A Survey of Ontology-Enabled Processes for Dependable Robot Autonomy

Finley Holt

2025-12-25

Table of contents

1 A Survey of Ontology-Enabled Processes for Dependable Robot Autonomy

Source: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2024.1377897/full
Authors: Esther Aguado, Virgilio Gomez, Miguel Hernando, Claudio Rossi, Ricardo Sanz
Affiliation: Autonomous Systems Laboratory & Centre for Automation and Robotics, Universidad Politécnica de Madrid-CSIC **Year**: 2024 **Journal**: Frontiers in Robotics and AI **DOI**: 10.3389/frobt.2024.1377897 **Article Type**: Systematic Review

---

1.1 Overview

This systematic review examines how knowledge representation and reasoning (KR&R) through ontologies enhance autonomous robot dependability in complex, open-ended environments. The survey addresses unresolved issues in robot autonomy concerning dependability and trust by analyzing how declarative knowledge enables robots to adapt structure and re-plan actions during mission execution, even when facing unexpected events.

Research Motivation: Knowledge-driven approaches offer potential improvements in flexibility, explainability, and efficacy for autonomous robotic systems. The survey focuses on ontologies that facilitate action selection and arrangement, and those that support contingency management and adaptation.

Primary Objective: Examine how ontologies are leveraged at runtime to ensure successful mission completion while aligning with user and owner expectations, thereby increasing system dependability.

1.2 Decision Impact for Flyby-F11

This section synthesizes survey findings into actionable decisions for the Flyby-F11 ontology-based autonomy architecture.

1.2.1 ADOPT - High Confidence

These approaches are validated by survey findings and should be adopted with confidence:

1. SUMO as Upper Ontology - **Evidence**: Largest open-source ontology with expressive formal definitions; foundational basis for IEEE CORA - **Decision**: Use SUMO as upper-level ontology foundation - **Rationale**: Extensive domain coverage, mature tooling, IEEE standards alignment - **Implementation**: Import SUMO core concepts, extend with UAV-specific domain ontology

2. OWL 2 DL Language - **Evidence**: Open-world assumption better suited for incomplete information in real-world robotics; description logic provides balance of expressiveness and computational tractability - **Decision**: Use OWL 2 DL profile for all ontology development - **Rationale**: Decidable reasoning guarantees, open-world assumption handles uncertainty, W3C standard ensures tooling support - **Implementation**: OWL 2 DL with explicit class axioms, property restrictions, and cardinality constraints

3. IEEE Autonomous Robotics Standard - **Evidence**: IEEE ORA working group standardizes concepts for autonomous systems; extends CORA with aerial robot support - **Decision**: Adopt IEEE 1872.2-2021 (Autonomous Robotics) as domain ontology foundation - **Rationale**: Unambiguous hardware/software component identification, validated for UAV domain, community-wide acceptance - **Implementation**: Import IEEE AUR ontology, extend with mission-specific concepts

4. HermiT and Pellet Reasoners - **Evidence**: Survey identifies HermiT and Pellet as primary OWL reasoners with production use in robotics - **Decision**: Use HermiT as primary reasoner with Pellet as fallback - **Rationale**: HermiT has hypertableau calculus for OWL 2 reasoning; Pellet provides incremental reasoning support - **Implementation**: HermiT for design-time classification, Pellet for runtime incremental updates

5. Cached Inference Strategy - **Evidence**: Survey identifies computational overhead as key challenge; recommends caching for time-critical decisions - **Decision**: Implement design-time inference caching with runtime query optimization - **Rationale**: Edge compute constraints (Jetson Orin NX 16GB) require minimizing reasoning latency - **Implementation**: Pre-classify ontology at mission load, cache common query results, use incremental reasoning for updates

1.2.2 CONSIDER - Needs Validation

These approaches show promise but require validation for edge-based UAV deployment:

1. Runtime Reasoning Performance - **Survey Finding**: Computational overhead identified as challenge; limited quantitative performance data - **Concern**: Jetson Orin NX has 50 TOPS GPU but limited CPU for symbolic reasoning - **Validation Needed**: Benchmark HermiT/Pellet reasoning latency on Jetson for UAV ontology size - **Mitigation Strategy**: Implement timeout-based reasoning with fallback to cached knowledge if inference exceeds threshold - **Test Plan**: Profile reasoning time vs ontology size, query complexity, ABox assertion count

2. OWLAPI vs OWLREADY for ROS 2 Integration - **Survey Finding**: Both OWLAPI (Java) and OWLREADY (Python) identified as integration options - **Concern**: Java overhead vs Python performance trade-offs; ROS 2 primarily Python/C++ - **Validation Needed**: Compare memory footprint, query performance, ROS 2 integration complexity - **Decision Criteria**: If OWLREADY meets performance requirements, prefer for Python/ROS 2 native integration; otherwise use OWLAPI with JNI bridge - **Test Plan**: Implement prototype perception-to-ontology pipeline with both libraries, measure latency and resource usage

3. Dynamic Knowledge Base Updates from Perception - **Survey Finding**: Runtime KB updates enable situation assessment, but update frequency impacts reasoning performance - **Concern**: High-frequency sensor data (T265 odometry at 200Hz, D455 depth at 30Hz) may overwhelm KB update cycle - **Validation Needed**: Determine optimal update frequency for different percept types - **Mitigation Strategy**: Implement percept filtering (only significant state changes trigger KB updates), batch updates for efficiency - **Test Plan**: Measure reasoning time degradation as ABox size grows during mission execution

4. Separation of Static and Dynamic Knowledge - **Survey Finding**: Recommended for performance optimization through partitioning - **Concern**: Unclear boundary between static (mission-independent) and dynamic (runtime-updated) knowledge - **Validation Needed**: Empirically determine which concepts belong in TBox vs ABox for UAV missions - **Implementation Strategy**: Static TBox includes mission plans, UAV capabilities, safety constraints; dynamic ABox includes current state, percepts, environmental conditions - **Test Plan**: Measure query performance with different TBox/ABox partitioning strategies

5. Incremental Reasoning - **Survey Finding**: Incremental reasoning recommended for efficiency; Pellet supports incremental classification - **Concern**: Limited documentation on incremental reasoning effectiveness for robotic ontologies - **Validation Needed**: Assess incremental reasoning benefit vs full re-classification for typical KB update patterns - **Decision Criteria**: Adopt if incremental reasoning provides >50% latency reduction for runtime updates - **Test Plan**: Compare Pellet incremental vs HermiT full classification for simulated mission percept updates

1.2.3 AVOID - Evidence Against

These approaches have limitations identified by the survey and should be avoided:

1. Prolog with Closed-World Assumption - **Survey Finding**: Closed-world assumption only considers explicitly true statements as true; absence of information implies falsity - **Evidence Against**: Open-world assumption in OWL better handles incomplete information in real-world robotics (lines 107-119) - **Decision**: Do NOT use Prolog as primary knowledge representation language - **Exception**: Consider Prolog for bounded, complete domains (e.g., finite state machines) where closed-world is appropriate - **Alternative**: Use OWL with open-world assumption; if Prolog integration needed, use SWI-Prolog with OWL import via rosprolog

2. CORA Alone (Without Extensions) - **Survey Finding**: CORA uses OWL-Lite subset with limited expressiveness; only necessary conditions, not sufficient conditions (lines 223-229) - **Evidence Against**: "Limited to simple classification queries" and "does not fully leverage SUMO's first-order and higher-order logic formulas" - **Decision**: Do NOT use CORA directly; instead use IEEE 1872.2-2021 (AUR) which extends CORA for autonomous systems - **Rationale**: CORA too generic for UAV autonomy; requires IEEE AUR extension for aerial robotics

3. First-Order Logic Reasoners (e.g., SUMO-KIF) - **Survey Finding**: SUMO originally in SUO-KIF (higher-order logic), but robotics applications use OWL translations - **Evidence Against**: Higher-order logic is undecidable; no tractable reasoning guarantees - **Decision**: Use SUMO-OWL translation, NOT native SUMO-KIF - **Rationale**: Description logic (OWL 2 DL) provides decidability and tractable reasoning; essential for real-time autonomy

4. Cloud-Based Reasoning - **Survey Context**: While not explicitly discussed, some approaches offload reasoning to cloud infrastructure - **Evidence Against**: Flyby-F11 mission profile includes communications-denied operations (see SYSTEM_CONSTRAINTS.md) - **Decision**: All reasoning MUST execute locally on Jetson Orin NX; no cloud dependency - **Rationale**: Edge-first autonomy is core architectural constraint; mission success cannot depend on network availability

5. Design-Time Only Ontology Use - **Survey Finding**: Many systems use ontologies primarily at design time, not runtime (identified as research gap, line 581) - **Evidence Against**: Survey emphasizes runtime use for situation assessment, action selection, and adaptation as key to dependability - **Decision**: Ontology MUST be used at runtime for decision-making, not just design-time system specification - **Rationale**: Runtime reasoning enables adaptation to unexpected events, fault tolerance, and communications-denied autonomy

1.2.4 INVESTIGATE - Open Questions

These areas require further research and experimentation:

1. Incremental Reasoning Scalability - **Open Question**: How does incremental reasoning performance scale with mission duration as ABox grows? - **Survey Gap**: Limited quantitative data on long-duration autonomous missions - **Investigation Plan**: - Simulate 30-minute mission with realistic percept update rates - Measure reasoning latency vs ABox size over time - Identify if/when full KB reset is needed - **Success Criteria**: Reasoning latency remains <100ms for mission duration up to 1 hour

2. Probabilistic Ontology Extensions - **Open Question**: Can probabilistic annotations improve situation assessment under sensor uncertainty? - **Survey Recommendation**: Future research direction (lines 708-710) - **Investigation Plan**: - Evaluate PR-OWL (Probabilistic OWL) for sensor fusion uncertainty representation - Compare with existing probabilistic filters (EKF, particle filter) - Assess computational overhead of probabilistic reasoning - **Success Criteria**: Probabilistic ontology provides interpretable uncertainty quantification without exceeding latency budget

3. Ontology-Behavior Tree Integration Pattern - **Open Question**: Optimal design pattern for ontology-driven BT node selection and parameterization - **Survey Finding**: Limited discussion of BT integration; most work focuses on classical planning - **Investigation Plan**: - Design BT condition nodes that query ontology (e.g., "IsObstaclePresent?") - Implement ontology-driven BT structure adaptation (e.g., switch from "Survey" BT to "Avoid" BT based on percepts) - Measure overhead of ontology queries in BT tick cycle - **Success Criteria**: Ontology queries complete within BT tick budget (target: 10Hz BT frequency)

4. Learning-Based Ontology Refinement - **Open Question**: Can mission execution experience refine ontology assertions over time? - **Survey Recommendation**: Future research direction for hybrid symbolic-subsymbolic approaches (line 698-701) - **Investigation Plan**: - Identify concepts amenable to learning (e.g., obstacle detection confidence thresholds) - Evaluate ontology learning techniques (e.g., association rule mining from mission logs) - Design validation approach to prevent incorrect learned assertions - **Success Criteria**: Learned ontology refinements improve mission success rate in repeated scenarios

5. Multi-UAV Shared Ontology Consistency - **Open Question**: How to maintain ontology consistency across multiple UAVs with intermittent communication? - **Survey Finding**: Limited work on distributed reasoning and multi-robot ontology synchronization (line 703) - **Investigation Plan**: - Design ontology synchronization protocol for opportunistic communication - Evaluate conflict resolution strategies (e.g., timestamp-based, logical monotonicity) - Test with simulated multi-UAV scenarios - **Success Criteria**: Ontology convergence within 5 minutes of communication restoration; no mission failures due to inconsistent knowledge - **Note**: Defer to future work; focus on single-UAV autonomy first

---

1.3 Ontology Stack Decision Matrix

This matrix compares candidate upper ontologies against Flyby-F11 requirements:

Criteria	SUMO	DOLCE/DUL	BFO	Weight	Winner
Robotics Domain Coverage	Extensive (via CORA)	Moderate (via SOMA)	Limited (biomedical focus)	HIGH	SUMO
IEEE Standards Alignment	Strong (CORA basis)	None	None	HIGH	SUMO
Expressiveness	Higher-order logic (use OWL translation)	Description logic	Description logic	MEDIUM	SUMO
UAV-Specific Concepts	Available via IEEE AUR	Requires custom extension	Requires custom extension	HIGH	SUMO
Open-Source Availability	Fully open	Fully open	Fully open	MEDIUM	TIE

Criteria	SUMO	DOLCE/DUL	BFO	Weight	Winner
Tool Support (OWL)	Good (SUMO-OWL translation)	Excellent (native OWL)	Good	MEDIUM	DOLCE
Runtime Use Precedent	Extensive in robotics	Extensive in robotics (SOMA/NEEMs)	Limited in robotics	HIGH	TIE
Community Adoption	High in robotics	Moderate in robotics	Low in robotics	MEDIUM	SUMO
Learning Curve	Steep (large ontology)	Moderate (smaller, cleaner)	Moderate	LOW	DOLCE
Ontology Size	Very large (~25K axioms)	Moderate (~2K axioms)	Small (~200 classes)	MEDIUM	DOLCE

Decision Rationale: - **SUMO** wins on robotics-critical criteria: IEEE standards alignment, UAV-specific concepts via IEEE AUR, extensive robotics domain coverage - **DOLCE/DUL** advantages (smaller size, cleaner OWL, easier learning) are outweighed by lack of IEEE standards integration - **BFO** biomedical focus makes it poor fit for UAV autonomy domain

Final Decision: Adopt **SUMO** as upper ontology foundation, imported via SUMO-OWL translation to maintain OWL 2 DL decidability.

Implementation Notes: - Use SUMO subset import (not full ontology) to reduce size: focus on Process, Device, Agent, Event, Attribute concepts - Extend with IEEE 1872.2-2021 (Autonomous Robotics) for UAV-specific domain concepts - Consider DOLCE design patterns (from DUL) for application ontology architecture if SUMO proves unwieldy

---

1.4 Tool Selection Comparison

1.4.1 Reasoner Comparison: HermiT vs Pellet vs FaCT++

Criteria	HermiT 1.4.5	Pellet 2.3.6	FaCT++ 1.6.5	Winner
OWL 2 DL Support	Full OWL 2 DL	Full OWL 2 DL	OWL 2 subset	HermiT/Pellet
Reasoning Algorithm	Hypertableau	Tableau	Tableau	HermiT (newer)
Incremental Reasoning	No	Yes (incremental classification)	No	Pellet
Performance (Classification)	Fast for complex axioms	Moderate	Very fast for simple ontologies	FaCT++
Performance (Query Answering)	Good	Good	Limited support	HermiT/Pellet

Criteria	**HermiT 1.4.5**	**Pellet 2.3.6**	**FaCT++ 1.6.5**	**Winner**
Consistency Checking	Excellent	Excellent	Good	HermiT/Pellet
Explanation Generation	Yes (proof trees)	Yes (proof graphs)	No	HermiT/Pellet
Java API	Native	Native	JNI wrapper	HermiT/Pellet
Active Maintenance	Moderate (2020 last release)	Low (2017 last release)	Moderate (2015 last release)	HermiT
ROS Integration Precedent	Common in ROS projects	Common in ROS projects	Limited	HermiT/Pellet
Memory Footprint	Moderate	High (caches for incremental)	Low	FaCT++
License	LGPL 3.0	AGPL 3.0	LGPL 2.1	HermiT (permissive)

Decision Strategy: - **Primary Reasoner**: HermiT 1.4.5 - Rationale: Full OWL 2 DL support, hypertableau algorithm efficient for complex axioms, LGPL license compatible with project, active ROS community use - Use Case: Design-time ontology classification, runtime consistency checking, query answering

- **Secondary Reasoner**: Pellet 2.3.6
  - Rationale: Incremental reasoning critical for runtime KB updates; worth AGPL license consideration
  - Use Case: Runtime ABox updates (percept insertion, state changes) where incremental classification avoids full re-classification overhead
  - License Note: AGPL requires careful integration (isolate in separate service, communicate via ROS 2 topics to maintain license separation)
- **Not Selected**: FaCT++
  - Rationale: While fastest for simple ontologies, limited OWL 2 support and query capabilities disqualify it for complex UAV autonomy reasoning
  - Exception: Consider if performance profiling reveals HermiT/Pellet too slow and ontology can be simplified to FaCT++ subset

Fallback Strategy: If reasoner latency exceeds budget (>100ms for runtime queries), implement timeout with fallback to cached knowledge or heuristic decision-making.

1.4.2 API Comparison: OWLAPI vs OWLREADY

Criteria	**OWLAPI 5.x (Java)**	**OWLREADY 2.x (Python)**	**Winner**
Language	Java	Python	Python (ROS 2 native)
ROS 2 Integration	Requires JNI bridge or rclpy subprocess	Native (import in Python nodes)	OWLREADY

Criteria	OWLAPI 5.x (Java)	OWLREADY 2.x (Python)	Winner
OWL 2 DL Coverage	Complete	Complete	TIE
Reasoner Integration	HermiT, Pellet, FaCT++ (native)	HermiT, Pellet via JPype or subprocess	OWLAPI
Performance (Loading)	Fast	Moderate (Python overhead)	OWLAPI
Performance (Querying)	Fast	Moderate	OWLAPI
Memory Footprint	High (JVM heap)	Moderate (Python objects)	OWLREADY
API Ergonomics	Verbose (Java beans)	Pythonic (attribute access)	OWLREADY
Triple Store Backend	In-memory or Jena TDB	Optimized quadstore (SQLite)	OWLREADY
SPARQL Support	Via Jena ARQ	Via RDFlib or OWLREADY queries	TIE
Dynamic Class Creation	Difficult (reflection)	Easy (Python metaprogramming)	OWLREADY
Active Maintenance	High (2023 releases)	High (2023 releases)	TIE
Documentation Quality	Excellent	Good	OWLAPI
License	Apache 2.0 or LGPL	LGPL 3.0	TIE (both permissive)
Biomedical Use Precedent	Extensive	Extensive	TIE
Robotics Use Precedent	Moderate (KnowRob uses Prolog/Java)	Emerging	OWLAPI

Decision Strategy:

Phase 1 - Prototype (Immediate): - **Use OWLREADY 2.x** - Rationale: Rapid prototyping with Python/ROS 2 native integration; Pythonic API reduces development time - Implementation: Import OWLREADY in ROS 2 Python nodes, load SUMO + IEEE AUR ontologies, implement percept-to-ontology mapping - Validation: Measure query latency, memory footprint on Jetson Orin NX

Phase 2 - Performance Validation (After Prototype): - **Benchmark**: Profile OWLREADY performance with realistic UAV ontology and query workload - **Decision Criteria**: - If OWLREADY meets latency budget ($<$100ms for 95th percentile queries) $\rightarrow$ **Adopt OWLREADY** - If OWLREADY exceeds budget $\rightarrow$ **Migrate to OWLAPI** with JNI bridge or rclpy subprocess architecture

Phase 3 - Production (Conditional): - **If OWLREADY Adequate**: - Stick with OWLREADY for simplicity and ROS 2 native integration - Optimize Python code (use Cython for critical paths if needed) - Leverage OWLREADY quadstore persistence for mission state recovery

- **If OWLAPI Required**:
  - Implement dual-process architecture: Java reasoning service + ROS 2 bridge
  - Use rclpy to spawn Java subprocess, communicate via ROS 2 topics/services
  - Accept JVM memory overhead (Jetson Orin NX has 16GB RAM, sufficient for JVM + ROS 2)

Recommended Initial Path: Start with **OWLREADY** for rapid development; migrate to OWLAPI only if performance profiling proves necessary.

Hybrid Approach (If Needed): - Use OWLREADY for design-time ontology development (Protégé integration, OWL manipulation) - Use OWLAPI for runtime reasoning (better performance, mature reasoner integration) - Export ontology from OWLREADY, load in OWLAPI-based reasoning service

---

1.5 Dependability in Autonomous Robotics

1.5.1 Dependability Definition

Dependability is defined as "the ability to provide the intended services of the system with a certain level of assurance," encompassing: - **Availability**: System readiness for correct service - **Reliability**: Continuity of correct service - **Safety**: Absence of catastrophic consequences - **Security**: Protection against unauthorized access and improper alterations

The authors cite Avižienis et al. (2004) who provide a comprehensive taxonomy of dependable computing and associated faults as a foundational framework.

1.5.2 Dependability vs. Reliability

Dependability goes beyond reliability by considering: - Fault tolerance mechanisms - Error recovery procedures - Maintaining service levels under degraded conditions

Model-Based Approaches: The survey emphasizes using explicit models (ontologies) to guide action selection at runtime, enabling robots to evaluate goals and develop alternative plans when faults occur. This approach differs from traditional reliability analysis tools (fault trees, FMEA, safety cases) by enabling dynamic, runtime decision-making.

1.6 Knowledge Representation and Reasoning Fundamentals

1.6.1 Core Concepts

Knowledge Representation and Reasoning (KR&R): A subarea of AI concerned with analyzing, designing, and implementing ways of representing information on computers so computational agents can derive implied information (Shapiro, 2003).

Reasoning: The process of extracting new information from implications of existing knowledge.

1.6.2 Benefits for Robotics

1. **Adaptability**: Programmers cannot fully predict world states in advance; KR&R provides background to reason about runtime situations and act accordingly

2. **Explainability**: Knowledge can be queried so humans or other agents understand why a robot acts in a certain way
3. **Reusability**: Knowledge bases can be stored in broadly applicable modular chunks and shared among different agents, applications, or tasks

1.6.3 Machine-Understandable Knowledge

For robot use, knowledge bases must be machine-understandable, allowing robots to: - Read knowledge content - Reason about encoded information - Update knowledge dynamically

Ontologies: Define the conceptualizations that robots require to support autonomous decision-making, written in specific computer languages (Prolog, OWL).

1.7 Knowledge Representation Languages

1.7.1 Prolog (PROgramming in LOGic)

Characteristics: - Most widely used declarative programming language - Based on decidable fragments of first-order logic (FOL) - **Closed-World Assumption**: Only predicates explicitly defined in the KB are true; no way to declare sentences as false - Notation differs from standard FOL - Used extensively in legal, medical, financial, and other domains

Integration with Robotics: - APIs: GNU-PROLOG, SWI-PROLOG - ROS Integration: `rosprolog` package interfaces between SWI-Prolog and ROS - Supports reasoning about knowledge represented in OWL

1.7.2 OWL (Ontology Web Language)

Characteristics: - Family of languages for knowledge representation (not a programming language like Prolog) - Designed for machine interpretability rather than human presentation - **Open-World Assumption**: A statement can be true whether it is known or not; only explicitly false predicates are false - Formal basis: Description Logic (DL) - decidable fragments of FOL with compromise between expressiveness and scalability - W3C standard, cornerstone of the semantic web

Formats: - Extensible Markup Language (XML) - Resource Description Framework (RDF) - RDF Schema (RDF-S)

Tools and APIs: - **Editors**: Protégé (Musen, 2015) for user-friendly environment - **Java APIs**: Jena Ontology API, OWLAPI - **Python API**: OWLREADY (Lamy, 2017) - **Reasoners**: FaCT++, Pellet, HermiT

1.7.3 Closed-World vs. Open-World Assumption

Prolog (Closed-World): - Only explicitly true statements are considered true - Absence of information implies falsity - Better for complete, bounded domains

OWL (Open-World): - Statements can be true even if not known - Absence of information does not imply falsity - Better suited for incomplete, evolving knowledge in robotic applications

Recommendation: The survey implicitly favors OWL for robotics due to its open-world assumption, which better handles uncertainty and incomplete information in real-world environments.

1.8 Ontology Abstraction Hierarchy

Following Guarino's hierarchy (Guarino, 1998), ontological systems are classified into three levels:

1.8.1 Upper-Level (Foundational) Ontologies

Conceptualize general terms such as: - Object, property, event, state - Relations: parthood, constitution, participation

Key Upper-Level Ontologies:

1.8.1.1 1. SUMO (Suggested Upper Merged Ontology)

- **Source**: Niles & Pease (2001), Pease (2011)
- **Description**: Largest open-source ontology with expressive formal definitions
- **Language**: SUO-KIF (Standard Upper Ontology Knowledge Interchange Format) - uses higher-order logic
- **Coverage**: Domain ontologies for medicine, economics, engineering, and many other topics
- **Advantages**: Rich formal definitions, extensive domain coverage
- **URL**: https://www.ontologyportal.org

1.8.1.2 2. DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering)

- **Source**: Gangemi et al. (2002)
- **Description**: "Ontology of universals" with classes but not relations
- **Purpose**: Capture ontological categories underlying natural language and human common sense
- **Basic Categories**: Abstract quality, abstract region, agentive physical object, amount of matter, temporal quality
- **Original Language**: First-order logic, later implemented in OWL
- **Extensions**: Most extensions use OWL

1.8.1.3 3. DUL (DOLCE + DnS Ultralite)

- **URL**: http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite
- **Purpose**: Simplifies DOLCE with simpler class/relation names and constructs
- **Key Innovation**: Architecture based on design patterns
- **Advantages**: More accessible than DOLCE while maintaining conceptual rigor

1.8.1.4 4. BFO (Basic Formal Ontology)

- **Source**: Arp et al. (2015)
- **Focus**: Continuant entities (3D reality) and occurrent entities (including time dimension)
- **Application**: Widely used in biomedical informatics

1.8.1.5 5. BWW (Bunge-Wand-Weber Ontology)

- **Source**: Bunge (1977), Wand & Weber (1993)
- **Basis**: Bunge's philosophical system

- **Application**: Conceptual modeling (Lukyanenko et al., 2021)

1.8.1.6 6. Cyc Ontology

- **Source**: Lenat (1995)
- **Purpose**: Long-term AI project to represent implicit knowledge and perform human-like reasoning
- **Scope**: Comprehensive common-sense knowledge base

1.8.2 Domain Ontologies

Provide formal representation of specific fields: - Define contractual agreements on term meanings within a discipline - Specify highly reusable vocabulary of an area - Include concepts, objects, activities, and governing theories

Examples: Medical ontologies, engineering ontologies, economic ontologies

1.8.3 Application Ontologies

Contain definitions required to model knowledge for particular applications: - Information about a robot in a specific environment - Descriptions of particular tasks - Environment or task knowledge can be subdomain ontologies depending on reusability

Note: The progression from upper-level to application ontologies is a continuous spectrum of concept subclassing with somewhat arbitrary divisions.

1.9 Robotic Domain Ontologies

1.9.1 IEEE Standard Ontologies

The IEEE Robotics and Automation Society (RAS) created the Ontologies for Robotics and Automation (ORA) working group to develop standards for knowledge representation in robotics.

1.9.1.1 CORA (Core Ontology for Robotics and Automation)

- **Standard**: Prestes et al. (2013)
- **Purpose**: Specify most general concepts, relations, and axioms for robotics and automation domain
- **Basis**: Built on SUMO
- **Core Entities**:
    1. Robot Part
    2. Robot
    3. Complex Robot
    4. Robotic System

Extensions (IEEE Standard 1872-2015):

1. **CORAX**: Bridge between SUMO and CORA
    - High-level concepts for design, interaction, and environment
    - Fills gaps not explicitly defined in SUMO

2. **RPARTS**: Robot parts and roles
    - Specific kinds of robot parts (grippers, sensors, actuators)
    - Performance roles for components
3. **POS**: Spatial knowledge
    - Position and orientation concepts
    - Points, regions, coordinate systems

Limitations: - Intended for broad community, so definitions use only necessary conditions (not sufficient conditions) - Limited expressiveness (mostly OWL-Lite) - Limited to simple classification queries - Does not fully leverage SUMO's first-order and higher-order logic formulas - Concepts must be specialized for specific subdomains

1.9.1.2 Robot Task Representation (IEEE ORA Subgroup)

- **Level**: Middle-level ontology
- **Focus**: Comprehensive task decomposition from goal to subgoals
- **Coverage**:
    - Task definitions and properties
    - Performance capabilities required
    - Catalog of tasks demanded by community (especially industrial processes)
- **Source**: Balakirsky et al. (2017)

1.9.1.3 IEEE Standard for Autonomous Robotics (AUR)

- **Standard**: IEEE SA (2022)
- **Purpose**: Extends CORA for autonomous robots domain
- **Coverage**: Aerial, ground, surface, underwater, and space robots
- **Focus**: Unambiguous identification of basic hardware and software components for autonomy

Intended Uses: 1. Describe design patterns of Autonomous Robotics (AuR) systems 2. Represent AuR system architectures in unified way 3. Guideline for building autonomous systems with robots in various environments

Additional Subgroups: - Autonomous Robots Ontology (active) - Industrial Ontology (inactive) - Medical Robot Ontology (inactive)

1.9.2 Non-IEEE Domain Ontologies

1.9.2.1 OASys (Ontology for Autonomous Systems)

- **Source**: Bermejo-Alonso et al. (2010)
- **Purpose**: Capture and exploit concepts to support description of any autonomous system
- **Emphasis**: Associated engineering processes
- **Levels**:
    1. Systems in general
    2. Autonomous systems in particular
- **Connections**: Links architecture, components, goals, and functions with engineering processes

1.9.2.2 SOMA (Socio-physical Model of Activities)

- **Source**: Beßler et al. (2021)

- **Purpose**: Represent physical and social context of everyday activities
- **Basis**: Built on DUL (DOLCE + DnS Ultralite)
- **Extensions**:
  - Event types: action, process, state
  - Objects participating in activities
  - Execution concepts

Key Innovation: Designed for runtime use with NEEMs (Narratively Enabled Episodic Memories)

NEEMs: Comprehensive logs containing: - Raw sensor data - Actuator control histories - Perception events - Semantic annotations about what robot is doing and why (using SOMA terminology)

1.10 Capabilities for Robot Autonomy

1.10.1 Autonomy Definition

Etymology: Being governed by the laws of oneself rather than by rules of others (Vernon, 2014)

Robotics Definition (Beer et al., 2014): The extent to which a robot can: 1. Sense its environment 2. Plan based on that environment 3. Act on that environment 4. Reach task-specific goals (given or self-created) 5. Operate without external control

Systems-Oriented Perspective (Sanz et al., 2000): Autonomy as a relationship between system, task, and context

1.10.2 Cognition and Autonomy

Vernon (2014) argues that cognition increases the repertoire of actions and extends the time horizon for anticipating outcomes.

Six Cognitive Attributes: 1. Perception 2. Learning 3. Anticipation 4. Action 5. Adaptation 6. Autonomy

1.10.3 Functional Capabilities for Autonomous Robots

Langley et al. (2009) review cognitive architectures and establish main functional capabilities:

1. **Recognition and Categorization**: Generate abstractions from perceptions and past actions
2. **Decision Making and Choice**: Represent alternatives for selecting most prosperous action considering situation
3. **Perception and Situation Assessment**: Combine perceptual information from different sources; understand current circumstances
4. **Prediction and Monitoring**: Evaluate situation and possible effects of actions
5. **Problem Solving and Planning**: Specify desired intermediate states and actions required to reach them
6. **Reasoning and Belief Maintenance**: Use and update KB in dynamic environments
7. **Execution and Action**: Support deliberative and reactive behaviors
8. **Interaction and Communication**: Share knowledge with other agents
9. **Remembering, Reflection, and Learning**: Use meta-reasoning and past executions as experiences for future

1.10.4 Reflective Agents

Brachman (2002) argues for reflective agents that "know what they are doing": - Understand the situation - Know what they are doing, where, and why - Require coordinated teams - Need robust software and hardware infrastructure

---

1.11 Processes for Knowledge-Enabled Autonomous Robots

The survey establishes fundamental processes that autonomous robots should perform, serving as the classification criterion for the ontology review.

1.11.1 1. Perception

Process: Belief production resulting from perceptor sensing the environment

Five Entities:

1. **Sensor**: Device that detects, measures, or captures a property
   - Simple: Thermometers (single aspect)
   - Complex: Segmenting cameras (multiple features)
2. **Perceived Quality**: Feature allowing perceptor to recognize environment or robot aspects
   - Examples: Temperature, visual images, wheel rotation from encoder
3. **Perceptive Environment**: Part of environment that sensor can detect
   - Delimited by sensor resolution or resource constraints
4. **Perceptor**: Agent that perceives; link between perception and categorization
   - Takes sensor information and categorizes it
   - Usually embodies sensor (but can be decoupled for external sensors)
5. **Percept**: Inner entity (belief) resulting from perceptual process

1.11.2 2. Categorization

Process: Finding patterns and categories to model the situation in robot's knowledge

Granularity Levels: - Sensor fusion from different sensor types with uncertainty propagation - Classification of entities (e.g., mobile obstacle for uncontrolled approaching object) - Abstract level recognition (e.g., miner robot recognizing mine ore type by geo-chemical properties)

Purpose: Combine information about objects, events, action responses, physical properties to create picture of what is happening in environment and in robot itself

Integration: Incorporates other processes such as reasoning and prediction

1.11.3 3. Decision Making

Purpose: Direct actions towards goal; select suitable alternatives when actions cannot be performed

Time Frame: Shorter than planning; focuses on successful plan completion

Decision vs. Planning: - **Decision Making**: Short-term, execution-level choices (e.g., slightly change trajectory to avoid obstacle, then return to initial path) - **Planning**: Longer time horizon, concerned with achieving goals through action sequences (e.g., examine mine, detect mineral vein, dig in that direction)

Scope: Acts upon different alternatives: - Directions and velocity - Component substitution for functional equivalence when defects occur - Execution changes while maintaining same plan

1.11.4 4. Prediction and Monitoring

Model-Based Operation: Robot uses internal model to supervise situation

Model Sources: - Given a priori - Created before stating task - Learned during operation

Model Content: Understanding of: - Robot's own characteristics - Interactions with environment - Relationships between actions and outcomes

Runtime Capabilities: - Predict effect of an action - Anticipate future events based on situation evolution - Monitor processes and compare obtained results with expected responses

Adaptation: In case of inconsistencies: - Inform external operator - Use adaptation techniques to solve errors - Example: If robot stuck, change motion direction; if still stuck, alert user

1.11.5 5. Problem Solving and Planning

Process: Define action sequence to achieve goal; establish when and why to execute each action

Granularity Levels: - **High-Level (Mission)**: Overall mission objectives - **Mid-Level (Task)**: Specific task decomposition - **Low-Level (Action)**: Individual action execution

Planning Types: - **Classical Planning**: Offline, deterministic environments - **Reactive Planning**: Online adaptation to dynamic environments - **Hierarchical Planning**: Multi-level decomposition (mission → task → action)

Requirements: - World model representation - Goal specification - Action preconditions and effects - Constraint satisfaction

1.11.6 6. Reasoning and Belief Maintenance

Purpose: Use and update knowledge base in dynamic environments

Types of Reasoning: - **Deductive**: Derive logical conclusions from premises - **Inductive**: Generalize from specific observations - **Abductive**: Infer most likely explanation for observations

Belief Maintenance: - Update beliefs based on new percepts - Resolve inconsistencies in knowledge base - Handle uncertainty and incomplete information - Truth maintenance systems

1.11.7 7. Execution and Action

Dual Behavior Support: - **Deliberative**: Goal-directed, planned actions - **Reactive**: Immediate responses to environmental stimuli

Execution Monitoring: - Track action execution progress - Detect execution failures - Trigger replanning or adaptation when needed

Action Selection: Choose appropriate action from plan considering current state

1.11.8 8. Interaction and Communication

Purpose: Share knowledge with other agents (robots, humans, systems)

Communication Types: - **Robot-Robot**: Multi-agent coordination, shared mission execution - **Human-Robot**: Task assignment, status reporting, explanation - **System-System**: Integration with external systems and infrastructure

Knowledge Sharing: - Ontology-based common vocabulary - Semantic interoperability - Shared situational awareness

1.11.9 9. Adaptation and Learning

Adaptation: Modify behavior or structure in response to environmental changes or failures

Learning Types: - **Supervised**: Learn from labeled examples - **Unsupervised**: Discover patterns in unlabeled data - **Reinforcement**: Learn from trial and error with rewards

Meta-Reasoning: Use past executions as experiences for future decision-making

Knowledge Acquisition: Expand and refine knowledge base through experience

---

1.12 Methodology

1.12.1 Systematic Review Process

The authors conducted a systematic exploration to analyze the use of ontologies in autonomous robots for facilitating complex mission development.

Research Questions: 1. How are ontologies used to support action selection and arrangement in autonomous robots? 2. How do ontologies facilitate contingency management and adaptation? 3. What are the primary application domains for ontology-enabled robot autonomy?

Selection Criteria: - Focus on ontologies for autonomous robots - Runtime use of ontologies for decision-making - Emphasis on dependability aspects (safety, reliability, adaptation, fault tolerance) - Published research with implemented or proposed ontological frameworks

1.12.2 Classification Framework

The survey classifies existing approaches based on:

1. **Application Domain**:
    - Space robotics
    - Aerial robotics (UAVs)
    - Ground robots (mobile, manipulation)
    - Service robots
    - Multi-robot systems
2. **Ontology Purpose**:
    - Action selection and planning
    - Fault detection and diagnosis
    - Adaptation and reconfiguration
    - Mission specification and execution

    - Human-robot interaction
3. **Knowledge Representation Language**:
    - Prolog-based ontologies
    - OWL-based ontologies
    - Hybrid approaches
4. **Integration Approach**:
    - ROS integration (rosprolog, OWLAPI)
    - Standalone reasoning engines
    - Embedded in control architectures

1.13 Key Findings

1.13.1 Ontology Usage Patterns

1. **Upper-Level Ontology Adoption**:
    - SUMO and DOLCE/DUL are most commonly used as foundations
    - Domain ontologies typically extend CORA or IEEE standards
    - Application ontologies highly specialized to specific platforms and missions
2. **Language Preference**:
    - OWL increasingly preferred over Prolog for new developments
    - Open-world assumption better suited for real-world robotics
    - Description logic provides good balance of expressiveness and computational tractability
3. **Runtime Integration**:
    - Ontologies used at multiple levels: design-time and runtime
    - Runtime querying for situation assessment and action selection
    - Dynamic knowledge base updates based on perception and execution results

1.13.2 Dependability Enhancement

How Ontologies Improve Dependability:

1. **Explainability**: Knowledge-based decisions can be traced and explained to operators
2. **Fault Tolerance**: Alternative actions can be selected when primary approach fails
3. **Adaptability**: Runtime reconfiguration based on changing conditions
4. **Safety**: Explicit representation of constraints and safety conditions
5. **Verification**: Formal reasoning about system properties and behaviors

1.13.3 Challenges Identified

1. **Computational Overhead**: Reasoning can be computationally expensive for real-time systems
2. **Knowledge Engineering**: Creating comprehensive ontologies requires significant expertise and effort
3. **Standardization**: Limited adoption of IEEE standards in practice; many custom ontologies
4. **Integration Complexity**: Connecting ontologies with existing robotic middleware (ROS, etc.)
5. **Validation**: Difficulty in validating completeness and correctness of ontologies

1.13.4 Research Gaps

1. **Limited Runtime Adaptation**: Most systems use ontologies primarily at design time
2. **Scalability**: Challenges scaling to complex, multi-robot scenarios
3. **Learning Integration**: Limited work on combining ontologies with machine learning
4. **Uncertainty Handling**: Inadequate representation of uncertainty in many frameworks
5. **Real-World Deployment**: Few examples of ontology-based systems in production environments

---

1.14 Relevance to Drone Autonomy

1.14.1 Direct Applicability

This survey is highly relevant to UAV autonomy development for the following reasons:

1. **ROS Integration**: Explicitly covers ontology integration with ROS through rosprolog and OWLAPI
    - Our software stack uses ROS 2
    - Integration patterns directly applicable to our architecture
2. **SUMO Foundation**: Discusses SUMO as upper-level ontology
    - Aligns with our ontology foundation approach
    - Provides validated upper-level concepts for UAV domain extension
3. **Dependability Focus**: Addresses critical dependability aspects
    - Safety (critical for UAV operations)
    - Reliability (mission success in GPS-denied environments)
    - Availability (edge-based autonomy without cloud dependency)
    - Security (NDAA compliance for defense applications)
4. **Action Selection**: Core focus on autonomous mission execution
    - Planning under uncertainty
    - Runtime decision-making
    - Alternative action selection when primary approach fails
5. **Contingency Management**: Handling unexpected situations
    - Communications-denied operations
    - Sensor failures
    - Dynamic obstacle avoidance
    - Emergency landing procedures

1.14.2 Technical Implementation Insights

For Flyby-F11 Development:

1. **Ontology Architecture**:
    - Use SUMO as upper-level ontology
    - Extend with IEEE Autonomous Robotics standard
    - Develop UAV-specific domain ontology
    - Create mission-specific application ontologies
2. **Language Selection**:
    - Prefer OWL over Prolog for open-world assumption
    - Use description logic for balance of expressiveness and performance

    - Consider OWL 2 DL profile for decidable reasoning
3. **Integration Approach**:
    - Use OWLAPI or OWLREADY for Python/ROS 2 integration
    - Implement runtime querying for situation assessment
    - Dynamic knowledge base updates from perception pipeline
4. **Reasoning Strategy**:
    - Separate design-time and runtime reasoning
    - Use cached inferences for time-critical decisions
    - Implement incremental reasoning for efficiency
5. **Dependability Mechanisms**:
    - Explicit safety constraints in ontology
    - Alternative action representation for fault tolerance
    - Mission abort criteria and safe landing locations

1.14.3 Application to Project-Drone

Development Platform Considerations:

1. **Edge Computing Constraints**:
    - Jetson Orin Nano Super (8GB RAM, 67 TOPS)
    - Optimize ontology size for memory constraints
    - Consider lightweight reasoners (e.g., HermiT with caching)
2. **Sensor Integration**:
    - T265 visual odometry → Odometry percepts
    - D455 depth camera → Obstacle percepts
    - MAVSDK telemetry → State percepts
3. **Behavior Trees Integration**:
    - Ontology-driven behavior tree node selection
    - Runtime BT structure adaptation based on ontological reasoning
    - Condition nodes query ontology for state assessment
4. **Mission-Intent Interpretation**:
    - High-level mission goals in ontology
    - Decomposition to task sequences
    - Action selection based on current capabilities and environment

1.14.4 Communications-Denied Operations

Ontology-Based Autonomy:

1. **Local Decision-Making**: All reasoning on edge compute (no cloud dependency)
2. **Situation Assessment**: Categorize environment state from local sensors
3. **Plan Adaptation**: Modify mission based on unexpected events
4. **Safe Behaviors**: Fallback actions when nominal plan fails

1.14.5 Multi-Agent Coordination (Future)

For Multi-UAV Scenarios:

1. **Shared Ontology**: Common vocabulary for inter-agent communication
2. **Role Assignment**: Dynamic task allocation based on capabilities

3. **Conflict Resolution**: Ontological reasoning about resource contention
4. **Emergent Behavior**: Coordination without centralized control

---

1.15 Future Research Directions

Based on survey findings, the authors recommend:

1. **Enhanced Runtime Adaptation**:
   - More sophisticated ontology-driven reconfiguration
   - Learning-based ontology refinement
   - Hybrid symbolic-subsymbolic approaches
2. **Scalability Improvements**:
   - Distributed reasoning for multi-robot systems
   - Incremental and approximate reasoning techniques
   - Hierarchical ontology organization
3. **Uncertainty Integration**:
   - Probabilistic ontologies
   - Fuzzy description logics
   - Bayesian networks integrated with ontologies
4. **Standardization Efforts**:
   - Broader adoption of IEEE ORA standards
   - Domain-specific extensions (UAV, underwater, space)
   - Interoperability testing and validation
5. **Real-World Validation**:
   - More field deployments
   - Long-duration autonomy demonstrations
   - Comparison with non-ontological approaches
6. **Human-Robot Collaboration**:
   - Ontologies for shared understanding
   - Natural language grounding to ontological concepts
   - Explainable AI through ontological reasoning

1.16 Implementation Recommendations

1.16.1 For UAV Ontology Development

1. **Start with Standards**: Build on SUMO and IEEE Autonomous Robotics ontology
2. **Domain Extension**: Create UAV-specific concepts (flight modes, airspace, weather)
3. **Mission Ontologies**: Develop reusable mission patterns (survey, inspection, delivery)
4. **Sensor Ontologies**: Model perception capabilities and uncertainty
5. **Safety Ontologies**: Explicit representation of safety constraints and geofences

1.16.2 Integration Strategy

1. **Design-Time Use**:
   - System architecture verification
   - Mission plan validation

    - Configuration generation
2. **Runtime Use**:
    - Situation assessment
    - Action selection
    - Fault diagnosis and recovery
    - Performance monitoring
3. **Tools and Infrastructure**:
    - Protégé for ontology development
    - HermiT or Pellet for reasoning
    - OWLAPI or OWLREADY for ROS 2 integration
    - Version control for ontology evolution

1.16.3 Performance Optimization

1. **Caching**: Pre-compute common inferences
2. **Partitioning**: Separate static and dynamic knowledge
3. **Incremental Reasoning**: Update only affected portions
4. **Approximation**: Trade precision for speed in time-critical scenarios

---

1.17 Critical Assessment

1.17.1 Strengths of Survey

1. **Comprehensive Coverage**: Thorough review of foundational and domain ontologies
2. **Clear Taxonomy**: Well-organized classification of ontology types and uses
3. **Practical Focus**: Emphasis on runtime application and dependability
4. **Implementation Guidance**: Specific recommendations for languages, tools, and approaches

1.17.2 Limitations

1. **Limited Quantitative Analysis**: Few performance comparisons between approaches
2. **Deployment Sparsity**: Relatively few real-world, long-duration deployments discussed
3. **Learning Integration**: Insufficient coverage of ontology-learning synergies
4. **Edge Computing**: Limited discussion of resource-constrained embedded platforms

1.17.3 Relevance to Our Work

Critical Resource: This survey is essential for our ontology-based UAV autonomy development:

- Provides validated architectural patterns
- Informs selection of upper-level ontology (SUMO)
- Guides language choice (OWL over Prolog)
- Recommends integration approaches (rosprolog, OWLAPI)
- Identifies pitfalls to avoid
- Suggests future research opportunities

Action Items:

1. Adopt SUMO + IEEE AUR as ontology foundation

2. Use OWL 2 DL for knowledge representation
3. Implement OWLREADY for ROS 2 integration
4. Design for runtime reasoning with cached inferences
5. Develop UAV domain ontology extending IEEE standards
6. Create mission-specific application ontologies
7. Integrate with behavior tree framework for action execution

---

1.18 References

- Aguado, E., Gomez, V., Hernando, M., Rossi, C., & Sanz, R. (2024). A survey of ontology-enabled processes for dependable robot autonomy. *Frontiers in Robotics and AI*, 11:1377897. https://doi.org/10.3389/frobt.2024.1377897
- Arp, R., Smith, B., & Spear, A. D. (2015). *Building Ontologies with Basic Formal Ontology*. MIT Press.
- Avižienis, A., Laprie, J. C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1), 11-33.
- Balakirsky, S., et al. (2017). Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from RoboCup Rescue. *Journal of Field Robotics*, 24(11-12), 943-967.
- Beer, J. M., Fisk, A. D., & Rogers, W. A. (2014). Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of Human-Robot Interaction*, 3(2), 74-99.
- Bermejo-Alonso, J., Sanz, R., Rodríguez, M., & Hernández, C. (2010). Ontology-based engineering knowledge management in ONTO-PDM. *Advanced Engineering Informatics*, 24(4), 484-500.
- Beßler, D., et al. (2021). Foundations of the socio-physical model of activities (SOMA) for autonomous robotic agents. In *Proceedings of the Formal Ontology in Information Systems (FOIS)* (pp. 159-174).
- Brachman, R. J. (2002). Systems that know what they're doing. *IEEE Intelligent Systems*, 17(6), 67-71.
- Bunge, M. (1977). *Treatise on Basic Philosophy: Ontology I: The Furniture of the World*. Springer.
- Fiorini, S. R., et al. (2017). Extensions to the core ontology for robotics and automation. *Robotics and Computer-Integrated Manufacturing*, 33, 3-11.
- Gangemi, A., et al. (2002). Sweetening ontologies with DOLCE. In *International Conference on Knowledge Engineering and Knowledge Management* (pp. 166-181). Springer.
- Guarino, N. (1998). Formal ontology and information systems. In *Proceedings of FOIS* (Vol. 98, pp. 81-97).
- Guiochet, J., Machin, M., & Waeselynck, H. (2017). Safety-critical advanced robots: A survey. *Robotics and Autonomous Systems*, 94, 43-52.
- Hepp, M., Leymann, F., Domingue, J., Wahler, A., & Fensel, D. (2006). Semantic business process management: A vision towards using semantic Web services for business process management. In *IEEE International Conference on e-Business Engineering* (pp. 535-540).
- IEEE SA (2015). *IEEE Standard Ontologies for Robotics and Automation* (IEEE Std 1872-2015).
- IEEE SA (2022). *IEEE Standard for Autonomous Robotics (AuR) Ontology* (IEEE Std

1872.2-2021).
- Lamy, J. B. (2017). Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80, 11-28.
- Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2), 141-160.
- Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), 33-38.
- Lukyanenko, R., Evermann, J., & Parsons, J. (2021). The IQ of crowds: Understanding and improving information quality in structured user-generated content using the BWW framework. *Information Systems Research*, 32(4), 1242-1265.
- Mascardi, V., Cordì, V., & Rosso, P. (2006). A comparison of upper ontologies. In *WOA* (Vol. 2007, pp. 55-64).
- Musen, M. A. (2015). The Protégé project: A look back and a look forward. *AI Matters*, 1(4), 4-12.
- Niles, I., & Pease, A. (2001). Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems* (Vol. 2001, pp. 2-9).
- Pease, A. (2011). *Ontology: A Practical Guide.* Articulate Software Press.
- Prestes, E., et al. (2013). Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61(11), 1193-1204.
- Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
- Sanz, R., López, I., Hernández, C., & Gómez, P. (1999). Reusable frameworks for complex domain modeling. In *Working Notes of the KI'99 Workshop on Application of Ontologies and Problem-Solving Methods.*
- Sanz, R., López, I., & Hernández, C. (2000). Self-awareness in real-time cognitive control architectures. In *Proceedings of the AAAI Fall Symposium on Self-Awareness.*
- SEBoK Editorial Board (2023). *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 2.9. The Trustees of the Stevens Institute of Technology.
- Shapiro, S. C. (Ed.). (2003). *Encyclopedia of Artificial Intelligence.* Wiley.
- Vernon, D. (2014). Artificial cognitive systems: A primer. *MIT Press.*
- Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal*, 3(4), 217-237.

1.19 Document Metadata

Last Updated: 2024-12-24 **Review Status**: Comprehensive enhancement from original PDF **Primary Use**: Foundational reference for flyby-f11 ontology-based autonomy development **Related Documents**: ONTOLOGY_FOUNDATION.md, SYSTEM_CONSTRAINTS.md