applied sciences

MDPI

Article

A Methodological Framework for Runtime Ontology Evolution in Dynamic Environments

Valeria Seidita *, Lucrezia Mosca and Antonio Chella

Dipartimento di Ingegneria, Universitá degli Studi di Palermo, Viale delle Scienze, 90128 Palermo, Italy; lucrezia.mosca@community.unipa.it (L.M.); antonio.chella@unipa.it (A.C.)
* Correspondence: valeria.seidita@unipa.it

Abstract

Intelligent systems operating in real-world environments are often required to make decisions in contexts that are only partially known at design time. In such scenarios, the assumption of a static and fully specified knowledge base becomes unrealistic, limiting the system's ability to adapt to novel situations. This challenge is particularly relevant for robotic systems, whose behavior cannot be entirely pre-programmed when operating in dynamic and evolving environments. This paper proposes a methodological and architectural approach for the runtime update of ontologies and knowledge bases, enabling intelligent systems to autonomously adapt their internal representation of the world during execution. The proposed approach enables the system to identify knowledge gaps by distinguishing between previously unknown concepts and known concepts enriched with newly observed instances, and to integrate such information into the ontology in a controlled and consistent manner. The approach is implemented as an end-to-end pipeline that combines visual perception, semantic interpretation through large language models, and a robust ontology update mechanism. Particular attention is devoted to ensuring formal consistency during runtime evolution, addressing challenges such as the generation of valid OWL constructs, the management of inverse properties, datatype normalization, and the prevention of semantic degradation over iterative updates. By enabling knowledge-driven adaptation at runtime, the proposed framework supports autonomous decision-making in environments that cannot be fully anticipated at design time. The approach was developed within the MUSIC4D and MHARA projects, which explore the use of intelligent systems in dynamic, partially structured contexts, focusing on knowledge-based adaptation.

Keywords: ontology update; knowledge-based systems; large language models; intelligent systems

Received: 14 February 2026
Revised: 23 March 2026
Accepted: 24 March 2026
Published: 3 April 2026

1. Introduction

Intelligent systems operating in real environments are increasingly required to make decisions in dynamic, partially structured contexts that are only partly known during the design phase. In such scenarios, the assumption of complete and static environmental knowledge is unrealistic, making it difficult, if not impossible, to define the system's behaviour a priori for all possible situations. This issue is particularly relevant in robotic systems, which must interact with complex physical environments, recognise new elements, adapt to unexpected configurations, and modify their behaviour during execution [1,2]. Without knowledge-based adaptation mechanisms, the system's behaviour is rigid and

highly dependent on manual design and programming, limiting its autonomy and effectiveness in unpredictable contexts. This highlights the need for approaches that enable intelligent systems to update their representation of the world at runtime, integrate newly observed information from the environment, and use it directly to guide decision-making. In this context, knowledge representation plays a central role in supporting the behaviour and decision-making of intelligent systems. In particular, ontologies provide a formal and explicit model of the domain of interest, allowing concepts, relationships, and constraints to be described in a structured and shared manner, and enabling forms of automatic reasoning based on rigorous semantics [3–5]. For these reasons, ontologies are widely adopted as a fundamental component in knowledge-oriented intelligent and robotic systems.
However, in most existing applications, ontologies are designed, populated, and validated offline, implicitly assuming that domain knowledge remains largely unchanged while the system is running [6]. This assumption is limiting in dynamic environments, where new entities, configurations, and relationships continually emerge from interaction with the real world. Without runtime update mechanisms, the ontology progressively loses alignment with the operating environment, reducing the effectiveness of reasoning and, consequently, the system's decision-making process [7].

The need to keep knowledge representation aligned with the evolution of the operating environment is widely recognised in the literature on ontological evolution and dynamic knowledge representation [7,8]. More recent studies also investigate the evolution of knowledge graphs and dynamic semantic representations in intelligent systems [9–11].

In dynamic and open contexts, this separation between the execution phase and the knowledge update phase is problematic. An intelligent system operating at runtime in an environment that is not fully known must be able to autonomously recognise when its representation of the world is incomplete or obsolete, and intervene to fill these gaps without interrupting the operational cycle. In particular, it is necessary to distinguish between situations in which new concepts emerge, not foreseen in the initial modelling, and situations in which already known concepts must be enriched with new instances observed in the environment [12].

This need highlights a limitation of traditional approaches to ontological evolution and calls for solutions that enable controlled updating of knowledge at runtime, while preserving the formal coherence of the model and its usability in the system's reasoning and decision-making processes.

In this paper, we propose a methodological and architectural approach for the runtime updating of ontologies and knowledge bases, aimed at supporting the decision-making processes of intelligent systems operating in dynamic and partially known environments. The approach enables the system to autonomously integrate new knowledge observed in the environment during execution, managing both the introduction of new concepts and the enrichment of existing concepts through the creation of new instances, without requiring a prior design or programming phase for behaviour.

The contribution is realised through an end-to-end pipeline that connects the perception of the environment to the symbolic representation of knowledge, allowing the transformation of heterogeneous information into formally coherent ontological constructs that can be directly exploited during execution [1].

The approach was developed within the MUSIC4D and MHARA projects, which address the use of intelligent systems in dynamic, collaborative, and highly variable contexts. The MUSIC4D project focuses on the autonomous and adaptive creation of creative content to enhance the enjoyment and experience of a musical work, while the MHARA project aims to develop an intelligent system to support the daily activities of elderly people, with the goal of counteracting cognitive impairment. Despite the diversity

of application domains, both projects share the need to operate in environments that are not completely predictable, where the system's ability to dynamically update its knowledge representation is a fundamental requirement.

To enable dynamic and controlled knowledge updates at runtime, it is necessary to adopt a representation that is formally rigorous, incremental, and easily manipulable by automated components. In this work, knowledge is modelled using subject–predicate–object triplets, which constitute the atomic unit of semantic representation in the Semantic Web and form the foundation of ontological languages such as OWL and RDF [13–15].

The use of triplets allows complex information to be decomposed into elementary and independent elements, facilitating the incremental integration of new knowledge during system execution. This representation explicitly distinguishes between concepts, instances, and relationships, supporting both the extension of the ontological schema when previously unmodelled concepts arise and the population of the ontology with new instances associated with already known concepts [16].

This feature is crucial in runtime update contexts, where information observed in the environment must be validated, normalised, and integrated into the ontological model without compromising the overall consistency of knowledge. In the proposed approach, triplets represent a key mechanism for operationalising the controlled evolution of knowledge and for directly linking environmental perception to the system's reasoning and decision-making processes [17–19].

The main contributions of this work are as follows:

- A methodological approach for run-time updating of ontologies and knowledge bases, designed to support the decision-making of intelligent systems operating in dynamic and partially known environments, overcoming the assumption of static knowledge defined a priori.
- An explicit distinction between conceptual updating and instance-level updating of knowledge, enabling the system to handle both the emergence of new concepts not previously modelled and the integration of new instances associated with already known concepts, observed during execution.
- An end-to-end architectural pipeline that connects the perception of the environment to the symbolic representation of knowledge, allowing the transformation of heterogeneous information into formally coherent ontological constructs that can be directly used in reasoning processes.
- A run-time controlled ontological update mechanism designed to preserve the formal consistency of the OWL model as knowledge evolves, addressing issues such as reverse property handling, datatype normalisation, and prevention of semantic degradation during iterative updates.
- A conceptual validation of the approach in dynamic application contexts, developed within the framework of the MUSIC4D and MHARA projects, which represent significant scenarios for the study of intelligent systems capable of adapting their behaviour on the basis of constantly evolving knowledge.

2. Related Work

The problem of representing and managing knowledge in intelligent systems has been extensively investigated in the fields of knowledge representation and artificial intelligence. Ontologies have emerged as a central paradigm for modelling domain knowledge in a formal and explicit way, enabling shared understanding, semantic interoperability, and automated reasoning. Foundational works have established the theoretical basis of ontologies and their role in knowledge-based systems, defining principles for modelling concepts, relations, and constraints using logic-based formalisms [3–5,16]. Ontological languages

such as OWL and RDF have further consolidated this approach, providing standardised representations grounded in description logics and Semantic Web technologies [13,14].

Despite their widespread adoption, most ontology-based systems assume that the knowledge model is designed, populated, and validated offline. This assumption has motivated a substantial body of research on ontology evolution and versioning, aimed at maintaining the consistency and relevance of ontological models as domains change over time. Surveys and classification frameworks have analysed the different types of ontology changes and proposed methods for managing schema evolution, instance updates, and consistency preservation [7,12,20]. However, the majority of these approaches focus on offline or semi-automated processes, often requiring human supervision and separating knowledge evolution from system execution [21]. As a result, they are not directly applicable to intelligent systems that must operate continuously in dynamic and partially unknown environments.

In the context of robotics and autonomous systems, ontologies have been successfully employed to support perception, reasoning, and high-level decision-making. Several works have demonstrated the benefits of integrating symbolic knowledge representations with sensor data and action models, enabling robots to reason about objects, actions, and situations at an abstract level. These approaches highlight the importance of structured knowledge for autonomy, but typically rely on pre-defined ontological models and limited mechanisms for runtime adaptation [1,2,22–24].

More recently, interest has grown in combining symbolic knowledge representations with data-driven and sub-symbolic approaches, giving rise to neuro-symbolic AI paradigms. Such approaches aim to exploit the expressive power and flexibility of learning-based models while preserving the interpretability and formal guarantees of symbolic reasoning [17–19]. Large language models have further stimulated this line of research, demonstrating remarkable capabilities in semantic interpretation and knowledge extraction [17–19,25], but also raising concerns regarding reliability, consistency, and controllability when used in isolation [25,26]. Recent studies have explored the integration of LLMs with symbolic structures to support reasoning and decision-making, although the problem of controlled ontology and knowledge base updates at runtime remains largely open [27,28].

The approach proposed in this work lies at the intersection of these research directions. Unlike traditional ontology evolution methods, it explicitly addresses the problem of updating ontologies and knowledge bases during system execution. At the same time, it differs from purely neuro-symbolic or LLM-centric approaches by introducing a controlled architectural separation between semantic interpretation and formal knowledge management. In this way, the proposed approach aims to preserve the benefits of symbolic reasoning while enabling autonomous and incremental knowledge updates driven by observations from dynamic environments.

3. Methodological Approach and Architecture

This section outlines the methodological approach and proposed architecture for updating the ontology and knowledge base at runtime. Its objective is to provide a clear, structured overview of the guiding principles and architectural choices made to ensure operational effectiveness in dynamic and partially known contexts. The section first presents the general methodological framework, clarifying the levels of abstraction and the role of the architecture in supporting the evolution of knowledge during execution, then details the pipeline design, the knowledge representation model, and the mechanisms for updating at runtime.

The following subsections describe the proposed modular architecture, the representation model based on triplets and ontologies, and the methods by which new knowledge observed in the environment is integrated into the model in a controlled and coherent manner, preserving its usability in the system's reasoning and decision-making processes [1,2].

3.1. Overview of the Proposed Methodological Approach

The approach proposed in this paper operates at the methodological and architectural levels, with the main objective of enabling the runtime updating of knowledge in intelligent systems functioning in dynamic and partially known environments [2,29–33]. Specifically, the approach aims to support the system's decision-making process through the controlled evolution of the ontology and its knowledge base during execution, overcoming the assumption of a static and exclusively predefined domain representation at the design stage [7,12].

From a methodological perspective, the approach is based on the idea that the system's autonomy derives from its ability to recognise and manage gaps in its knowledge, distinguishing between information already conceptually modelled and new information observed in the environment. This requires a distinction between conceptual and instance-level updates, which has been recognised as a key issue in ontology evolution and knowledge management [20].

The methodological approach is realised through a modular architecture, organised as a pipeline connecting environmental perception to the symbolic representation of knowledge. This pipeline transforms heterogeneous, potentially unstructured information into formally coherent ontological constructs, which can be integrated at runtime into the ontology and used directly in the system's reasoning and decision-making processes [13,14]. The modularity of the architecture enables a clear separation of the different process steps and promotes the extensibility and reusability of individual components.

It is important to emphasise that the proposed approach is not tied to any specific perception technology or particular application domain. On the contrary, it defines a general framework for the runtime updating of knowledge, which can be instantiated in various application contexts.

A fundamental aspect of the approach is the adoption of a knowledge representation based on subject–predicate–object triplets, consistent with OWL-based formalisms and the principles of symbolic knowledge representation [3,16]. This choice enables explicit modelling of relationships between entities in the environment and, in particular, the representation of actions as semantic relations between an active subject and an involved object, where the predicate captures the nature of the action itself.

Triplet-based representation also supports the modelling of inverse and reflexive relations, allowing the description of situations in which an entity undergoes an action through inverse properties. This mechanism preserves a coherent and queryable description of possible interactions between agents and objects, strengthening the connection between knowledge representation and the system's reasoning and decision-making processes.

3.2. Architecture Layers

The architecture separates the responsibilities of the main functional components, distinguishing between perception, semantic interpretation, knowledge management, and reasoning support [22,23]. This organisation enables the isolation of formal knowledge updating mechanisms from the specific technologies used for perception and interpretation, promoting the extensibility of the approach and its applicability to various contexts and domains.

In particular, the knowledge management layer is responsible for updating the ontological schema when new concepts not previously modelled are encountered, as well as for populating and updating the knowledge base by integrating new instances and observed facts from the environment. Figure 1 provides an overview of the proposed architecture, highlighting the main functional layers and the information flows between them.

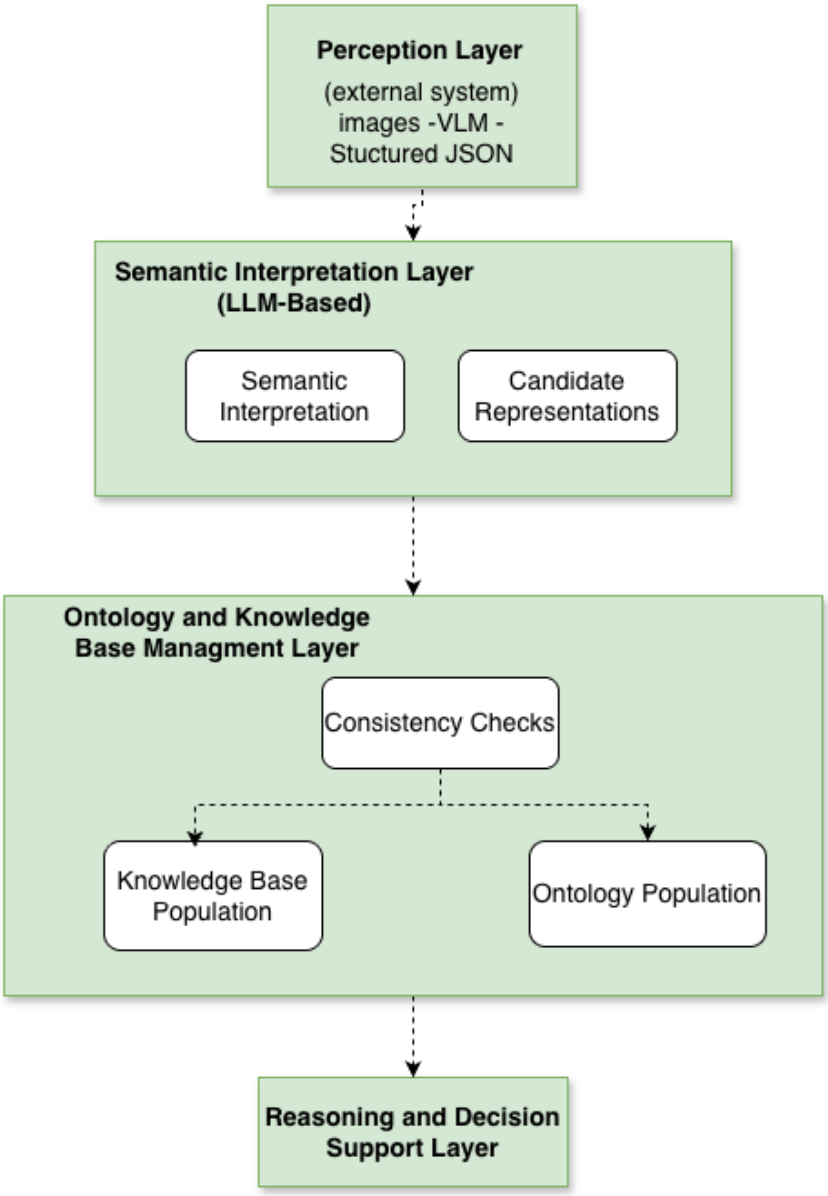

Figure 1. Layered architecture supporting runtime knowledge updating. The architecture separates perception, semantic interpretation, formal knowledge management, and reasoning support, enabling the controlled transformation of environmental observations into validated ontological knowledge while preserving the consistency of the symbolic model.

Based on this architectural structure, the process of updating knowledge is implemented as a pipeline that progressively transforms observations of the environment into formally coherent symbolic representations, as described below.

3.2.1. Perception Layer

The *Perception Layer* is responsible for capturing observations from the operating environment and provides the initial input to the knowledge update process. In this work, the *Perception Layer* is deliberately treated as a black box, and its outputs are assumed to

be structured observations of the environment. The proposed architecture is intentionally designed to remain independent of specific vision or multimodal models. In the prototype implementation, the *Perception Layer* provides structured observations derived from vision-based models, but the architecture itself does not depend on any particular perception technology. This choice allows the focus to remain on the mechanisms of semantic interpretation and knowledge updating at runtime, independently of the underlying perception technologies [34].

The information produced by the *Perception Layer* represents a preliminary description of the environment, including detected objects, perceptual properties, and spatial relations, but cannot be directly integrated into the ontology. Structured representations such as JSON files serve as the operational starting point of the update process, providing explicit descriptions of detected entities and their attributes.

The JSON files produced by the perception system are the operational starting point of the knowledge update process. They provide a structured representation of the detected entities in the environment, organised as a set of attributes associated with each observed object. As shown in Figure 2, each object is identified by a name and a class to which it belongs, which represent the key information used in the subsequent phases of semantic interpretation and ontological updating.


```
{
    "name": "laptop",
    "class": "electronic device",
    "color": "silver",
    "shape": "rectangular",
    "texture": "smooth",
    "position": {
        x="86.5",
        y="28.6"
    },
    "confidence": 0.85,
    "definition": "portable computer with a built-in screen",
    "actions": "open, close, type, surf internet"
}
```


Figure 2. Example of a structured JSON file generated by the *Perception Layer* in a simplified indoor scenario.

In addition to this central information, the structure includes further attributes related to perceptual characteristics (e.g., colour, shape, and texture), spatial position expressed in Cartesian coordinates, a confidence measure associated with recognition, a textual description, and a set of actions or affordances that describe possible interactions with the object. These attributes are domain-specific elements of interest and may vary depending on the scenario.

The example shown in the figure is for illustrative purposes only and refers to a simplified scenario in which the system operates within a room containing objects that must be identified based on properties such as colour, shape, or name and subsequently manipulated, such as being moved or grabbed. In this context, the JSON file provides the information necessary for the system to recognise the relevant objects and support subsequent operational decisions.

For this reason, the *Perception Layer* does not play any active role in updating the ontology or knowledge base, but merely provides structured observations that are subsequently processed by the Semantic Interpretation Layer.

3.2.2. Semantic Interpretation Layer

The *Semantic Interpretation Layer* transforms the structured observations provided by the *Perception Layer* into an explicit semantic representation suitable for formalisation. This layer employs large language models to interpret the input information, identify relevant entities, properties, and relations, and generate candidate subject–predicate–object triplets [25].

From this information, the layer generates a preliminary semantic representation in the form of **subject–predicate–object candidate triplets**, which explicitly describe the concepts and instances observed in the environment, along with the relationships connecting them. More precisely, the language model is used to produce an intermediate structured semantic representation of the observed input. The subject–predicate–object candidate triplets are then derived from this representation and provided as input for the knowledge management phase, rather than being integrated directly into the ontology. They represent a semantic interpretation of the perceptual input, which may be incomplete, ambiguous, or affected by uncertainties, and therefore require a subsequent phase of validation and formalisation. For this reason, the *Semantic Interpretation Layer* does not play any direct role in updating formal knowledge, but provides the *Ontology and Knowledge Base Management Layer* with a set of candidate triplets that form the basis for the controlled evolution of the ontological model.

The separation between semantic interpretation and knowledge management enables the isolation of mechanisms based on probabilistic and linguistic models from symbolic and formal ones, reducing the risk of inconsistencies and preserving the reliability of the runtime update process. Furthermore, this architectural choice makes the approach flexible with respect to the evolution of semantic interpretation technologies, allowing the *Semantic Interpretation Layer* to be replaced or updated without modifying the underlying knowledge management mechanisms.

In practice, the output of the *Semantic Interpretation Layer* takes the form of an intermediate structured representation, similar to a dictionary of semantically relevant attributes, generated from the perceptual input. This representation does not yet correspond to an ontological formalisation, but organises information in a way that is consistent with the conceptual schema of the domain, making explicit elements such as the entities involved, potentially relevant relationships or actions, and the associated semantic roles. A detailed example of such an interim representation can be found in the Supplementary Materials.

From this intermediate structure, subject–predicate–object triplets are subsequently derived and subjected to the validation and integration mechanisms provided by the knowledge management level. This intermediate step keeps the results of linguistic interpretation separate from the formal constraints of the ontology, maintaining control over the evolution of knowledge at runtime.

The validation and normalisation phase aims to ensure the formal correctness and structural integrity of the ontology during runtime updates. As candidate symbols are derived from the output of a large language model, they may be syntactically invalid, ambiguous, or incompatible with the constraints imposed by OWL-based representations.

The validation process operates at the level of ontological symbols, including class names, individual identifiers, and property labels, and is applied before any integration into the ontology or associated knowledge base. Its purpose is to ensure that dynamically generated symbols comply with formal naming conventions and do not introduce conflicts with reserved terms or internal constructs of the ontology framework.

By decoupling semantic interpretation from formal validation, this mechanism establishes a protective symbolic barrier between sub-symbolic, probabilistic components and the symbolic knowledge representation. As a result, the ontology can be safely updated at

runtime while preserving the robustness, consistency, and reproducibility of the knowledge evolution process.

For example, identifiers generated during semantic interpretation may contain leading or trailing whitespace, numeric prefixes, or reserved keywords, which are not permitted in OWL-based ontologies. During the validation phase, such identifiers are automatically normalised (for instance, by trimming whitespace or adding safe prefixes) before being integrated into the ontology.

3.2.3. Ontology and Knowledge Base Management Layer

The *Ontology and Knowledge Base Management Layer* governs the validation, normalisation, and integration of candidate triplets into the ontological model and its knowledge base. This layer distinguishes between updates that require modifications to the ontological schema and updates that can be handled through the creation of new instances associated with existing concepts [7,12].

In particular, the layer explicitly distinguishes between two types of knowledge updates. On the one hand, it manages **ontological updating**, introducing new concepts and relationships into the ontological schema when information observed in the environment cannot be traced to previously modelled concepts. On the other hand, it supports **updating the knowledge base** by populating the ontology with new instances and facts associated with already known concepts, derived from observations made at runtime. This distinction enables systematic management of both the structural and instantiated evolution of knowledge.

The process of integrating candidate triplets includes a formal validation phase, aimed at ensuring the compatibility of new information with the existing ontological model. In this step, the layer verifies the syntactic and semantic correctness of the triplets, compliance with the constraints defined in the ontology, and the overall consistency of the resulting model. Particular attention is given to managing critical aspects such as datatype normalisation, the definition and updating of inverse properties, and the prevention of inconsistencies that could compromise the effectiveness of reasoning processes.

Once validated, the information is integrated into the ontology and knowledge base, making the new knowledge immediately available to the modules that use it. In this way, the layer enables the system instantiating the architecture to maintain a representation of the domain that remains aligned with the evolution of the operating environment, without requiring manual intervention or offline upgrade phases. The *Ontology and Knowledge Base Management Layer* thus constitutes the main mechanism through which the proposed approach enables knowledge-based adaptation and supports the decision-making process at runtime.

The validation process ensures the syntactic and semantic correctness of dynamically generated symbols, the consistency of datatype usage, and the correct handling of inverse properties, preserving the overall coherence of the ontology [14]. Only validated information is integrated into the formal knowledge representation, enabling safe and controlled evolution at runtime.

3.2.4. Reasoning and Decision Support Layer

The Reasoning and Decision Support Layer exploits the updated ontology and knowledge base to support reasoning and decision-making activities. Although not the primary focus of this work, its inclusion highlights that runtime knowledge updating is intended to directly support adaptive behaviours based on formal reasoning [1,23]. In particular, the presence of this layer highlights that the controlled evolution of the ontology and

knowledge base at runtime is not an end in itself, but is intended to make knowledge immediately usable in the system's decision-making processes.

The inclusion of this layer also allows the architecture to remain general and reusable in different application contexts, where various reasoning or decision-making strategies can be adopted without requiring changes to the semantic interpretation and knowledge management mechanisms described in this work.

3.3. Runtime Knowledge Update Pipeline

Starting from the architecture described above, the runtime knowledge update process is implemented as a pipeline that incrementally transforms environmental observations into formally coherent symbolic representations. The pipeline defines the operational flow across the architectural layers and enables the integration of new knowledge without interrupting system execution [35].

The pipeline includes phases for structured observation acquisition, semantic interpretation, candidate triplet generation, validation, and integration into the ontology and knowledge base. Figure 3 illustrates the main steps of the process and their interactions.

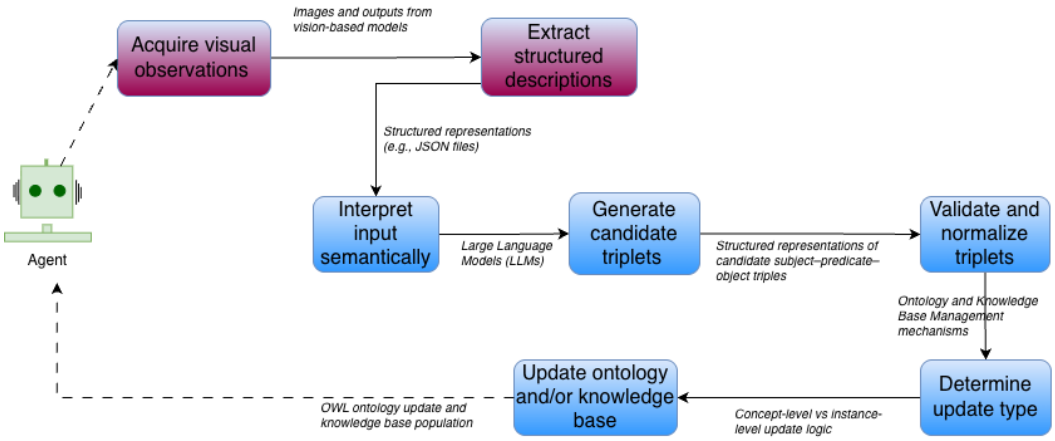

Figure 3. Runtime knowledge update pipeline. The process progressively transforms structured observations into candidate semantic representations and validated subject–predicate–object triples, which are then integrated into the ontology and knowledge base through controlled validation and update mechanisms.

3.3.1. Step 1—Acquisition of Structured Observations

The pipeline begins with the capture of environmental observations provided by the Perception Layer. In this phase, the system receives structured information, such as images, vision model outputs, and representations in JSON format, which describe what is detected in the operating environment. This information constitutes a preliminary description of the observed domain and is not yet expressed in symbolic form.

3.3.2. Step 2—Semantic Interpretation and Candidate Triple Generation

The structured observations are then processed by the Semantic Interpretation Layer, which interprets the information content of the inputs and transforms it into an explicit semantic representation.

The triplets generated in this phase represent a semantic interpretation of the observed environment and may include references to concepts already present in the ontology, as well as information suggesting the introduction of new concepts or instances. However, these triplets are not yet integrated into formal knowledge and are considered preliminary inputs for the knowledge management phase. This phase also filters out semantically

inconsistent or implausible interpretations with respect to the observed context, reducing the risk of propagating erroneous knowledge into the formal representation.

3.3.3. Step 3—Triple Validation and Knowledge Update

The candidate triplets produced by the semantic interpretation phase are then forwarded to the Ontology and Knowledge Base Management Layer, which governs their validation and integration into the system's formal knowledge. In this step, the layer checks the compatibility of the triplets with the existing ontological model and distinguishes between updates that require changes to the ontological schema and those that can be managed by populating the knowledge base with new instances.

The validation process includes checks for syntactic and semantic correctness, datatype normalisation, and reverse property management, in order to preserve the overall consistency of the ontology. Only triplets that pass this validation step are integrated into the ontology and knowledge base, making the new knowledge immediately available to downstream modules.

3.3.4. Step 4—Knowledge Availability for Reasoning

Once the knowledge update is complete, the updated ontology and knowledge base can be used by the Reasoning and Decision Support Layer to support reasoning and decision-making activities. Although these activities are not the focus of this work, the pipeline is designed to make updated knowledge immediately actionable, highlighting the role of runtime refresh as a prerequisite for adaptive knowledge-based behaviours.

To clarify the relationship between the conceptual steps of the proposed pipeline and their concrete realisation within the prototypical system, Table 1 summarises the mapping between each pipeline phase and the corresponding implementation elements. This mapping is provided for explanatory purposes and is intended to illustrate one possible instantiation of the proposed approach, without limiting its applicability to specific technologies or tools.

Table 1. Mapping between the conceptual steps of the runtime knowledge update pipeline and the corresponding implementation elements in the prototype instantiation.

Pipeline Step	Implementation Elements
Acquire visual observations	Images and outputs produced by vision-based models
Extract structured descriptions	Structured representations of perceptual data (e.g., JSON files)
Interpret input semantically	Large Language Models (LLMs) for semantic interpretation
Generate candidate triples	Structured representations of candidate subject–predicate–object triples
Validate and normalize triples	Ontology and Knowledge Base Management mechanisms, including consistency checks
Determine update type	Logic for distinguishing concept-level and instance-level updates
Update ontology and knowledge base	OWL ontology update and population of the knowledge base

4. Validation in Dynamic Scenarios

In this section, we present a validation of the proposed approach, aimed at verifying the correctness and feasibility of the mechanisms for updating knowledge at runtime described in the previous sections. The validation is not intended to provide a quantitative assessment of system performance, but rather to demonstrate that the architecture and pipeline enable the ontology and knowledge base to be autonomously and consistently updated in response to observed changes in the operating environment, in line with established practices for qualitative validation of knowledge-based systems [7,35].

To this end, the approach was applied in a set of dynamic scenarios within a laboratory environment, designed to highlight the main capabilities of the system in terms of recog-

nising knowledge gaps, distinguishing between conceptual and instance-level updates, and controlled integration of new knowledge. The following subsections describe the validation context, the scenarios considered, and a discussion of the behaviour observed during execution.

4.1. Validation Setup

It should be noted that the validation, conducted in a laboratory environment, is qualitative and oriented towards verifying the mechanisms for runtime updating of the ontology and knowledge base, rather than measuring performance metrics or comparing alternative approaches [21].

The validation context consists of a work environment in which the system is required to operate with an initial partial knowledge of the domain. Some elements of the environment are known to the system at design time, while others are introduced or modified during execution, requiring an update to the knowledge representation. Observations of the environment are provided to the system via the Perception Layer, while the subsequent semantic interpretation and knowledge management steps operate according to the pipeline described in Section 2.

Validation focuses in particular on the ability of the system to (i) identify gaps in its representation of the domain as a result of new observations, (ii) distinguish between situations that require conceptual updating and those that can be managed by populating the knowledge base with new instances, and (iii) integrate new knowledge while preserving the formal coherence of the ontology, as discussed in the literature on ontology evolution and dynamic knowledge bases [12,20].

4.2. Runtime Knowledge Update Scenarios

To illustrate the functioning of the proposed approach, the validation refers to a set of scenarios inspired by the MUSIC4D project, characterised by creative and collaborative environments in which the configuration of the space and the elements present can vary over time. Such contexts are representative of dynamic application domains in which intelligent systems must continuously adapt their internal knowledge representation to remain operationally effective [22].

4.2.1. Scenario 1: Known Concept, New Instance

In the first scenario, the system operates in an environment where some relevant concepts are already modelled in the initial ontology, but not all the corresponding instances are known. For example, the ontology includes the concept of a musical instrument, along with a set of properties that describe its general characteristics, but does not contain information about a specific instrument present in the operating environment.

During execution, the Perception Layer provides the system with new observations that include the presence of a previously unknown musical instrument. Following the semantic interpretation of the input, the Semantic Interpretation Layer generates candidate triplets that describe the observed object as an instance of an already known concept. In this case, the Ontology and Knowledge Base Management Layer recognises that no modification of the ontological schema is required and proceeds to populate the knowledge base by introducing a new instance and associating the relevant properties with it.

This scenario highlights the system's ability to enrich its knowledge at runtime through the integration of new instances, keeping the conceptual structure of the ontology unchanged and preserving its formal coherence, in accordance with established approaches to instance-level ontology evolution [7].

4.2.2. Scenario 2: Concept Not Previously Modelled

The second scenario considers a situation in which the system observes elements of the environment that are not attributable to concepts already present in the initial ontology. In a creative context such as MUSIC4D, this can occur, for example, when a device or tool is introduced with functional characteristics not foreseen during the design phase.

In this case, the observations provided by the Perception Layer and interpreted by the Semantic Interpretation Layer lead to the generation of candidate triplets that do not find a direct correspondence in the existing ontological schema. The Ontology and Knowledge Base Management Layer then identifies a gap at the conceptual level and activates an ontological updating process, introducing a new concept and the related relationships necessary to correctly represent the observed element, as described in approaches addressing schema-level ontology evolution [12,20].

Once the ontological schema has been updated, the system proceeds to populate the knowledge base with the observed instances, making the new knowledge immediately available for downstream reasoning processes. This scenario highlights the approach's ability to manage both instance enrichment and structural evolution of knowledge at runtime, without requiring manual intervention or offline update phases.

Although the scenario considered is deliberately circumscribed, it includes all the elements necessary to observe the system's behaviour in the presence of a conceptual gap and to verify the controlled activation of the ontological update mechanism at runtime. In this sense, the scenario allows validation of the fundamental principles of the approach, independently of the overall complexity of the application domain.

In this way, the proposed approach enables the system to adapt its representation of the domain in response to observed changes in the environment, distinguishing between situations that require simple population of the knowledge base and those that require modification of the ontological schema.

It should be noted that the proposed approach does not assume continuous and uninterrupted monitoring of the environment by the system. In the context of this work, the Perception Layer is considered as a source of observations activated according to the needs of the task and the operational context, rather than as a constantly active process. This assumption is consistent with common design choices in context-aware and knowledge-based systems.

Aspects related to computational resource management, processing times, and interaction fluidity in real-time scenarios are outside the scope of this validation and will be addressed in future developments, once the proposed methodological and architectural pipeline has been consolidated.

4.3. Quantitative Evaluation of the Symbolic Knowledge Management

In addition to the qualitative validation scenarios presented above, a small-scale quantitative evaluation was conducted to analyse the reliability of the symbolic knowledge management mechanisms within the proposed architecture. In particular, the evaluation focuses on the consistency of the symbolic retrieval process used by the Reasoning and Decision Support Layer when accessing information stored in the Knowledge Base after runtime updates.

The objective of this evaluation is not to measure the linguistic performance of the language model involved in the semantic interpretation stage, but rather to assess whether the knowledge produced by the pipeline can be reliably retrieved and used to support decision-making processes.

4.3.1. Experimental Setup

The evaluation is structured around a set of controlled scenarios, each representing a query posed to the system under a fixed state of the Knowledge Base. For each scenario, the expected outcome (ground truth) is defined based on the actual content of the ontology and the knowledge base, allowing the system output to be compared with the expected result.

More specifically, each scenario consists of a query expressed as a subject–predicate pair, where the predicate represents a boolean affordance property of the form *canBeX*. These properties represent the possibility of performing a specific action on an object within the environment.

4.3.2. Dataset

The evaluation was conducted using a dataset of 100 controlled scenarios designed to test the system's ability to distinguish between the presence and absence of information in the Knowledge Base. The dataset was balanced as follows:

- 50 scenarios in which the queried triple was present in the Knowledge Base;
- 50 scenarios in which the triple was absent.

For each scenario, the ground truth was determined a priori according to the actual state of the ontology and the knowledge base.

4.3.3. Evaluation Metrics

Given the deterministic nature of the symbolic retrieval process, the evaluation relies on metrics that measure the correctness of knowledge retrieval. The following metrics were considered:

Accuracy

$$Accuracy = \frac{N_{correct}}{N}$$

where $N_{correct}$ represents the number of correctly classified scenarios and N is the total number of evaluated scenarios.

False Missing Rate (FM Rate)

$$FMRate = \frac{FM}{N_{present}}$$

This metric measures the proportion of triples that were present in the Knowledge Base but were not retrieved by the system.

False Known Rate (FK Rate)

$$FKRate = \frac{FK}{N_{absent}}$$

This metric captures cases in which the system incorrectly assumes the presence of knowledge that is not contained in the Knowledge Base.

4.3.4. Results

The results obtained from the controlled evaluation scenarios provide a first quantitative indication of the reliability of the symbolic knowledge management mechanisms within the proposed architecture. The symbolic retrieval mechanism was evaluated on the 100 controlled scenarios described above. The system correctly classified 96 scenarios, achieving an overall retrieval accuracy of 96%.

Four cases were classified as *False Missing*, corresponding to scenarios in which the triple was present in the Knowledge Base but was not retrieved by the system. Considering

the 50 scenarios in which the triple was present, this corresponds to a False Missing Rate of 8%.

No cases of *False Known* were observed, resulting in a False Known Rate of 0%. This result indicates that the system does not generate decisions based on knowledge that is not explicitly represented in the Knowledge Base.

Overall, the results provide preliminary quantitative evidence that the proposed architecture enables reliable access to symbolic knowledge generated and updated through the pipeline, while preserving a conservative decision strategy that avoids unsupported inferences. These results complement the qualitative validation scenarios discussed earlier and provide preliminary empirical evidence supporting the robustness of the proposed knowledge management pipeline. Detailed examples of the pipeline execution, including perceptual inputs, candidate triplets, and the resulting ontology updates, are provided in the following section.

5. Example

In this paper, we provide a step-by-step example of the pipeline presented in Figure 2 of the paper, with the aim of clarifying how the proposed methodological and architectural approach operates in practice, allowing us to understand how the system works.

The example is intended as an explanatory walkthrough rather than an additional experimental validation. It illustrates how perceptual observations are progressively transformed into formally integrated knowledge through the interaction between the Semantic Interpretation Layer and the Ontology and Knowledge Base Management Layer. To make the process explicit and traceable, we consider a simplified scenario in which a Tiago Pro robot operates in a conservatory classroom with the goal of assisting teachers and students in moving objects within the environment.

During the design phase, an initial ontology is defined in Figure 4, representing the knowledge available to the system before execution. This ontology contains classes such as WritingSurface or Furniture, together with their known instances, modelling what designers and programmers assume about the working environment at design time. The robot itself is represented as an instance (Tiago), which is a relevant element for subsequent decision-making processes.

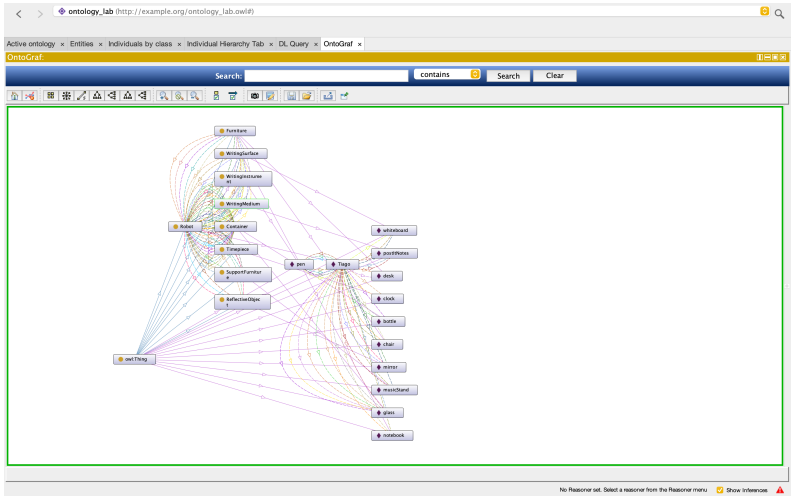

Figure 4. Initial ontology available at design time before runtime knowledge updates.

Indeed, the system evaluates requested or planned actions also according to the robot's own capabilities. For example, a robot such as Tiago can manipulate objects like a monitor, whereas a smaller platform such as NAO may not be able to do so. This self-representation

allows the system to justify its actions based on its internal description and on the domain knowledge available at runtime.

Now, suppose that the system is put in place and that the perception system detects the presence of a piano in the classroom. The piano is a musical instrument, and the output produced by the Perception Layer is like the one shown in Figure 5. In this figure, it is clear that the piano belongs to the Musical Instrument class and has a set of characteristics such as color, etc. This information is interpreted semantically and then passed to the module for generating candidate triplets, obtaining, for example, a pair of triplets such as the following two (Figure 6):

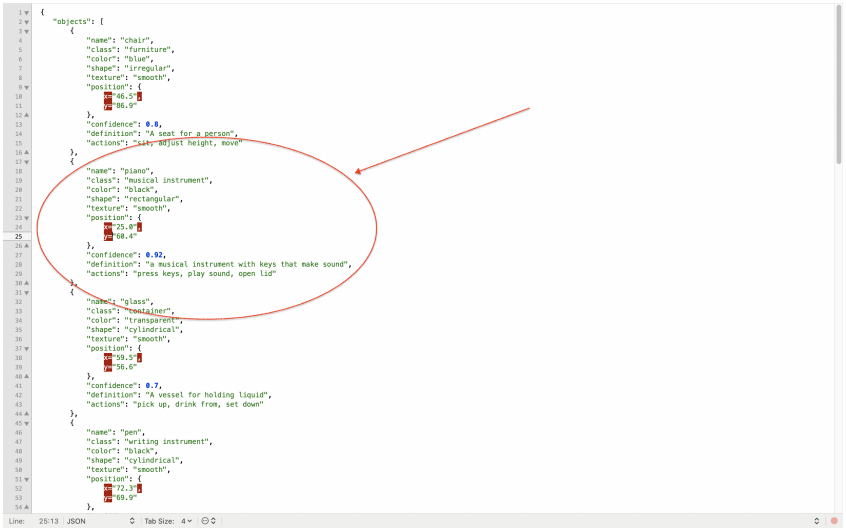

Figure 5. Structured JSON file generated by the Perception Layer describing the detected object and its attributes.

Figure 6. Candidate subject–predicate–object triples derived from the semantic interpretation of the perceptual input.

From which we see that the hasPosition predicate expresses the concept related to the position of the object and is of the type float, and again that the piano can be played and has the object True, so an action that can be done on the piano is to play. These builds are automatic and obtained with the help of the LLM from a prompt. The LLM is not used as a reasoning component but as a controlled semantic parser constrained by explicit generation rules. Its role is limited to producing a structured semantic representation of the perceptual input. The formal validation and ontological integration are entirely governed by the symbolic layer. Once the candidate representation has been generated, the Ontology and Knowledge Base Management Layer analyses it through schema-matching mechanisms against the existing ontology. Through this process, the system detects that no *MusicalInstrument* class is present in the initial ontology and therefore identifies a conceptual gap requiring an ontological extension. The detection of the conceptual gap is thus performed by the Ontology and Knowledge Base Management Layer, not by the LLM, which remains confined to the interpretation stage. The new class is then created and

populated as illustrated in the following figure (Protegé is used for ontology inspection, while Apache Fuseki supports RDF storage and SPARQL querying of the knowledge base).

Now suppose that, following a second survey of the environment, the robot realizes that there is a violin, another instrument whose class is now known, but there is no instance in the knowledge base. As can be seen from the sequence of Figures 7–10, after the extraction of the descriptors, the semantic interpretation and the generation of the correct triplets, the system focuses only on the generation of the new instance and generates the concept of the violin.

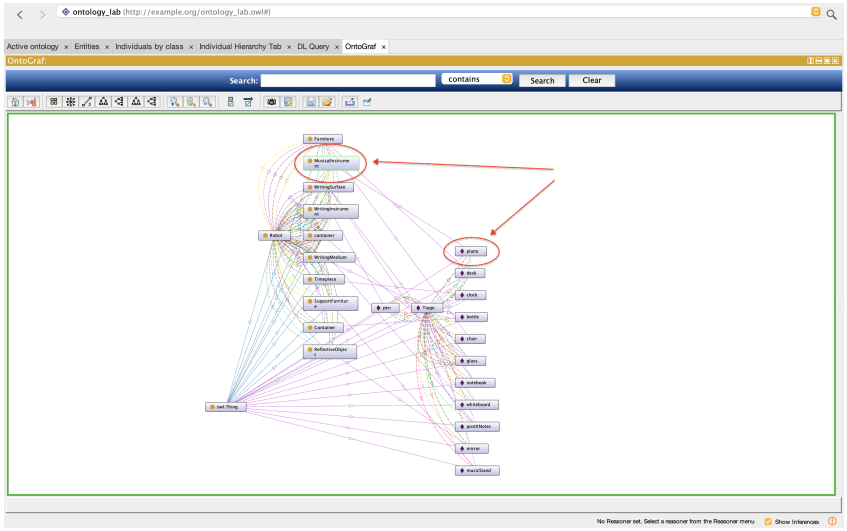

Figure 7. Extended ontology after detection of a conceptual gap and creation of the new class (MusicalInstrument) with its first instance (Piano).

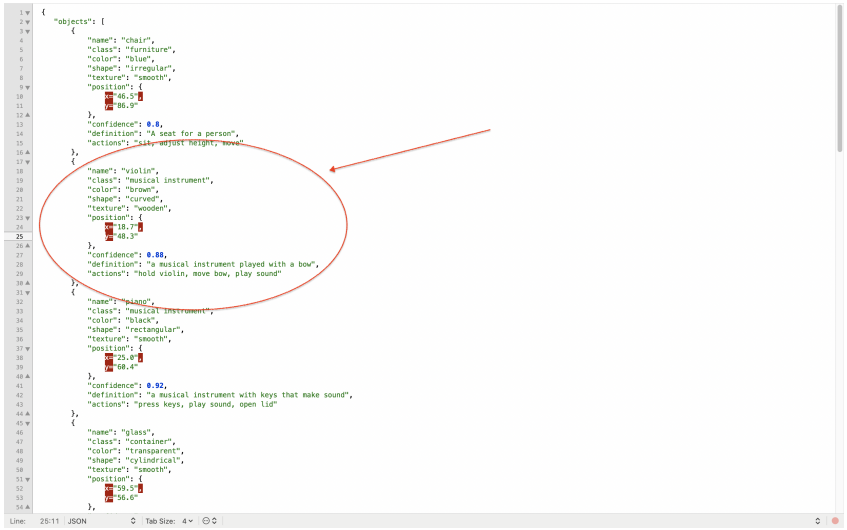

Figure 8. Structured JSON file describing a newly observed object (Violin) recognised as belonging to an already modelled concept.

```
{
  "subject": "violin",
  "subject_class": "MusicalInstrument",
  "predicate": "hasColor",
  "predicate_type": "DataProperty",
  "object": "brown",
  "object_class": "",
  "inverse_of": "",
  "datatype": "xsd:string"
},
```

```
{
  "subject": "violin",
  "subject_class": "MusicalInstrument",
  "predicate": "canBePlayed",
  "predicate_type": "DataProperty",
  "object": "true",
  "object_class": "",
  "inverse_of": "",
  "datatype": "xsd:boolean"
},
```

Figure 9. Candidate triples generated for the new observation, mapped to the existing ontological concept.

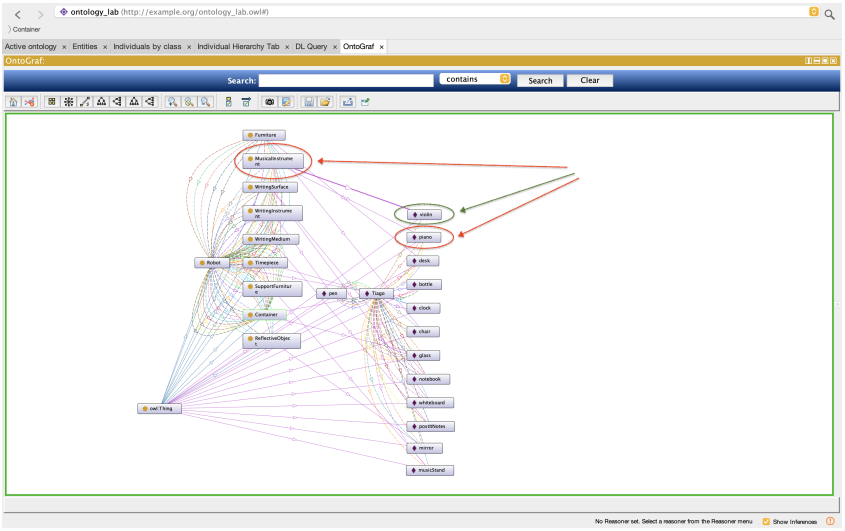

Figure 10. Ontology after runtime update limited to instance population, with no modification of the conceptual schema.

This example illustrates how the proposed pipeline enables a controlled transition from perceptual observations to formally validated ontological knowledge, ensuring that dynamically acquired information can be integrated without compromising consistency. The walkthrough highlights the complementary roles of LLM-based semantic interpretation and symbolic validation mechanisms in supporting runtime knowledge evolution.

6. Discussion

In this paper, we have presented a methodological and architectural approach to enable runtime updates of ontologies and knowledge bases in intelligent systems operating in dynamic and partially known environments. The validation scenarios considered demonstrate how the proposed system updates its knowledge representation in a differentiated manner, depending on the nature of newly observed elements. When observations can be mapped to concepts already present in the initial ontology, the update process is limited to instantiating new individuals and assigning the corresponding properties. Conversely, when observed elements do not have a direct correspondence in the existing ontological schema, the system identifies a conceptual gap and activates a controlled update of the ontology by introducing new concepts and the relations required to represent the observed entities correctly. Although the scenarios considered are intentionally circumscribed, they include the essential elements needed to validate the mechanisms underlying runtime conceptual and instance-level updates.

A central aspect of the proposed approach is the explicit separation between semantic interpretation and formal knowledge management. The use of large language models for interpreting perceptual input enables the extraction of rich and flexible semantic representations, but also introduces potential sources of ambiguity. By treating the representations produced by the Semantic Interpretation Layer as candidate knowledge and subjecting them to subsequent validation and normalisation, the architecture isolates such uncertainties and preserves the coherence of the ontological model. This design choice allows the system to leverage the expressive capabilities of language models while maintaining the reliability of the symbolic knowledge used during execution.

The distinction between conceptual updates and instance-level updates emerges as a key principle for managing the structured evolution of knowledge in dynamic environments. While the mere introduction of new instances is often insufficient to capture previously unmodelled aspects of the domain, the uncontrolled creation of new concepts

may lead to the unstructured growth of the ontology. The proposed approach provides a systematic mechanism to address both situations, enabling the system to extend the ontological schema only when required and to maintain a clear separation between conceptual structure and instantiated data.

During execution, the triplets generated through semantic interpretation are treated as candidate representations and validated before integration into the formal knowledge base. This mechanism prevents the direct insertion of ambiguous or inconsistent information into the ontology and supports the progressive enrichment of the knowledge representation without requiring manual intervention or offline reconfiguration. Although the Reasoning and Decision Support Layer is not actively used in the validation scenarios, its inclusion in the architecture highlights the role of knowledge update as a prerequisite for enabling adaptive, knowledge-driven behaviours. In this context, semantic degradation refers to the progressive introduction of inconsistent, ambiguous, or structurally invalid elements into the ontology during iterative updates. The proposed architecture mitigates this risk through a validation stage that checks the syntactic correctness of generated symbols, enforces datatype normalisation, and verifies compatibility with the existing ontology schema before integration. This mechanism ensures that runtime updates preserve the structural integrity of the knowledge representation.

From an applicability perspective, the scenarios inspired by the MUSIC4D project have been used as a reference context to observe system behaviour in dynamic settings. However, the proposed approach is conceived as general and domain-independent. Other contexts, such as those addressed in the MHARA project, share similar requirements for operating in partially known and evolving environments, making runtime knowledge update a transversal capability. In this sense, the architectural principles and the proposed pipeline can be instantiated across different application domains while preserving the same methodological foundations.

7. Conclusions

This paper presents a methodological and architectural framework designed to support runtime updates of ontologies and knowledge bases in intelligent systems operating in dynamic and partially known environments. The proposed approach introduces a modular pipeline that connects environmental perception, semantic interpretation, and formal knowledge management, enabling the controlled integration of new knowledge during system execution. By explicitly distinguishing between conceptual updates and instance-level updates, the framework allows intelligent systems to adapt their knowledge representation as new elements are observed in the environment.

The validation presented in this work is intended to demonstrate the feasibility and correctness of the proposed mechanisms rather than to provide a full quantitative performance evaluation or a comparison with alternative approaches. Nevertheless, the results obtained from the controlled evaluation scenarios provide preliminary empirical evidence supporting the reliability of the symbolic knowledge management mechanisms within the proposed architecture.

Overall, the study shows that the proposed pipeline enables the controlled integration of new knowledge at runtime while preserving the formal coherence of the ontological model. By combining semantic interpretation mechanisms with formal validation and knowledge management procedures, the architecture provides a structured framework for supporting knowledge-driven behaviour in dynamic environments. Additional examples illustrating the pipeline execution, including structured inputs, candidate triplets, and ontology updates, are provided in the Example section in this paper.

For future work, a natural extension is the full integration of the knowledge update mechanisms with reasoning modules explored in previous studies, to assess the impact of dynamic knowledge evolution on system behaviour. Additional developments include more extensive experimental evaluations in domain-specific scenarios and a deeper analysis of the structured inputs and semantic representations generated along the pipeline.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app16073494/s1, Supplementary Materials are available and include detailed examples of the pipeline execution, structured perceptual inputs, candidate triplets, and ontology updates.

Author Contributions: Conceptualization, V.S.; methodology, V.S.; software, L.M.; validation, L.M. and V.S.; formal analysis, V.S.; investigation, V.S. and L.M.; resources, V.S.; data curation, L.M.; writing—original draft preparation, L.M.; writing—review and editing, V.S. and A.C.; visualization, L.M.; supervision, A.C. and V.S.; project administration, V.S.; funding acquisition, V.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the MHARA (Motivating Healthy and Active Aging through Robotic Companionship) project funded by European Union—Next Generation EU, PNRR MUR M4C2 Iniziativa 1.3—Partenariati estesi (PE) CUP E63C22002050006 and MUSIC4D (Musica, Imprenditorialità, Creatività e Rivoluzione Digitale: il Futuro delle Arti Performative nel sistema AFAM) project funded by European Union—Next Generation EU PNRR MUR M4C1 Investimento 3.4 CUP J77G23000170006.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors would like to thank the members of the Robotics Laboratory and the Software Engineering Laboratory of the Department of Engineering for their valuable support during the development of this work. In particular, they are gratefully acknowledged for the discussions that contributed to refining the ideas underlying the proposed approach, as well as for the technical support provided during the implementation and validation activities. During the preparation of this manuscript, the authors used automatic language assistance tools to improve the fluency, clarity, and grammatical correctness of the English text. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflict of interest. All authors have read and agreed to the published version of the manuscript.

References

1. Tenorth, M.; Beetz, M. Knowledge processing for autonomous robots. *Artif. Intell.* **2011**, *173*, 415–445.
2. Beetz, M.; Beßler, D.; Haidu, A.; Pomarlan, M.; Bozcuoğlu, A.K.; Bartels, G. Know rob 2.0—A 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 512–519.
3. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
4. Guarino, N.; Oberle, D.; Staab, S. What is an ontology? In *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–17.
5. Staab, S.; Studer, R. *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2009.
6. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Knowledge Systems Laboratory Stanford University: Stanford, CA, USA, 2001; pp. 1–28.
7. Flouris, G.; Manakanatas, D.; Kondylakis, H.; Plexousakis, D.; Antoniou, G. Ontology change: Classification and survey. *Knowl. Eng. Rev.* **2008**, *23*, 117–152. [CrossRef]
8. Khattak, A.M.; Latif, K.; Lee, S.; Lee, Y.-K. Ontology Evolution: A Survey and Future Challenges. In *International Conference on U-and E-Service, Science and Technology*; Ślęzak, D., Kim, T.-H., Fang, W.-C., Meersman, R., Shiratori, N., Eds.; Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 62, pp. 68–75.

9. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 494–514. [CrossRef] [PubMed]
10. Trivedi, R.; Dai, H.; Wang, Y.; Song, L. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017; 2022; Volume 70, pp. 3462–3471.
11. Pan, S.; Razniewski, S.; Kalo, J.C.; Singhania, S.; Chen, J.; Dietze, S.; Jabeen, H.; Omeliyanenko, J.; Zhang, W.; Lissandrini, M.; et al. Large Language Models and Knowledge Graphs: Opportunities and Challenges. *arXiv* **2023**, arXiv:2308.06374. [CrossRef]
12. Wang, X.H.; Zhang, D.Q.; Gu, T.; Pung, H.K. Ontology based context modeling and reasoning using OWL. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second, Orlando, FL, USA, 14–17 March 2004*; IEEE: Piscataway, NJ, USA, 2004; pp. 18–22.
13. Antoniou, G.; van Harmelen, F. *A Semantic Web Primer*; MIT Press: Cambridge, MA, USA, 2004.
14. Available online: https://www.w3.org/TR/owl2-primer/ (accessed on 1 March 2026).
15. Available online: https://www.w3.org/TR/2009/WD-owl2-primer-20090421/ (accessed on 1 March 2026).
16. Baader, F. (Ed.) *The Description Logic Handbook*; Cambridge University Press: Cambridge, UK, 2003.
17. Garcez, A.d.A.; Gori, M.; Lamb, L.C.; Serafini, L.; Spranger, M.; Tran, S.N. Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning. *arXiv* **2019**, arXiv:1905.06088. [CrossRef]
18. Besold, T.R.; Bader, S.; Bowman, H.; Domingos, P.; Hitzler, P.; Kühnberger, K.U.; Lamb, L.C.; Lima, P.M.V.; de Penning, L.; Pinkas, G.; et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv* **2017**, arXiv:1711.03902. [CrossRef]
19. Hitzler, P.; Sarker, M.K. *Neuro-Symbolic Artificial Intelligence: The State of the Art*; SAGE Publications Ltd.: London, UK, 2022.
20. Stojanovic, L. Methods and Tools for Ontology Evolution. Ph.D. Thesis, University of Karlsruhe, Karlsruhe, Germany, 2004.
21. Hartmann, J.; Palma, R.; Gómez-Pérez, A. Ontology Repositories. In *Handbook on Ontologies*; Staab, S., Studer, R., Eds.; International Handbooks on Information Systems; Springer: Berlin/Heidelberg, Germany, 2009; pp. 551–571.
22. Lemaignan, S.; Warnier, M.; Sisbot, E.A.; Clodic, A.; Alami, R. Artificial cognition for social human–robot interaction. *Artif. Intell.* **2017**, *247*, 45–69. [CrossRef]
23. Tenorth, M.; Beetz, M. KnowRob—A knowledge processing infrastructure for cognition-enabled robots. *Int. J. Robot. Res.* **2013**, *32*, 566–590. [CrossRef]
24. Jabla, R.; Khemaja, M.; Buendia, F.; Faiz, S. Automatic ontology-based model evolution for learning changes in dynamic environments. *Appl. Sci.* **2021**, *11*, 10770. [CrossRef]
25. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* **2024**, *15*, 1–45. [CrossRef]
26. Kautz, H. The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Mag.* **2022**, *43*, 105–125.
27. Tang, X.; Zheng, Z.; Li, J.; Meng, F.; Zhu, S.-C.; Liang, Y.; Zhang, M. Large Language Models are In-Context Semantic Reasoners rather than Symbolic Reasoners. *arXiv* **2023**, arXiv:2305.14825. [CrossRef]
28. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
29. Tapus, A.; Mataric, M.J.; Scassellati, B. Socially assistive robotics [grand challenges of robotics]. *IEEE Robot. Autom. Mag.* **2007**, *14*, 35–42. [CrossRef]
30. Pollack, M.E. Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI Mag.* **2005**, *26*, 9.
31. Kyrarini, M.; Lygerakis, F.; Rajavenkatanarayanan, A.; Sevastopoulos, C.; Nambiappan, H.R.; Chaitanya, K.K.; Babu, A.R.; Mathew, J.; Makedon, F. A survey of robots in healthcare. *Technologies* **2021**, *9*, 8. [CrossRef]
32. Dautenhahn, K. Socially intelligent robots: Dimensions of human–robot interaction. *Philos. Trans. R. Soc. B Biol. Sci.* **2007**, *362*, 679–704. [CrossRef] [PubMed]
33. Fong, T.; Nourbakhsh, I.; Dautenhahn, K. A survey of socially interactive robots. *Robot. Auton. Syst.* **2003**, *42*, 143–166. [CrossRef]
34. Galindo, C.; Saffiotti, A.; Coradeschi, S.; Buschka, P.; Fernandez-Madrigal, J.A.; González, J. Multi-hierarchical semantic maps for mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005*; IEEE: Piscataway, NJ, USA, 2005; pp. 2278–2283.
35. Lopez, B.; Meseguer, P.; Plaza, E. Knowledge based systems validation: A state of the art. *AI Commun.* **1990**, *3*, 58–72. [CrossRef]