Moore, J.S. (1991). MJRTY—A Fast Majority Vote Algorithm. *Automated Reasoning*. Automated Reasoning Series, vol 1. Springer, Dordrecht. https://doi.org/10.1007/978-94-011-3488-0_5

Burgin, M. (2016). *Theory Of Knowledge: Structures And Processes*. World Scientific. ISBN 9789814522694.

Calspan-University of Buffalo Research Center. *Data Science and Information Fusion*. Retrieved June 19, 2022, from https://www.cubrc.org/index.php/data-science-and-information-fusion/

Calspan-University of Buffalo Research Center. *Common core ontologies for data integration*. Retrieved June 19, 2022, from https://www.cubrc.org/index.php/data-science-and-information-fusion/ontology

Choudhary, J., Balasubramanian, P., Varghese, D. M., Singh, P. D., & Maskell, D., (2019) "Generalized Majority Voter Design Method for N-Modular Redundant Systems Used in Mission- and Safety-Critical Applications" *Computers* 8, no. 1: 10. https://doi.org/10.3390/computers8010010

Cox, A.P., Nebelecky, C.K., Rudnicki, R., Tagliaferri, W.A., Crassidis, J.L. & Smith, B. (2016). The space object ontology. In *2016 19th International Conference on Information Fusion (FUSION)* (pp. 146-153). IEEE.

Ebeling, C.E., (2004). *An introduction to reliability and maintainability engineering*. Tata McGraw-Hill Education.

Elert, G. (2022). *The Standard Model*. The Physics Hypertextbook. https://physics.info/standard/#:~:text=Bosons%20are%20divided%20when%20it,Mass%20is%20energy.

Fouillat, P., Pouget, V., Lewis, D., Buchner, S., McMorrow, D., (2004) Investigation of single-event transients in fast integrated circuits with a pulsed laser. *International journal of high speed electronics and systems*. 2004 Jun;14(02):327-39.

Halvorson, M. & Thomas, L. D., (2022). "Architecture Framework Standardization for Satellite Software Generation Using MBSE and F Prime." 2022 IEEE Aerospace Conference. IEEE, 2022.

Heidergott, W. F. (2004) "System Level Single Event Upset Mitigation Strategies." *International journal of high speed electronics and systems.* 14.02 (2004): 341-352.

Ingarden, R., (1983). *Man and Value*, Munich.

International Organization for Standardization. (2020). *Information Technology – Metamodel framework for interoperability (MFI) – Part 3: Metamodel for ontology registration.* (ISO/IEC Standard no. 19763-3). Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:19763:-3:ed-3:v1:en

International Organization for Standardization. (2021). *Information Technology – Top-level ontologies (TLO) – Part 1: Requirements.* (ISO/IEC Standard no. 21838-1). Retrieved from https://www.iso.org/obp/ui/#iso:std:71954:en

International Organization for Standardization. (2021). *Information Technology – Top-level ontologies (TLO) – Part 2: Basic Formal Ontology*. (ISO/IEC Standard no. 21838-1). Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:21838:-2:ed-1:v1:en

International Organization for Standardization. (2017). *Systems and software engineering – Vocabulary*. (ISO/IEC/IEEE Standard no. 24765). Retrieved from https://www.iso.org/standard/71952.html

Jackson, R., Matentzoglu, N., Overton, J.A., Vita, R., Balhoff, J.P., Buttigieg, P.L., Carbon, S., Courtot, M., Diehl, A.D., Dooley, D.M. and Duncan, W.D., 2021. OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. Database, 2021.

Kwatny, H., (2007) Lecture Notes from Engineering Reliability Course, Department of Mechanical Engineering & Mechanics, Drexel University.

Lechner, A., (2018). CERN: Particle interactions with matter. *CERN Yellow Reports: School Proceedings, Vol. 5/2018, CERN-2018-008-SP.* p.47.

Leitner, J. (2022). Phasing in COTS EEE parts in NASA. *Community of Practice Webinar Series* . NASA Goddard Space Flight Center.

McNaught, A. D. (2019). *Compendium of chemical terminology*. (2nd Edition). Oxford: Blackwell Science.

Mishap Investigation Board. (1999). Mars Climate Orbiter Mishap Investigation Board Phase I Report November 10, 1999.

Musen, Mark. (2015) "The Protégé project: A look back and a look forward." AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.

Nöldeke, C. M., (2015). *The Space Radiation Environment*. Monsenstein Und Vannerdat.

Orilia, F., & Paoletti, M. P., (2020). *Properties*. Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/entries/properties/#DisTer

Seppälä, S., Ruttenberg, A., & Smith, B. (2017). Guidelines for writing definitions in ontologies. *Ciência da Informação* 46 (1): 73-88. PhilArchive copy v3: https://philarchive.org/archive/SEPGFWv3

Smith, B., & Brogaard, B., (2003). "Sixteen Days", *The Journal of Medicine and Philosophy*, 28 (2003), 45–78.

Smith, B., (2015). *Basic Formal Ontology 2.0: Specification and User's Guide.* University of Buffalo.

Smith, B., (2018). *Applied Ontology: Lecture 1*. *Introduction to Ontology*. University of Buffalo.

Smith, B. (2022). "The birth of ontology." Journal of Knowledge Structures and Systems 3.1.

Smith, B., Kumar, A., & Bittner, T., (2005). "Basic formal ontology for bioinformatics."

Smith, B., Malyuta, T., Rudnicki, R., Mandrick, W., Salmen, D., Morosoff, P., Duff, D.K., Schoening, J. & Parent, K., (2013). IAO-Intel: an ontology of information artifacts in the intelligence domain. *Proceedings of the Eighth International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS)*, CEUR, vol. 1097. pp. 33-40

Somani, A. K., and Vaidya, N. H., (1997) "Understanding fault tolerance and reliability." *Computer 4*: 45-50.

Thomsen, D., Kim, W., & Cutler, J. (2015) "Shields-1, A SmallSat radiation shielding technology demonstration."

Wasson, C. S., (2016*) System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*. Wiley Blackwell.

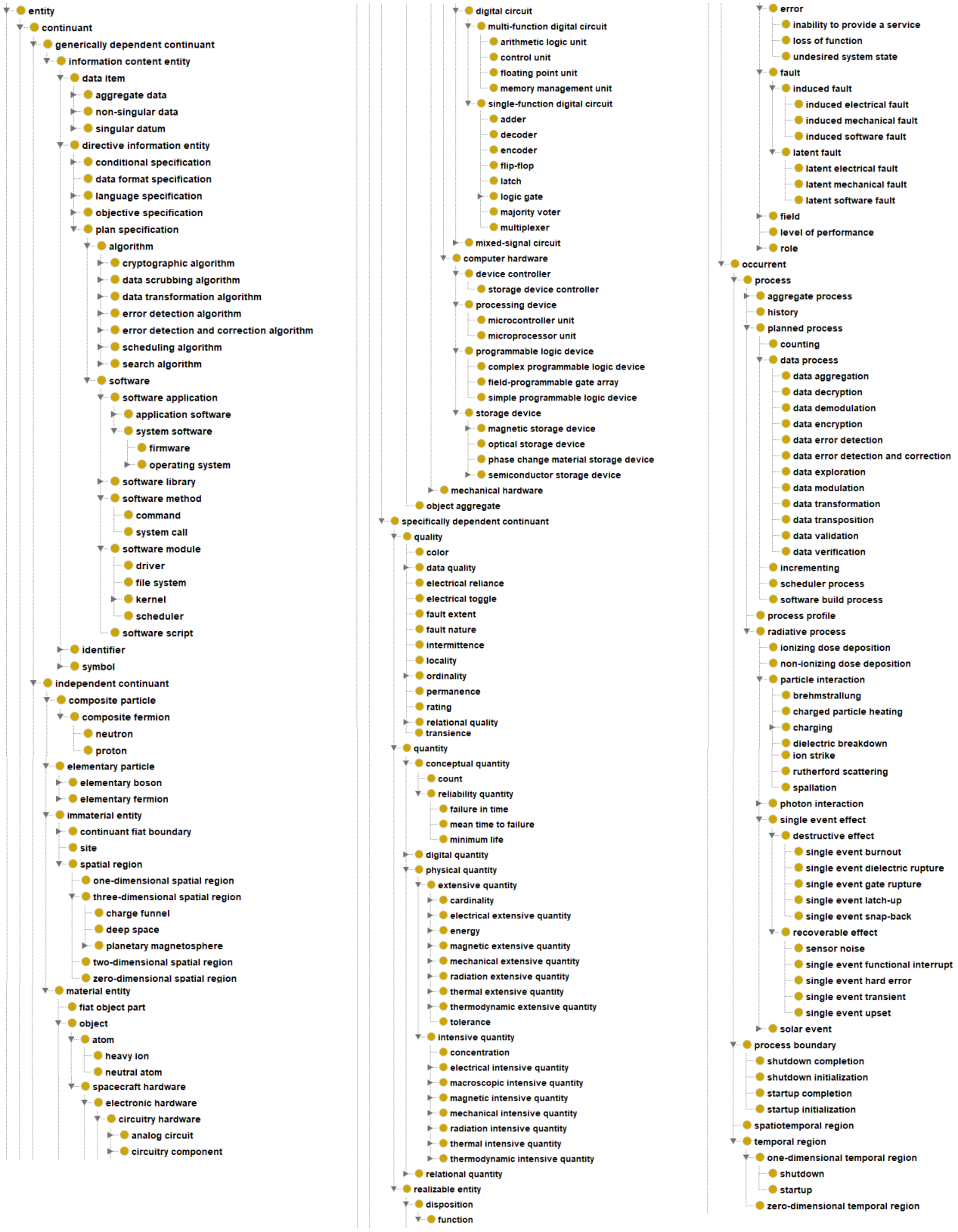
BIOGRAPHIES

Michael Cullen Halvorson received a B.S. in Aerospace Engineering and a B.S. in Mechanical Engineering from Auburn University in 2017. He then received a M.S. in Mechanical Engineering from Auburn University in 2020 and is now a doctoral student in Aerospace Systems Engineering at the University of Alabama in Huntsville. Michael has been Chief Engineer for three satellite programs and has been Chief Engineer and acting Lead Systems Engineer for the Alabama Burst Energetics eXplorer since January 2021. Michael has been a NASA Research Fellow with the Alabama Space Grant Consortium since 2018 and leads collaborative research projects on spacecraft thermal analysis, Model-Based Project Management, space engineering ontologies, and climate change mitigation strategy scalability.

Noah Moyers received a B.S. in Software Engineering from Auburn University in May 2022. He is currently a graduate student in the Computer Science and Software Engineering department at Auburn University. Noah has been a member of the Flight Software team for the Alabama Burst Energetics eXplorer since January 2022. His research interests include computer architectures, embedded systems, and operating systems.

L. Dale Thomas currently serves as a Professor and Eminent Scholar of Systems Engineering in the Department of Industrial and Systems Engineering and Engineering Management at the University of Alabama in Huntsville (UAH). He teaches system engineering students in the art and science of systems architecture and design, systems integration, test, and verification, and systems management. Dale also serves as director of the Alabama Space Grant Consortium and as deputy director of the UAH Propulsion Research Center. Prior to his retirement from NASA in July 2015, Dale served as the Associate Center Director (Technical) for the NASA Marshall Space Flight Center (MSFC) in Huntsville, Alabama, providing technical leadership for all MSFC spaceflight projects. He had previously served as the NASA Constellation Program Manager, leading the Constellation Program Office at Johnson Space Center in Houston, Texas, leading a nationwide team including all NASA field centers and five prime contractors.

APPENDIX


entity
continuant
generically dependent continuant
information content entity
data item
aggregate data
non-singular data
singular datum
directive information entity
conditional specification
data format specification
language specification
objective specification
plan specification
algorithm
cryptographic algorithm
data scrubbing algorithm
data transformation algorithm
error detection algorithm
error detection and correction algorithm
scheduling algorithm
search algorithm
software
software application
application software
system software
firmware
operating system
software library
software method
command
system call
software module
driver
file system
kernel
scheduler
software script
identifier
symbol
independent continuant
composite particle
composite fermion
neutron
proton
elementary particle
elementary boson
elementary fermion
immaterial entity
continuant fiat boundary
site
spatial region
one-dimensional spatial region
three-dimensional spatial region
charge funnel
deep space
planetary magnetosphere
two-dimensional spatial region
zero-dimensional spatial region
material entity
fiat object part
object
atom
heavy ion
neutral atom
spacecraft hardware
electronic hardware
circuitry hardware
analog circuit
circuitry component
digital circuit
multi-function digital circuit
arithmetic logic unit
control unit
floating point unit
memory management unit
single-function digital circuit
adder
decoder
encoder
flip-flop
latch
logic gate
majority voter
multiplexer
mixed-signal circuit
computer hardware
device controller
storage device controller
processing device
microcontroller unit
microprocessor unit
programmable logic device
complex programmable logic device
field-programmable gate array
simple programmable logic device
storage device
magnetic storage device
optical storage device
phase change material storage device
semiconductor storage device
mechanical hardware
object aggregate
specifically dependent continuant
quality
color
data quality
electrical reliance
electrical toggle
fault extent
fault nature
intermittence
locality
ordinality
permanence
rating
relational quality
transience
quantity
conceptual quantity
count
reliability quantity
failure in time
mean time to failure
minimum life
digital quantity
physical quantity
extensive quantity
cardinality
electrical extensive quantity
energy
magnetic extensive quantity
mechanical extensive quantity
radiation extensive quantity
thermal extensive quantity
thermodynamic extensive quantity
tolerance
intensive quantity
concentration
electrical intensive quantity
macroscopic intensive quantity
magnetic intensive quantity
mechanical intensive quantity
radiation intensive quantity
thermal intensive quantity
thermodynamic intensive quantity
relational quantity
realizable entity
disposition
function
error
inability to provide a service
loss of function
undesired system state
fault
induced fault
induced electrical fault
induced mechanical fault
induced software fault
latent fault
latent electrical fault
latent mechanical fault
latent software fault
field
level of performance
role
occurrent
process
aggregate process
history
planned process
counting
data process
data aggregation
data decryption
data demodulation
data encryption
data error detection
data error detection and correction
data exploration
data modulation
data transformation
data transposition
data validation
data verification
incrementing
scheduler process
software build process
process profile
radiative process
ionizing dose deposition
non-ionizing dose deposition
particle interaction
brehmstrallung
charged particle heating
charging
dielectric breakdown
ion strike
rutherford scattering
spallation
photon interaction
single event effect
destructive effect
single event burnout
single event dielectric rupture
single event gate rupture
single event latch-up
single event snap-back
recoverable effect
sensor noise
single event functional interrupt
single event hard error
single event transient
single event upset
solar event
process boundary
shutdown completion
shutdown initialization
startup completion
startup initialization
spatiotemporal region
temporal region
one-dimensional temporal region
shutdown
startup
zero-dimensional temporal region