UDC 004.89:004.94:004.04. **doi: 10.32620/reks.2026.2.05**

Oleksii ATAMANIUK, Yevgeniya SULEMA

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnical Institute", Kyiv, Ukraine

TEMPORAL EVENT-DRIVEN SCHEDULED ACTIONS (TESA) ONTOLOGY FOR CYBER PHYSICAL SYSTEMS WITH PRIORITY MECHANISM

*The **article discusses** the development and evaluation of lightweight ontologies with scheduling and conflict-resolution mechanisms for modeling tightly coupled, complex systems in cyber-physical systems**,** such as Digital Twins. The **goal** is to develop an OWL-Time extension with scheduling, event-driven actions, and conflict resolution, and to confirm its efficiency by four measurable criteria: Halstead effort reduction versus the ISO-standard PSL; near-linear reasoning latency for 5–50 conflict groups; no reasoner halts; and consistency with foundational ontologies such as BFO, SUMO, UFO, and DOLCE. The **tasks** to be solved are: to collect requirements for the new ontology; to describe core elements of the ontology-modeled system ensuring compatibility with foundational ontologies; to develop a priority mechanism for resolving conflicting actions; to describe and evaluate the new ontology with particular focus on limiting halts caused by conflicting actions during ontology reasoner execution. The **methods** used are: test system design for experimental validation; formalization of the system for modeling using the proposed and existing ontologies; software engineering techniques for running experiments and collecting performance data; Halstead effort metrics for measuring cognitive and developmental effort. The following **results** were obtained: the proposed TESA ontology achieves a 70% reduction in the Halstead effort metric compared to the ISO-standard PSL in the considered test scenarios, indicating a significant decrease in cognitive and developmental effort; TESA prevents the reasoner from encountering halts during reasoning operations; under high-density conflict scenarios, TESA exhibits superior temporal stability with sub-second reasoning latency, while traditional methodologies undergo exponential performance degradation; the ontology demonstrates a lightweight architecture and practical compatibility with foundational ontologies; TESA's modular structure and reduced semantic overhead enable streamlined adoption and integration. **Conclusions**. The scientific novelty of the results obtained is as follows: in contrast to the existing scheduling ontologies, a lightweight ontology for event scheduling and conflict prevention has been developed that strikes a balance between detailed semantics and ease of integration with foundational ontologies; the modular structure enables specialized domain adaptations while maintaining compatibility with higher-level foundational ontologies; the priority-based conflict resolution mechanism prevents reasoner halts and maintains stable performance under high-conflict scenarios; the ontology architecture reduces cognitive overhead while preserving robust scheduling logic capabilities.*

Keywords*: scheduling event ontology; conflict resolution; event-driven ontology; cyber-physical system; digital twin; software engineering.*

1. Introduction

1.1. Motivation

Owing to the rapid development of information technologies, cyber-physical systems are evolving rapidly. In recent years, typical programmed systems have transitioned from simple models to complex intelligent software capable of analyzing large amounts of data and making decisions. One of the most popular and effective technologies for virtualizing real objects is the Digital Twin technology [1-2]. Digital twins offer advanced features due to the connection between physical and virtual environments, which allows the faster tracking of any changes and their reaction. The modeled system should be analyzed in depth to enable such complex abilities, including the collection of various usage scenarios, edge cases, and failure-handling situations. Real systems in the modern world consist of multiple separate parts that are connected to each other and can have different types of relations. Further, such systems are referred to as "complex tightly-coupled systems". Such complexity of the examined systems can complicate the development process in the early phases due to the ever-increasing amount of data that should be processed and organized to formulate clear requirements and to clearly describe the investigated cases. To simplify the design of the digital twin of a complex tightly coupled system [3], developing an ontology of the system is a common practice [4]. The further use of the considered system ontology simplifies the implementation due to easier tracking of relationships between elements, modeling of their communication, analysis of their data, prediction of the results of actions, and decision-making. One of the most critical tasks in designing an ontology for a complex tightly-coupled system is to consider the cases when relations can change, when the result of performing an action depends on the time when it was done, as well as event scheduling and the handling of numerous events. Digital twins, automated workflows, and sensor-driven processes produce persistent and frequently time-critical data streams in current industrial ecosystems. Controlling corrective actions, such as shutting down a machine when a sensor indicates overheating, relies not only on the device hardware but also on a higher con-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

ceptual layer. Omitting time-based constructs in an ontology can lead to partially automated solutions that rely on manual input and external scripts, ultimately increasing complexity and limiting scalability.

The advancement of Industry 4.0 and the widespread implementation of smart factories highlight the strong need for integrating semantic precision with real-time adaptability. In this situation, creating and validating a high-level ontology that maintains robust high-level semantic structures, incorporates time-related dimensions, and supports event-condition-action logic is critical. Moreover, for further integration with existing systems, it is necessary to consider popular mechanisms for defining the ontology structure and elements, such as RDF and OWL [5].

1.2. State of the art

The development and analysis of ontologies for cyber-physical systems, including Digital Twins, have been demonstrated in numerous scientific papers. Karabulut et al. [6] presented one of the latest research projects in the investigated area. This study presents a systematic literature review of the ontologies in Digital Twins and highlights the variety of solutions created. Notably, many of the developed ontologies are "domain-specific" or consider only specific tasks or processes. Despite the importance of backward compatibility between ontologies, the analysis of more than 80 ontologies shows that more than half of the considered articles did not match their proposed ontology with a top-level ontology. Such issues with matching ontologies of different levels create re-usability problems and hinder their further integration into existing or newly developed digital twins.

A top-level ontology should be used when designing the digital twin for the complex tightly coupled system to prevent the described re-usability issues [7]. Ermolayev et al. [8] investigated the question of time ontologies and described the trends and approaches used to model time-based systems. This paper includes several high-level ontologies that can be used for the cyber-physical system. Let us examine the foundational ontologies for the considered type of systems, considering time-dependent cases.

OWL-Time [9] is a widely recognized and widely adopted model for representing temporal information within the Semantic Web. It focuses on describing core temporal elements, i.e., instants, intervals, and durations, by leveraging standard XML Schema datatypes for dates and times. This approach fits seamlessly within the RDF and OWL technology framework, enabling developers to harness a single, efficient vocabulary to cover all aspects of time representation. The close integration with XSD types facilitates comparisons and indexing, enhancing compatibility across various semantic technologies. OWL-Time has become the go-to module for domain ontologies that require labeling facts with specific instants and intervals because it serves as a key descriptive framework for time. Many researchers and practitioners who work with overarching ontologies, such as SUMO, BFO, or DOLCE [10], frequently incorporate or reference OWL-Time to position events or processes on a timeline. By defining instants as zero-duration points and representing intervals with clearly defined (and sometimes open-ended) boundaries, OWL-Time offers a reliable way to specify when an event starts or finishes and how long it endures. The ontology also has optional structures for breaking down date and time, accommodating months, days, hours, and similar subdivisions when more detail is needed. Although OWL-Time excels at detailing when events occur, it does not include built-in constructs for procedural or operational logic. In situations that require event-condition-action rules, conflict resolution, or sophisticated scheduling, implementations often depend on OWL-Time's external rule engines or custom extensions. Common methods involve drafting rules in languages such as SWRL or embedding logic [11] within custom code to interpret data that references OWL-Time. Such rules can correlate intervals and instants to specific triggers, such as "execute an action if the sensor reading exceeds a threshold at time T." However, these rules are developed outside the OWL-Time framework. Therefore, OWL-Time remains fundamentally descriptive, allowing for consistent temporal annotation while entrusting higher-level scheduling or real-time procedures to additional modules or external logic engines.

DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [12] emphasizes cognitive and conceptual distinctions. The ontology separates endurance (objects that exist wholly at each moment in time) from perdurants (events or processes that unfold in time). This approach makes DOLCE well suited for high-level semantic modeling and has allowed researchers to build digital twins that capture physical or organizational systems' structural and functional facets. However, the application of DOLCE to scenarios of tightly coupled systems that require immediate synchronization between digital twins and real-world data is complicated. DOLCE's default method for handling time often relies on temporal slices, a technique that adds complexity when processing frequent or large-scale updates is required. While it supports temporal considerations, the emphasis remains on static conceptual clarity, and further modules or rule-based scripts are typically required to cover the dynamic behaviors of complex systems.

BFO (Basic Formal Ontology) [13] is commonly found in biomedical and life science contexts. The ontology is grounded in a philosophical division between

continuants (entities persisting through time) and occurrents (processes happening over time). Although BFO provides a systematic framework for describing biological and industrial processes, its native handling of time-centric relations is not direct. BFO generally involves labeling an entity as either a process or a participant in a process, rather than offering a native method to encode rules or triggers within the ontology. While BFO's rigor appeals to those designing digital twins for complex systems, it often needs specialized extensions or external logic to manage rapid reconfigurations and scheduling tasks in closely knit manufacturing processes. The ontology core model handles time as part of the occurrent dimension, but it does not inherently address time ranges for fine-grained rule enforcement in real-time.

The SUMO (Suggested Upper Merged Ontology) [14-15] has been proposed as a large-scale ontology that encompasses physical, abstract, social, and psychological concepts. The ontology typically includes basic representations for time, causes, and processes through statements such as before and after, along with various types of relationships, such as subProcessOf. This coverage offers the potential for building robust digital twins, especially in domains that benefit from a unified view of physical devices, organizational structures, and knowledge artifacts. However, the ability of SUMO to handle complex or tightly coupled production systems in real time often relies on external logic encodings. As a result, developers frequently need to write elaborate custom scripts or reasoner extensions to build digital twins that must adapt to continuous data streams.

The UFO (Unified Foundational Ontology) [16] stands out for its emphasis on roles, social contexts, and event modeling. The ontology supports the notion of events (UFO-B) and the interplay of actors or agents (UFO-C), making it appealing to complex enterprise systems. Developers have leveraged UFO to create digital twins that combine organizational elements, machine states, and event-driven transitions. However, while UFO's event concept provides a way to represent processes, it does not inherently offer direct methods for scheduling or rule-based triggers if time must be handled as a primary driver for changes. Similar to other foundational ontologies, time ranges, scheduling logic, or conflict resolution for rule-based transitions usually require custom extensions or external engines, rather than operating natively within the UFO model.

The ability to represent structural and functional components of digital twins is a marked strength across DOLCE, BFO, SUMO, and UFO, and each one can describe aspects of a system's entities and processes [17-18]. Their applicability to complex tightly coupled or dynamically changing environments is somewhat constrained by the lack of integrated constructs that handle time-ranged relations or direct rule enforcement for event triggers. This issue becomes particularly pronounced in production scenarios where multiple real-time updates, immediate scheduling, or conflict resolution are required. Despite their conceptual depth, these foundational ontologies often require significant additional modules to implement sophisticated time-based rules, highlighting a gap in the development of more comprehensive, time-centric semantic frameworks.

Examining scheduling ontologies to determine how effectively they can address the identified challenges and whether they can be integrated with foundational ontologies without introducing significant complications is an obvious and logical step. Existing scheduling ontologies, such as the Process Specification Language (PSL) [19], the Generic Task Ontology (GTO) [20], and iCalendar standard [21], provide essential frameworks for expressing temporal structures, tasks, and events. The Process Specification Language is an ISO-standard ontology notable for its formalism. The structure of PSL effectively delineates the complexities of manufacturing and business processes in substantial detail, making it particularly suitable for intricate scheduling scenarios. While these ontologies and standards offer considerable value, each comes with limitations that make practical implementation challenging, especially when integrating with foundational or domain-specific ontologies. For example, the Process Specification Language, though robust and detailed, can become overly complex for simpler scheduling scenarios. Aligning it with broader knowledge systems also requires a substantial time and effort investment.

The Generic Task Ontology, meanwhile, provides a flexible structure for defining tasks and subtasks, which supports modular activity breakdowns. However, effectively applying it in real-world environments usually means adding extra layers or extensions to capture scheduling nuances, which reduces its out-of-the-box usefulness.

iCalendar stands out for its accessibility and wide adoption, offering a straightforward format for managing and exchanging event data. Its strength lies in its simplicity and compatibility across platforms. However, this simplicity becomes a drawback when more advanced scheduling features, such as task interdependencies and constraints, are required, as the standard lacks the semantic depth to handle such complexity.

Taken together, these tools demonstrate both the progress and obstacles in formalizing scheduling processes. PSL can overwhelm with its expressive power, iCalendar may fall short in semantic richness, and the Generic Task Ontology often needs tailoring for practical use. These challenges point to a clear need for a more streamlined yet semantically meaningful solution that bridges the gap between expressive capability and ease of integration. The ideal approach would preserve

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

essential semantic structure while remaining lightweight enough to integrate smoothly into foundational ontologies, avoiding the burdens of excessive complexity.

The current state of the art highlights the gap in ontology-based modeling for high-frequency, real-time, and time-dependent cyber-physical applications. It is important to develop a new top-level lightweight ontology that is reusable, incorporates time-based constructs, and simplifies the design of Digital Twins for complex, tightly coupled systems. It is important for such ontologies to be compatible with OWL-Time, which is the de-facto standard for describing time values. Moreover, using the base concepts of the examined upper-level ontologies can simplify development and increase compatibility among them.

1.3. Objectives and tasks

This study aims to develop a high-level ontology that incorporates time-based constructs and event handling into knowledge representation and demonstrate that it measurably outperforms existing approaches on a complex tightly coupled test system. The goal is considered achieved if the following criteria are met:

1) the Halstead effort of describing the test scheduling scenarios is reduced by at least 50% compared with the ISO-standard PSL;

2) the mean reasoning latency stays below one second and grows no worse than linearly for 5 to 50 simultaneous conflict groups;

3) no reasoner halts occur in any of the conflict scenarios executed;

4) The ontology, combined with the foundational ontologies BFO, SUMO, UFO, and DOLCE, remains logically consistent under the Pellet and HermiT reasoners.

The specific objectives are as follows:

1) Collect base requirements for the new ontology, including the supported constructs and logic.

2) Develop a new ontology capable of dynamic adaptation through time-based constructs.

3) Describe a complex, tightly coupled system to be used for testing.

4) Examine how the new ontology can be used for modeling and how its constructs manage updates over specified time ranges using the described test system.

5) Define test scenarios that will be evaluated to compare the ontologies' performance and capabilities.

6) Conduct an analysis of the new ontology and the four foundational ontologies using the developed complex system.

7) Examine the results of the analysis and interpret the findings in terms of the efficiency and applicability of the ontology.

The test system will include at least five semantic levels of components and at least 20 interrelations, ensuring that the relations span multiple levels. The parameters of each entity will be governed by time-based logic that reflects the scheduled events or real-time triggers. To develop and formalize the ontologies the Resource Description Framework (RDF) will be applied. The Python programming language will then be used to programmatically implement the modeled systems, allowing for both systematic testing and the demonstration of rule execution within the newly proposed and comparative foundational ontologies.

This paper has the following structure:

Section 2 describes the requirements for the new ontology and its detailed description.

Section 3 describes the complex, tightly coupled system to be used for testing and its implementation with the created ontology, as well as the test scenarios and analysis of the new ontology.

Section 4 discusses the findings, ontology limitations, and potential applications in real-world scenarios.

Section 5 summarizes the key contributions and outlines potential directions for future research.

2. Development of a new ontology

2.1. Requirements of the new ontology

First, to develop the new ontology, a list of clear requirements must be collected. Foremost, this ontology must be high-level so that it can be reused for building lower-level or more domain-specific ontologies. A high-level design ensures that its fundamental constructs are not overly tied to any single industry, allowing designers to easily adapt them for a range of systems. However, ensuring broad reusability also implies that the ontology must have native constructs that enable the definition and management of various user-defined entities. Such a structure will allow system architects to introduce new classes, relationships, and their attributes while designing specialized digital models.

The core feature of the ontology is native support for time-based logic. In practice, this means that the ontology must accommodate the dynamic creation and management of entity attributes and inter-entity relationships that change in accordance with a schedule or an event trigger. Instead of merely describing time, the ontology must natively accommodate scheduling triggers and dynamic actions, allowing entities or their relationships to change according to recurring intervals or real-time thresholds. Extending OWL-Time with such operational logic helps address real-world scenarios, such as daily maintenance tasks or sensor-driven responses.

The ontology must also resolve the issue of contradictory directives that can arise when multiple triggers overlap. For instance, a production line might be required to run continuously under one rule but forced to stop under another. The ontology must assign a nu-

meric or ordinal priority to each rule to prevent the entire reasoner from halting due to inconsistencies, clarifying which trigger takes precedence. This ensures the system continues to function even under partial constraint collisions by automatically overriding lower-priority directives.

Moreover, the ontology design should be compact and modular. It needs only a few targeted constructs, such as classes, to represent triggers, periodic intervals, and a conflict-resolution mechanism. Such an approach allows existing high-level ontologies, such as BFO, UFO, or SUMO, that already rely on OWL-Time to adopt these scheduling capabilities with minimal changes. System integrators would continue to describe basic intervals using OWL-Time while selectively adding new constructs for daily or monthly recurrences and rule priority.

Attention to alignment with standard semantic frameworks, such as RDF and OWL, is essential to ensure that reasoners and editors can handle the new classes without discarding existing annotations. The ontology should also scale well for environments containing many entities or rules. By addressing the following requirements: reusability, extensibility, interoperability, and conflict management, the ontology can bridge the gap between static time modeling and real operational needs in diverse, fast-changing production settings.

2.2. Temporal Event-driven Scheduled Actions (TESA) ontology

The core structures of the base ontologies should be defined and described. In this section, we explain how triggers, conditions, priorities, and extended intervals fit into a higher-level framework compatible with standard OWL-Time definitions. The goal of this study is to retain the well-established notions of time instants and intervals found in OWL-Time and then to embed additional concepts for rule-based scheduling and conflict management. By clarifying the role of each component within the wider OWL model, we ensure that the new features of the ontology are easy to integrate into existing foundational ontologies [22-24]. Furthermore, the constructs are defined in a way that makes the development of a cyber-physical system easier due to the set-based description, and they do not rely on specific features of any foundational ontology.

Let the system under study have n types of entities E (concepts) that describe its objects:

$$E = \{E_1, E_1, ...E_n\}. \quad (1)$$

Each entity E_i is characterized by a name, which serves as a unique identifier for the type, and a set of attributes $A(E_i)$ that describe it:

$$E_i = (\text{name}, A(E_i)) \quad (2)$$

Since the name acts as a unique identifier for the concept, in cases where a specific entity is considered, it is more practical to refer to it by its name E_{name} rather than using an arbitrary index, E_i. Entities in both OWL ontologies and programmed systems correspond to "classes". As the ontology will primarily be used for cyber-physical systems such as Digital Twins, it is advisable to include additional attributes, created_at and updated_at, to facilitate change tracking and enable more efficient monitoring of entity behavior over time. A distinct entity should be introduced to represent the system Eenv's global environment. The inclusion of such an entity significantly simplifies the construction of global interconnections among entities and the calculation of global system characteristics. Instances of a particular entity type will be denoted accordingly $O_k^{E_i}$.

Accordingly, the system comprises a complete set of attributes A, which can be applied to describe any entity from the given entity set E:

$$A = \{A_1, A_2, ..., A_n\}, \quad (3)$$

$$A(E_i) = \{A_{i1}, A_{i2}, ..., A_{ik}\}, A(E_i) \subseteq A. \quad (4)$$

Each attribute is defined by a name, serving as its identifier, and a unit of measurement $U(A_j)$:

$$A_j = (\text{name}, U(A_j)). \quad (5)$$

Specifying the unit of measurement for an attribute is crucial for simplifying heterogeneous data processing from various sources. Therefore, a unified set of measurement units U is employed for system description as follows:

$$U = \{U_1, U_2, ..., U_n\}. \quad (6)$$

Measurement units can only be assigned using a unique textual name. In specific cases where complex types (e.g., enumerations) must be defined, an element U may contain additional fields. Units can be classified as the types of values in the cyber system and OWL ontologies. Notably, the set of attributes thus forms the set of modalities for the model of the system. For example, the set of attributes defines the modalities of the Muxel model used for Digital Twin data representation [25].

Let us proceed with the constructs for the values. At a given point in time t_n, an attribute A_j with a specific name assigned to an object $O_k^{E_i}$ takes on a particular value $V_{Name(A_j)}$. For example, at a certain moment, the temperature in a refrigerator might register a value of 30 °C: $V_{temp} = 30$. Consequently, for a given object, an attribute value is represented as a tuple consisting of timestamp, attribute name, absolute value, and metadata fields:

$$V = (O_k^{E_i}, t_n, \text{Name}(A_j), \text{value}, \text{metadata}). \quad (7)$$

Metadata can encompass various values depending on the system characteristics and complexity of the on-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

tology. The recommended metadata fields for a value include the following:

- created_at – timestamp of record creation;
- updated_at – timestamp of the last update;
- rule_id – reference to the rule used for value computation;
- source_data_row – original data or a reference to it.

The creation and update timestamps facilitate better tracking of system behavior, whereas references to computation rules and original data help trace the derived values.

Within the system, entities establish relationships, where each relationship type belongs to the set of relationship types R:

$$R = \{R_1, R_2, ..., R_k\}. \quad (8)$$

Relationships are defined by a unique textual name that serves as their identifier. The list of relationship types depends on the system or base ontology. The most common types of relationships can be classified as follows.

Hierarchical Relationships (Structural Relationships) are used to describe connections that express the structure or hierarchy between entities:

- belongs-to– Indicates that one entity is an internal part of another. Example: "Engine belongs to Car" (The engine is an internal part of the car).
- subclass-of – Defines a subclass relationship with a superclass. Example: "Car subclass-of Vehicle" (A car is a vehicle subclass).
- instance-of – Specifies that a particular entity is an instance of a given class. Example: "Honda Civic instance-of Car" (Honda Civic is an instance of a car).

Associative Relationships (Relational Relationships) are used to describe associations between entities without establishing a hierarchy:

- has – Indicates the possession of a property or component. Example: "Car has Engine" (A car has an engine).
- uses – Specifies that one entity utilizes another. Example: "Car uses Fuel" (A car uses fuel).
- depends-on – Denotes dependency between entities. Example: "Car depends-on Engine" (A car depends on an engine).

Logical Relationships are used to define connections based on the following logical conditions:

- equivalence – Specifies that two entities are equivalent. Example: "X is equivalent to Y" (X is equivalent to Y).
- implication – Indicates that one entity logically implies another. Example: "If X, then Y" (If X occurs, then Y follows).
- contradiction – Specifies that two entities cannot coexist. Example: "X contradicts Y" (X contradicts Y).

Role-Based Relationships are used to define connections between entities based on their roles or functions in a given context:

- role-of – Indicates that an entity plays a specific role in relation to another entity. Example: "John role-of Manager" (John plays the role of a manager).
- responsible-for – Specifies responsibility for a particular action or event. Example: "Manager responsible-for Project" (The manager is responsible for the project).

Temporal Relationships are used to describe changes or events over time. They are based on the relationships between distinct intervals defined in the Algebraic System of Aggregates (ASA) to provide accuracy and flexibility in the description of events [26]. Unlike Allen's interval-based theory [27] employed in OWL's time ontology [9], ASA defines 14 relationships that include: Coincides With; Is Before; Is After; Meets; Is Met By; Overlaps; Is Overlapped By; During; Contains; Starts; Is Started By; Finishes; Is Finished By, and Between. For example:

- Is Before specifies that one event occurs before another. Example: "Event A before Event B" (Event A occurs before Event B).
- Is After specifies that one event occurs after the other. Example: "Event A after Event B" (Event A occurs after Event B).
- During indicates that an event occurs within the timeframe of another event. Example: "Event A during Event B" (Event A happens during Event B).
- Between means that one or several events occur between two specified time-points (the relationship Between can be applied to any number of discrete intervals, each of which corresponds to a time period of a certain event). Example: "Event A and Event B between moment X and moment Y" (Event A and Event B happen between moment X and moment Y).

Spatial Relationships describe the connections between entities. These relationships comply with the W3C recommendations [28] and include four categories: topological, mereological, directional, and distance-related. Topological relationships are as follows: Disconnected From, Part Of, Proper Part Of, Equal To, Overlaps With, Discrete From, Partially Overlaps With, Externally Connected To, Non-Tangential Part Of, Tangential Proper Part Of, Non-Tangential Proper Part Of. They correspond to the topological relations defined in the Region Connection Calculus [29]. Mereological relationships are defined as part-whole. Directional relationships are as follows: Left, Right, Up, Down, North, South, East, West, North-East, North-West, South-East, South-West. Distance-related relationships are as follows: Near, Far, Within a Radius Of, By Distance Of. Some examples of spatial relationships are as follows:

- near – Indicates that two entities are located close to each other. Example: "Point A near Point B" (Point A is near Point B).
- in – Specifies that one entity is contained within another. Example: "Object A in Container B" (Object A is inside Container B).

These classifications provide a structured framework for defining relationships within a system and facilitating efficient modeling and analysis of complex entity interactions.

A set of relationships Rel between entities is established in the process of researching the ontology of a tightly coupled system. Each relationship is defined as a triple consisting of the following elements: subject, predicate, object:

$$Rel_{E_i}^{E_j} = \{(E_i, R_k, E_j) \mid E_i, E_j \in E, R_k \in R\}, \quad (9)$$

where an entity E_i serves as the object of the relationship, the predicate R_k represents the type of relationship, and the subject E_j defines the initiating entity. Notably, some relationships may only be added for specific instances of entities; therefore, both objects and subjects of relationships can be system objects (instances of the entities) $O_k^{E_i}$. Such relationship triples are widely used in OWL and RDF, which are the most common standards for ontology description.

Despite the extensive capabilities of defining relationships, covering all necessary scenarios and specific conditions is not always possible. Therefore, for causal dependencies with complex conditions, introducing a set of Rules is reasonable. These rules can define relationships involving more than two entities or objects. For instance, if object A of entity E_1 has a temperature value T_1, and component B of entity E_2 has a temperature value T_2, then the temperature of component C of entity E_3 can be determined as $T_1 + T_2 - 10$. Such rules are developed using predicates.

Let the system contain a set of objects O:

$$O = \{O_1, O_2, ..., O_n\}, \quad (10)$$

a set of predicates P:

$$P = \{P_1, P_2, ..., P_k\}, \quad (11)$$

and a set of consequences Q are as follows:

$$Q = \{Q_1, Q_2, ..., Q_m\}. \quad (12)$$

A rule is then defined as follows:

$$(\forall i \in \{1,2,...,n\}, \forall j \in \{1,2,...,k\}, P_j(O_i)) \Rightarrow \left(\bigwedge_{l=1}^{m} Q_l(O_{res}) \right) \quad (13)$$

This can be interpreted as: "If, for each i-th object, the corresponding j-th predicate holds true, then the object O_{res} acquires the l-th consequences." Accordingly, for each rule, a subset of objects is specified, where all predicates must hold for a particular object to acquire the defined consequences. Predicates P can be described using the attribute values of an object belonging to a specific entity:

$$P = (E_i, Name(A_j), V), A_j \in A(E_i). \quad (14)$$

It is also necessary to consider temporal constraints. Thus, introducing a verification mechanism that ensures that a predicate holds within a given time range $[t_{min}; t_{max}]$

Additionally, the definition and impact of the set of consequences within the complex, tightly coupled system must be examined. In general, each consequence determines how an object's attribute values will change. An attribute change can be defined as a constant value assignment (e.g., updating an object's status). Alternatively, attribute updates may be computed using mathematical formulas, such as determining an object's temperature as the arithmetic mean of the other objects' temperatures.

To describe a consequence, the following functions are used:

- $V_{A_j}(O_i)$ – a function that retrieves or sets the value of attribute A_i for object O_j;
- $f(V_{A_1}(O_1), V_{A_2}(O_2), ..., V_{A_j}(O_i))$ – a function that computes a value based on attribute A_i of object O_j from a given set of objects.

Thus, a consequence is formally represented as a tuple consisting of:

- O_{res} - an entity or object for which the attribute value is modified.
- $Name(A_{res})$ - a name of the attribute being updated.
- $f_{A_{res}}$ - the attribute's future value.

$$Q = (O_{res}, Name(A_{res}), f_{A_{res}}), A_{res} \in A(O_{res}) \quad (15)$$

Such rules are usually described via Web Rule Language (SWRL) extension, which is compatible with most reasoners. Since the model is developed for digital twin applications, time-based attribute modifications must be considered. Therefore, it is appropriate to introduce a function that determines the time f_{time} at which the consequence is applied:

$$Q = (O_{res}, Name(A_{res}), f_{A_{res}}, f_{time}), A_{res} \in A(O_{res}). \quad (16)$$

This approach enables accurate simulation, tracking, and prediction of object behavior within the digital twin system by incorporating temporal logic.

Since the ontological model is being developed for digital twins, defining distinct entities dedicated to event-driven processing for data acquisition and processing is advisable.

The "Data Source" entity represents data sources and their locations. These data sources may include sensors, external services, and other input mechanisms.

The "Raw Data" entity is responsible for storing and describing the original data obtained from a source. Preserving these data in their initial form, along with the timestamp of acquisition, a reference to the source, and relevant metadata, is crucial to ensure accurate processing and correct transformation of heterogeneous data.

Furthermore, to implement an event-driven approach for the digital twin's operation, it is reasonable to introduce the "Event" entity. This entity describes the time and conditions under which data were acquired or modified for a specific object. The creation of such an entity's objects will trigger subscribed event handlers for further processing.

The structure and attributes of the "Event" entity depend on the developers' implementation choices and the selected message line framework. To ensure the correct sequencing of event processing, it is recom-

mended that messages be ordered chronologically based on their creation timestamps.

The core constructs of the new ontology should be defined and explained to proceed. The first and most important construct is the TimeTrigger, which represents a rule or directive that should take effect when one or more time-related conditions are satisfied. In many practical settings (e.g., daily maintenance intervals), system administrators want to declare "If it is this day, then perform the specified action." The TimeTrigger class has the following attributes: the timing aspect – generally a TimeInterval or an instant from OWL-Time – the logic needed to decide when the trigger fires, and the event or action that should be done – in most cases, it will be a reference to an instance of the Action class. The construct also includes a time range that indicates its validity, a description, and metadata. Event conditions that include more than a time range can be described with SWRL rules, as they natively provide numerous value comparison variants.

The next construct is the Action, which codifies the system behavior that should occur once the condition is true. A separate set of Actions can be defined as follows:

$$\mathrm{Act} = \{\mathrm{Act}_1, \mathrm{Act}_2, ..., \mathrm{Act}_n\}, \mathrm{Act} \in \mathrm{O}. \quad (17)$$

Actions may be as simple as "log the event," "boost throughput," or "shut down the line". Thus, an action can represent a state change of an instance (object) or a simple function to be executed by the programmed system layer. This construct contains several attributes: name, global priority indicator (prior), execute indicator (exec), time range indicating its validity (lifetime), and various metadata, including a description, that can be used during execution.

$$\mathrm{Act}_n = (\mathrm{name}, \mathrm{prior}, \mathrm{exec}, \mathrm{lifetime}, \mathrm{metadata}) \quad (18)$$

By natively representing these actions in the ontology, system architects can specify them without using external bridging scripts; the semantic model directly describes the relationships between triggers, conditions, and actions. Although having triggers, conditions, and actions are essential for scheduling and ECA logic, conflicting directives are often encountered in real-world implementations. Here, the PriorityRelator construct comes in. When multiple triggers are applied simultaneously, the ontology can compare the numeric or ordinal priority values attached to each trigger, avoiding global inconsistencies that would normally halt a reasoner. A higher-priority rule effectively overrides lower-priority ones, allowing the system to continue operating even in the presence of contradictory statements. A PriorityRelator instance contains a link to the action Act, a priority value, metadata, and a lifetime period to support the ability to assign different priority values for different time intervals:

$$\mathrm{Pr}_k = (\mathrm{Act}_n, \mathrm{value}, \mathrm{lifetime}, \mathrm{metadata}). \quad (19)$$

A new construct called ConflictedActions is introduced to mark that several actions conflict when applied simultaneously..

$$\mathrm{C} = \{\mathrm{C}_1, \mathrm{C}_2, ..., \mathrm{C}_m\}, \mathrm{C} \subseteq \mathrm{Act}, \left|\mathrm{C}\right| \geq 2. \quad (20)$$

An instance of this construct contains a description, a time range that indicates its validity, and other necessary metadata. A ConflictedActions individual defines a conflict group $C \subseteq Act$, $|C| \geq 2$, whose members cannot be executed simultaneously. The activity of an action at time moment t is defined by the Boolean predicate as follows:

$$\mathrm{active}(\mathrm{Act}, t) = \mathrm{exec}(\mathrm{Act}) \wedge (t \in \mathrm{lifetime}(\mathrm{Act})), \quad (21)$$

where the value exec(Act) = true is set exclusively by a fired TimeTrigger rule during the reasoning phase. Thus, an action can be active only within its validity interval and only after its trigger has been fired.

The arbitration function selects the single active action to be executed for every conflict group:

$$\mathrm{choose}(\mathrm{C}, t) = \arg\max\{\mathrm{prior}(\mathrm{Act}) \mid \mathrm{Act} \in \mathrm{C}\}, \quad (22)$$

where each considered action is active, i.e., active(Act, t) = true. A deterministic tie-breaking rule is applied if several active actions share the maximal priority value: the action with the lexicographically smallest IRI (or, alternatively, the earliest created_at timestamp) is selected. This guarantees that choose(C, t) is a total, single-valued, and deterministic function; thus, the arbitration result is reproducible for any state of the knowledge base.

The key property of this semantics is that the reasoning process is split into two phases. In the first phase, the reasoner monotonically applies the SWRL rules and marks only the candidate actions by setting exec(Act) := true. Conflicting directives are represented as assertions grouped by ConflictedActions individuals, rather than as logically contradictory axioms. Thus, the knowledge base remains consistent in OWL 2 DL, and the reasoner cannot encounter unsatisfiability-related halts by construction. In the second phase, the arbitration function is evaluated outside the description-logic entailment using either an SQWRL aggregation query or the host programming language. The selection operation's computational cost is linear and depends on the number of conflict groups and the number of conflicting actions within each group.

Taken together, these constructs – TimeTrigger, Action, PriorityRelator, and ConflictedActions – allow the TESA Ontology to deliver time-based rules within an otherwise descriptive OWL-Time environment.

Table 1

Special constructs of the TESA ontology

Construct	Description
TimeTrigger	Identifies the time period when action Action should be triggered
Action	Identifies the action that should be executed
PriorityRelator	Identifies the priority of the action
ConflictedActions	Identifies the list of conflicted actions

They provide a concise, reusable vocabulary for embedding conflict-free scheduling and real-time trig-

gers, allowing the handling of recurring tasks, sensor-driven actions, and overlapping directives in a single, integrated framework. To illustrate TESA's position within its ontological context in the cyber-physical system, Fig. 1 presents the relationships between TESA constructs and external ontologies, as well as their connection to the target application.

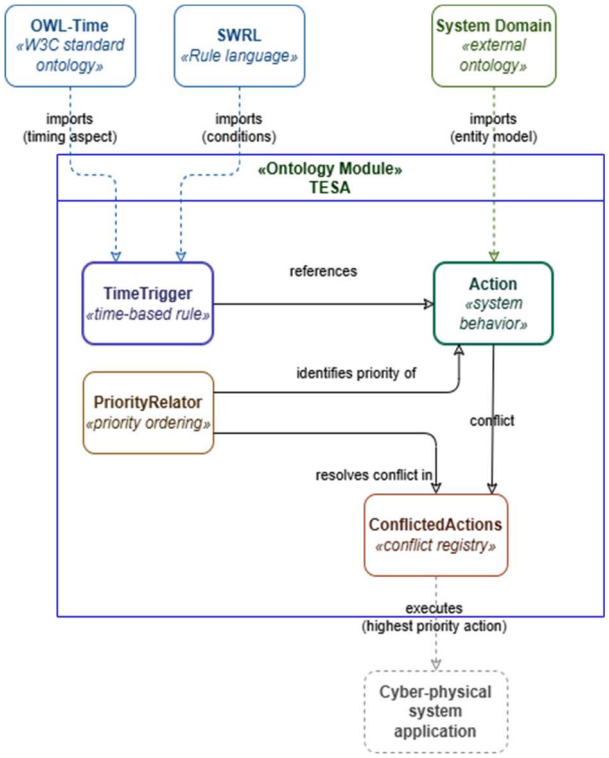

Fig. 1. Relations schema of the TESA Ontology Module and its external dependencies

The TESA ontology module is shown as a bounded unit that imports temporal primitives from OWL-Time, condition logic from SWRL, and entity definitions from the System Domain ontology. The diagram shows how TimeTrigger, Action, PriorityRelator, and ConflictedActions interact to turn the scheduled action events into a single instruction passed to the target system.

3. Analysis of the ontology

The next steps in the research involve testing and analyzing the proposed ontology. First, a detailed description of the test system must be provided. The system should be modeled and implemented using the new ontology. Subsequently, the test cases should be defined and examined. Finally, the analysis of the results should be presented.

3.1. Testing system

Let us begin by describing a complex, tightly coupled system to be tested. The system must include at least five semantic levels of components and at least 20 interrelations to ensure that these relationships span multiple levels.

A car manufacturing facility embodies a complex, tightly coupled system that is ideal for ontology testing and is characterized by a high number of hierarchical levels and inter-level relationships. At the apex of this structure, the Corporate Center oversees the overarching strategy and resource allocation, while the subordinate Divisions delineate specific directives concerning manufacturing, quality assurance, and logistics. Below the Divisional tier, individual Plants are tasked with managing local operations and aligning production objectives across multiple plants. Each Plant is home to several Production Lines, where various car models or components are assembled according to stringent schedules. Within a Production Line, Stations are organized to group related activities such as welding, painting, and quality inspections. These Stations may be further divided into Sub-Stations that focus on more specialized processes, such as laser welding or primer application.

Machinery units, including presses and robotic welders, represent the fundamental production equipment allocated to sub-stations at a more granular level. These units can be segmented into sophisticated robotic arms or specialized mechanical modules that perform repetitive tasks. Microcontrollers are employed to manage real-time operating systems that facilitate communication, error detection, and operational optimization. These Microcontrollers interface with a diverse range of Sensors, including temperature gauges, vibration detectors, and optical scanners, providing continuous feedback for process refinement.

The extensive interrelationships across these hierarchical levels result in a tightly integrated manufacturing environment. The Corporate Center periodically assesses performance indicators from Divisions and may intervene directly with Production Lines when strategic deadlines or model launches necessitate changes in priorities. Divisions coordinate with Plant and Machinery suppliers to establish production targets and define maintenance schedules. Plants leverage real-time data from Sensors, which are aggregated by Microcontrollers, to optimize throughput across Production Lines. Stations frequently share specialized Machinery and rely on various Sub-Stations for component supply or post-production finishing tasks, with certain lines, such as painting, establishing direct interfaces with Divisions tasked with final quality assessments.

Sub-Stations engage in signal exchange to maintain safety compliance, such as suspending upstream processes when Sensors at a downstream Station detect a blockage. Subsequently, the control logic embedded within Microcontrollers prompts robotic arms to modify their operational parameters, such as speed or torque,

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

based on prevailing conditions. Each relationship within this complex system typically incorporates both time-based and event-driven components. For example, the Corporate Center's quarterly revisions to budgetary allocations impact Divisions, which in turn initiate monthly maintenance cycles at each Plant, or Production Lines that respond to sensor alerts by halting certain Stations in real time. This intricate network of relationships facilitates a rapid feedback loop, ensuring that decisions made at the Corporate Center can effectively influence and resonate through the Sensors, thereby providing an optimal environment for examining how ontologies manage cross-level interactions.

3.2. Testing system time-rule based ontology model

The next step in the research is to model the testing system with a new time-rule-based ontology. First, we define the elements of the base ontology sets. To simplify the understanding of the elements' logic, each element will be assigned a unique identifier.

The conceptual types and Entities of the system (Set E): CarFactory, Division, Plant, ProductionLine, Station, SubStation, Machinery, Microcontroller, Sensor, MaintenanceTeam.

List of Objects that are concrete instances of a given entity (Set O):

- CarFactory: FactoryA,
- Division: DivProd,
- Plant: PlantAlpha,
- ProductionLine: LineA1,
- Station: StationWelding,
- SubStation: SubStationLaserWeld,
- Machinery: MachineryRobotWeld,
- Microcontroller: MCtrlWeld,
- Sensor: TempSensorSubWeld,
- MaintenanceTeam: Team4NightShift.

For instance, "Station:StationWelding" is an actual welding station in a factory. An example of the modeled system in the OWL and RDF notations is presented in Fig. 2:

```
<!-- Classes -->
<owl:Class rdf:about="#ProductionLine" />
<owl:Class rdf:about="#Station" />
<owl:Class rdf:about="#StationAssignment" />

<!-- Object property -->
<owl:ObjectProperty rdf:about="#hasStation">
    <rdfs:domain rdf:resource="#ProductionLine" />
    <rdfs:range rdf:resource="#StationAssignment" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#usesStation">
    <rdfs:domain rdf:resource="#StationAssignment" />
    <rdfs:range rdf:resource="#Station" />
</owl:ObjectProperty>

<!-- Individuals -->
<owl:NamedIndividual rdf:about="#Line1">
    <rdf:type rdf:resource="#ProductionLine" />
    <hasStation rdf:resource="#Assignment2022" />
    <hasStation rdf:resource="#Assignment2023" />
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#StationWelding">
    <rdf:type rdf:resource="#Station" />
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#StationAutoWeld">
    <rdf:type rdf:resource="#Station" />
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#Assignment2022">
    <rdf:type rdf:resource="#StationAssignment" />
    <usesStation rdf:resource="#StationWelding" />
    <tesa:validDuringInterval rdf:resource="#Interval2022" />
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#Assignment2023">
    <rdf:type rdf:resource="#StationAssignment" />
    <usesStation rdf:resource="#StationAutoWeld" />
    <tesa:validDuringInterval rdf:resource="#Interval2023" />
</owl:NamedIndividual>
```

Fig. 2. Example of the TESA relations in RDF/XML notation

The list of Attributes, constructs that describe properties (characteristics) that an entity can possess (Set A): operationalState, temperature, throughputRate, scheduleStatus, location, capacity, powerUsage, maintenanceInterval.

The list of Units of Measure (Set U): Celsius, Percentage, PiecesPerHour, Kilowatts, Seconds, Hours. Units are reference measures for numerical attributes (e.g., temperature in Celsius, power usage in Kilowatts).

The time range is omitted in the description of the next elements to avoid overloading the text and to enhance readability. List of Entity Relations (Set R) with examples:

- hasDivision (CarFactory → Division),
- hasPlant (Division → Plant),
- hasProductionLine (Plant → ProductionLine),
- hasStation (ProductionLine → Station),
- hasSubStation (Station → SubStation),
- hasMachinery (SubStation → Machinery),
- hasSensor (Microcontroller → Sensor),
- dependsOn (Station → Station),
- sharesEquipmentWith,
- reportsTo (Plant → Division),
- directsOperation,
- receivesMaintenanceOrder.

The list of Predicates (Set P), that express conditions or logical statements about entities and their attributes (e.g., whether a temperature reading goes above

a threshold):

- tempExceedsThreshold(TempSensor, Threshold),
- isScheduled(MaintenanceTeam),
- hasSufficientPower(Machinery, PowerLevel),
- lineIsOverloaded(ProductionLine, ThroughputRate),
- stationIsBlocked(Station).

The list of Consequences (Set Q), that expresses the results:

- stopLine(ProductionLine),
- reduceThroughput(ProductionLine, Rate),
- triggerMaintenance(Machinery, Team),
- overrideSchedule(Station),
- sendAlertToDivision(Division, Message).

Finally, the most important – Rules set, that combines predicates (P) with consequences (Q) to specify what should happen (e.g., stopping a line, sending an alert) when certain conditions are met. They can also include time triggers or priorities as part of the logic.

- If tempExceedsThreshold(TempSensorSubWeld, 100) then stopLine(LineA1).
- If lineIsOverloaded(LineA1, 200 PiecesPerHour) then reduceThroughput(LineA1, 180 PiecesPerHour).
- If maintenanceInterval(MachineryRobotWeld) expires then triggerMaintenance(MachineryRobotWeld, Team4NightShift)
- If stationIsBlocked(StationPainting) then overrideSchedule(StationPainting) with priority = 100.

To better investigate the new TESA ontology, we provide some examples with relations and rules with time ranges. For example, in 2022, the operational schema of the car manufacturing facility establishes the relation "ProductionLine LineA1 hasStation StationWelding," indicating that LineA1 is designated to the StationWelding for executing welding tasks. Starting in 2023, this relation is revised to "ProductionLine LineA1 hasStation StationAutoWeld," denoting the transition to an enhanced automated welding station.

Additionally, a temporal regulatory framework in place for 2022 stipulates that if the welding station records a temperature exceeding 100°C, the production line's throughput is restricted to a maximum of 200 units per hour. In contrast, from 2023 onwards, the same threshold introduces an elevated capacity of 220 units per hour, thereby enhancing machine efficacy. To implement these rules with the TESA ontology, the set of conditions and consequences must be defined (see Fig. 3).

```
<owl:Class rdf:about="http://example.org/factory#OverheatRulePayload" />

<!-- http://example.org/factory#hasOverheatPayload -->

<owl:ObjectProperty rdf:about="#hasOverheatPayload">
    <rdfs:domain rdf:resource="http://example.org/tesa#Action" />
    <rdfs:range rdf:resource="#OverheatRulePayload" />
</owl:ObjectProperty>


<!-- http://example.org/factory#OverheatPayload2022 -->

<owl:NamedIndividual rdf:about="#OverheatPayload2022">
    <rdf:type rdf:resource="#OverheatRulePayload" />
    <tesa:validDuringInterval rdf:resource="#Interval2022" />
    <capacityLimit rdf:datatype="xsd:float">200.0</capacityLimit>
    <tempThreshold rdf:datatype="xsd:float">100.0</tempThreshold>
</owl:NamedIndividual>


<!-- http://example.org/factory#OverheatPayload2023 -->

<owl:NamedIndividual rdf:about="#OverheatPayload2023">
    <rdf:type rdf:resource="#OverheatRulePayload" />
    <tesa:validDuringInterval rdf:resource="#Interval2023" />
    <capacityLimit rdf:datatype="xsd:float">220.0</capacityLimit>
    <tempThreshold rdf:datatype="xsd:float">100.0</tempThreshold>
</owl:NamedIndividual>


<!-- http://example.org/factory#SetLineCapacityAction -->

<owl:NamedIndividual rdf:about="#SetLineCapacityAction">
    <rdf:type rdf:resource="http://example.org/tesa#Action" />
    <hasOverheatPayload rdf:resource="#OverheatPayload2022" />
    <hasOverheatPayload rdf:resource="#OverheatPayload2023" />
</owl:NamedIndividual>
```

Fig. 3. Example of the TESA rules in RDF/XML notation

Additionally, it is possible to create a dynamic payload for the rules by specifying the effect and how long it remains valid using the tesa:validDuringInterval property of the specific Action. This approach allows declaring how long an action (such as a reduced throughput) stays in effect within a time:Interval and then relying on SWRL rules to evaluate the triggered conditions, check the time constraints, and apply the desired consequence accordingly. As we can see, adding the rules are flexible due to the simple presentation of the constructs. The test scenarios were selected to cover the main sources of complexity identified in the requirements: recurring scheduling changes and high-density conflicting directives. Scenario sizes (from 5 to 50 conflict groups and up to 500 total actions, with randomly assigned priorities) were chosen to stress-test the arbitration mechanism.

3.3. TESA analysis experiments

TESA analysis involves multiple dimensions to ensure that the ontology meets practical scheduling and conflict-resolution requirements. The methodology centers on empirical experiments (tests) using the modeled car manufacturing system, supplemented by interoperability and simplicity qualitative checks.

One aspect of the study examines how easily TESA-based triggers can be created or updated for recurring tasks, such as daily or monthly maintenance, and how many additional ontology constructs each change requires. Another component assesses how TESA manages conflicting directives by observing its PriorityRelator and ConflictedActions features in scenarios where triggers overlap. This component investigates whether these capabilities help prevent reasoner-

 ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

level halts or indefinite loops, and whether the system can remain operational under high load across different numbers of conflicts. Finally, the analysis determines whether TESA remains compatible with upper-level ontologies, such as BFO, UFO, or SUMO, without requiring major rewrites of time references, and contrasts TESA's scheduling approach with other ontologies to highlight any differences in model overhead, event-condition-action logic, and integrated priority-based conflict handling.

4. Results and Discussion

4.1. Ontology complexity evaluation

The initial investigation focused on how quickly and simply new triggers could be introduced or existing ones modified for daily, monthly, or arbitrary recurring tasks. Fig. 4 shows an example of creating a scheduled recurring task within the TESA ontology, expressed in the Terse RDF Triple Language (Turtle) [30].

```
:DailyWeldingCheckTrigger a :TimeTrigger ;
    :hasTimeInterval :DailyMorningInterval ;
    :triggersAction :WeldingCheckAction ;
    :hasPriority 5 ;
    :validFrom "2026-01-01T00:00:00"^^xsd:dateTime ;
    :validUntil "2026-12-31T23:59:59"^^xsd:dateTime ;
    rdfs:label "Daily Welding Station Check" ;
    rdfs:comment "Performs daily inspection at 8 AM" .
```

Fig. 4. Example of the TESA scheduling in Turtle notation

The Halstead Complexity Metrics [31] were used to evaluate the complexity level of the TESA ontology. Halstead metrics quantify program complexity based on the number of distinct operators and operands. Therefore, for ontologies, the following values and formulas are applied:

- number of distinct predicates (operators) (n_1);
- number of distinct individuals and literals (operands) (n_2);
- total occurrences of predicates (N_1);
- total occurrences of individuals and literals (N_2);
- vocabulary (n): $n = n_1 + n_2$;
- length (N): $N = N_1 + N_2$;
- calculated volume (V): $V = N \times \log_2(n)$;
- difficulty (D): $D = (n_1 / 2) \times (N_2 / n_2)$;
- effort (E): $E = V \times D$.

A lower effort value indicates a less complex ontology, as measured by the selected metric. To evaluate the ontology, the effort values were calculated for TESA and the Process Specification Language (PSL) according to the ISO standard for the following two tasks: adding three daily triggers with different time specifications (S1) and adding two monthly triggers with recurrence patterns (S2).

For each new trigger in TESA, the following elements were required:

- a TimeTrigger individual with references to the relevant interval.
- one or more Action instances or references if they did not already exist.
- a reference to PriorityRelator if conflict resolution was needed.
- zero to one minimal SWRL expansions if the rule logic was domain-specific (e.g., "If sensor reading is above threshold and/or time is 2 PM, run ActionXYZ").

The empirical results (Table 2) from the Halstead complexity assessment reveal a stark contrast in architectural efficiency between TESA and the ISO-standard PSL. In Scenario 1 (S1), which involves the integration of three daily triggers, TESA demonstrated a significantly reduced cognitive footprint, requiring an effort (E) of 5808 compared to PSL's 18886 – effectively a 70% reduction in developmental strain. This trend persists into Scenario 2 (S2), where the effort required for monthly recurrence patterns dropped to 2016 for TESA, whereas PSL remained nearly four times as demanding at 7103.

Table 2

Halstead Complexity Metrics Comparison for TESA and PSL

Metric	TESA (S1)	PSL (S1)	TESA (S2)	PSL (S2)
n_1	8	12	8	12
n_2	12	21	8	15
N_1	36	72	18	38
N_2	48	84	24	45
n	20	33	16	27
N	84	156	42	83
V	363	786.9	168	394.7
D	16	24	12	18
E	5 808.7	18 886.2	2016	7103.8

These discrepancies are largely driven by TESA's optimized use of unique operators and operands, which suggests that the framework's internal logic is more streamlined for task scheduling than traditional, more verbose ontological standards. From these metrics, TESA offers a superior balance between expressive power and structural simplicity. By requiring fewer individual instances – such as TimeTrigger and specific Action references – the framework successfully lowers the barrier for modifying or expanding autonomous systems.

4.2. Conflict resolution performance

The next phase of the TESA ontology evaluation is an investigation into the efficacy of its conflict resolution mechanisms when confronted with overlapping

triggers. Fig. 5 shows an example of conflicted action creation in the TESA ontology.

```
<!-- http://example.org/factory#BoostProductionLine -->

<owl:NamedIndividual rdf:about="#BoostProductionLine">
    <rdf:type rdf:resource="http://example.org/tesa#Action" />
    <change rdf:resource="#Line1" />
    <tesa:priorityValue
        rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">
        1
    </tesa:priorityValue>
    <tesa:executeIndicator
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
    </tesa:executeIndicator>
</owl:NamedIndividual>

<!-- http://example.org/factory#StopProductionLine -->

<owl:NamedIndividual rdf:about="#StopProductionLine">
    <rdf:type rdf:resource="http://example.org/tesa#Action" />
    <change rdf:resource="#Line1" />
    <tesa:priorityValue
        rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">
        100
    </tesa:priorityValue>
    <tesa:executeIndicator
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
    </tesa:executeIndicator>
</owl:NamedIndividual>


<!-- http://example.org/factory#ConflictedProductionLineActions -->

<owl:NamedIndividual rdf:about="#ConflictedProductionLineActions">
    <rdf:type
        rdf:resource="http://example.org/tesa#ConflictedActions"
    />
    <tesa:hasAction rdf:resource="#BoostProductionLine" />
    <tesa:hasAction rdf:resource="#StopProductionLine" />
</owl:NamedIndividual>
```

Fig. 5. Example of the TESA conflicted actions in RDF/XML notation

The TESA ontology provides the PriorityRelator and ConflictedActions to analyze and identify the applicable actions. The following steps should be performed to run only top-priority actions: first, run the reasoner with all logical rules, and then have the actions' executor select the one with the highest priority. The selection process can be implemented in different ways depending on the system's technical stack. The most popular options are to execute the SQWRL [32] query that can contain aggregations or perform filtering using the programming language. Figure 6 shows an example of SQWRL query for top-priority action selection in the TESA ontology.

Notably, the highest-priority action selection should run only after all candidates have been considered by the rules. Most reasoners, such as Pellet [33] and HermiT [34], guarantee this, but additional options can be added to the Action instances. It is also crucial to check that all actions have the execute indicator value set to false before the reasoner run to avoid any inconsistencies between the runs.

```
:ConflictedActions(?conflict) ^
:hasAction(?conflict, ?action) ^
:executeIndicator(?action, true) ^
:hasPriority(?action, ?priority) ^
sqwrl:makeBag(?priority, ?b) ^
sqwrl:groupBy(?b, ?conflict) ^
sqwrl:max(?maxp, ?b) ^
swrlb:equal(?priority, ?maxp)
-> sqwrl:select(?conflict, ?action, ?maxp)
```

Fig. 6. Example of the SQWRL query for top-priority actions selection

Conflict resolution testing was conducted to investigate the topic in detail. The testing setup involved the described car factory system with various TESA triggers referencing the same station or production line. Each trigger carried a different action and priority, posing a collision scenario (e.g., "Boost production" vs. "Stop line"). A series of collision scenarios was introduced, beginning with fewer than 5 conflicted actions (small scenario) and increasing to over 50 actions (heavier scenario). Each scenario featured randomly assigned priorities on triggers targeting the same resource, leading to overlapping or contradictory directives. The following aspects were monitored during the testing: any indefinite loops or halts, and whether manual intervention was needed. The test results show that no indefinite loops or reasoner halts arose, demonstrating that TESA's conflict resolution logic remains robust even under many collisions.

The subsequent phase of the evaluation assessed the efficacy of the TESA conflict resolution mechanism. A series of experiments with varying configurations was executed on a local workstation running Windows 11 with an Intel i5-12450H processor and 16 GB of RAM. The software environment comprised Java 2025-10-21 LTS and Python 3.13, leveraging the Pellet reasoner (version 2.3.1) via the owlready2 (version 0.47) library. The ontology files, conflict scenario generator, and benchmark scripts are provided in the publicly available repository (see the Data Availability section).

In the initial experimental suite, a fixed number of ten Actions was implemented for each ConflictedActions, while the number of ConflictedActions varied from 5 to 50. The second experimental suite featured a variable number of Actions, ranging from 2 to 50, for each ConflictedActions, while keeping the number of ConflictedActions constant at 10.

Table 3

Mean reasoning latency for TESA with ten actions per conflict, ms

Conflicted Actions	Total Actions	Mean latency	Standard deviation
5	50	765.5	42.7
10	100	776.9	28.7
15	150	795.1	34.3
25	250	838.5	37.1
50	500	927.2	50.8

Each unique configuration was subjected to 50 iterations to ensure statistical significance and minimize the impact of transient system noise. Furthermore, 10 initial "warm-up" runs were performed before the recorded trials to mitigate the effects of just-in-time compilation and resource caching, yielding more consistent and reliable performance data. The experimental results

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

presented in Tables 3 and 4 elucidate the scalability and temporal efficiency of the TESA reasoning mechanism as the structural complexity increases. In the first test suite, increasing the number of ConflictedActions from 5 to 50, corresponding to an increase from 50 to 500 total actions, was associated with a mean latency increase from 765.5 to 927.2 ms. This indicates sublinear growth in processing time with respect to the number of distinct conflict instances.

Table 4

Mean reasoning latency for TESA with 10 conflicts, ms

Actions per ConflictedActions	Total Actions	Mean latency	Standard deviation
2	20	741.1	38.4
5	50	753.6	34.7
10	100	777.4	38.9
15	150	812.6	43.8
20	200	819.3	35.1
30	300	851.4	38.7
50	500	943.1	62.4

The second suite similarly demonstrated that augmenting the internal density of actions within a constant number of conflicts (from 2 to 50 actions per conflict) resulted in a latency range spanning from 741.1 ms to 943.1 ms. The observed standard deviations across both suites, averaging 28.7-62.4 ms, highlight the stability of the Pellet reasoner within the TESA framework, even as the system faces increased computational demands.

Given the importance of evaluating the TESA's conflict-resolution mechanism, the time to reach a conclusion (latency) was measured and compared with the standard OWL+SWRL approach, in which conflicts are automatically resolved by calculating Minimal Unsatisfiability-Preserving Subsets (MUPS). Computing MUPS is a complex task and may vary across systems; thus, a simplified approach based on a one-time deletion of the conflicting axioms was adopted. Using such an approach allows the comparison of the conflict-resolution latency of TESA with that of other approaches.

Table 5

Mean reasoning latency for TESA and OWL+SWRL approaches, ms

Conflicted Actions	Total Actions	TESA mean latency	OWL+SWRL mean latency
5	10	751.1	53.2
10	20	756.4	175.5
20	40	766.5	731.7
30	60	770.9	2 315.4
40	80	788.2	6 325.6
50	100	807.7	13 680.2

The empirical analysis of the TESA conflict resolution mechanism (Table 5) demonstrates robust performance stability, particularly when compared with the standard OWL + SWRL approach. In the primary experimental suites (Tables 3 and 4), reasoning latency exhibited growth even as the system was scaled to 500 total actions. This stability highlights TESA's efficiency in managing high-density conflict sets. However, the most significant findings emerge from the comparative analysis in Table 5. While the OWL + SWRL approach initially shows lower latency for very small datasets (53.2 ms for 5 conflicts), it suffers from exponential performance degradation as complexity increases, skyrocketing to over 13680 ms for 50 conflicts. In contrast, TESA maintains a near-linear performance curve, reaching only 807.7 ms under the same conditions.

The primary conclusion from these findings is that TESA effectively alleviates the computational bottleneck in traditional MUPS-based conflict resolution. In contrast to standard OWL + SWRL techniques, which are often impractical in large-scale autonomous systems due to complex axiom-deletion processes and consequent latency spikes, TESA is a reliable and scalable alternative. The experimental results substantiate that the streamlined logical framework of TESA enables it to sustain a stable operational cadence, rendering it significantly more suitable for real-time settings where consistent response times are essential.

4.3. Compatibility with foundational ontologies

The final part of the analysis was determining whether TESA remains compatible with high-level ontologies, such as BFO, UFO, or SUMO, without requiring a major rewrite of core references to time. As an optional module built upon OWL-Time, TESA is designed to operate without interfering with the upper-level ontological commitments of BFO, UFO, or SUMO. TESA references standard OWL-Time classes for intervals and instants rather than redefining temporal primitives, extending them by introducing constructs for triggers, actions, and conflict resolution. This approach preserves existing domain-specific or foundational axioms, allowing integrators to incorporate the scheduling and prioritization mechanisms of TESA without modifying core temporal semantics. All TESA usage examples with foundational ontologies are available in the public repository, in the alignments folder.

Within a BFO-based domain, TESA operates by referencing or importing the standard "time region" classes defined by BFO, upon which it superimposes its scheduling logic (Fig. 7). Similarly, UFO-based systems can associate TESA triggers with the "moments" or "events" recognized by the UFO framework. For ontologies grounded in SUMO, TESA's interval references remain non-intrusive, ensuring that upper-level axioms remain intact. This deliberate separation of concerns contributes to TESA's relatively low integration over-

head, particularly when recurring temporal constructs, such as daily or monthly triggers or conflict-resolution mechanisms, are implemented. This design is especially advantageous for systems that prioritize direct support (ECA) semantics and concurrency management based on priority levels.

```
<owl:Class rdf:about="#ProductionLine">
    <rdfs:subClassOf
        rdf:resource="http://purl.obolibrary.org/obo/bfo.owl#Entity" />
    <rdfs:label>Production Line</rdfs:label>
</owl:Class>

<owl:NamedIndividual rdf:about="#LineA1">
    <rdf:type rdf:resource="#ProductionLine" />
    <rdfs:label>Production Line A1</rdfs:label>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#ActionCheckLineA1">
    <rdf:type rdf:resource="http://example.org/tesa#Action" />
    <rdfs:label>Daily Check Action for A1</rdfs:label>
    <tesa:priorityValue
        rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">
        1
    </tesa:priorityValue>
    <tesa:executeIndicator
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
    </tesa:executeIndicator>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#LineA1DailyCheck">
    <rdf:type rdf:resource="http://example.org/tesa#TimeTrigger" />
    <tesa:hasAction rdf:resource="#ActionCheckLineA1" />
    <tesa:validDuringInterval rdf:resource="#Interval2023" />
    <time:hour
        rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">
        2
    </time:hour>
</owl:NamedIndividual>
```

Fig. 7. Example of the TESA usage with BFO in RDF/XML notation

TESA's minimal extensions to OWL-Time enable developers to incorporate triggers, actions, and conflict-resolution mechanisms into otherwise purely descriptive temporal models with significantly lower overhead than many existing alternatives.

For example, while the Process Specification Language (PSL) provides detailed constructs for process and temporal modeling in manufacturing contexts, it does not include TESA's native numeric priority handling and typically requires external scripting to manage concurrency or event-condition-action (ECA) triggers.

Similarly, Generic Task Ontology (GTO) supports representations of scheduling intervals and abstract tasks, but it lacks native support for ECA logic and conflict resolution, often necessitating custom extensions to address complex concurrency scenarios.

In addition, iCalendar-based models are well-suited for managing recurring events but do not offer ontology-level mechanisms for conflict resolution and generally require additional development effort to align with upper-level ontologies. In contrast, TESA addresses these limitations directly by introducing TimeTrigger and Action as core constructs for representing conditions and their consequences, along with PriorityRelator for managing conflicts. This integrated design allows TESA to support priority-based reasoning and ECA semantics without requiring invasive modifications to existing temporal or process-oriented definitions.

4.4. Limitations and threats to validity

The presented evaluation has several limitations that should be considered when interpreting the results. First, the Halstead effort metric is an indirect proxy for ontology usability: it quantifies the number of distinct and total operators (predicates) and operands (individuals and literals) that a developer must manipulate. However, it does not directly measure comprehension time, error rate, or long-term maintainability. The metric was applied with the same counting convention as the TESA and PSL descriptions of the same scenarios, so the comparison remains internally fair; nevertheless, complementary structural ontology metrics and user-based studies are planned for future work.

Second, the evaluation is based on a single, representative, test system (a car manufacturing facility) and on synthetic conflict scenarios with randomly assigned priorities. Additional experiments are required to generalize the results to other domains. Third, the baseline conflict-resolution procedure approximates the MUPS computation by a one-time deletion of conflicting axioms; this is an optimistic (lower-bound) estimate of the baseline cost, and therefore, the advantage of TESA is not overstated in the comparison.

Finally, the compatibility with BFO, UFO, SUMO, and DOLCE was verified practically by importing the corresponding modules together with TESA and confirming that the combined ontology remains consistent under the Pellet and HermiT reasoners for the described test system, rather than by a formal proof of alignment.

In summary, the experimental findings indicate that TESA is proficient in accommodating frequent scheduling adjustments. They can be daily, monthly, or otherwise, requiring only minimal enhancements to the existing ontology. The efficiency of the integrated priority-based conflict resolution mechanism is proven, even when faced with numerous overlapping triggers. TESA demonstrates a reduced need for implementation resources and facilitates straightforward integration with well-established foundational ontologies when compared to other scheduling frameworks. Moreover, it allows for the management of concurrency with negligible computational overhead. This combination of diminished modeling complexity, dependable conflict handling, and compliance with OWL-Time standards underscores the potential of TESA to improve scheduling workflows within dynamic and closely interconnected cyber-physical systems, such as Digital Twins.

5. Conclusions

In this study, the new ontology, Temporal Event-driven Scheduled Actions (TESA), was developed and examined. TESA integrates dynamic conflict resolution with optimized concurrency handling, which makes it well-suited for use with foundational ontologies. The

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

OWL-Time ontology served as the basis for describing time instants and intervals in TESA, enabling easy integration with most time-based ontologies. The comprehensive evaluation of TESA shows that it significantly reduces modeling overhead for time-based triggers, recurring scheduling tasks, and concurrency handling. Building on a flexible gating mechanism for dynamic conflict resolution, TESA's approach introduces a TimeTrigger aggregator, a streamlined action-resolution pipeline based on event-condition-action logic, and selective deactivation of lower-priority triggers, thereby reducing the modeling effort by approximately 70% in terms of the Halstead effort metric without sacrificing scheduling accuracy. Empirical tests confirm that TESA's built-in priority mechanisms effectively prevent halts or indefinite loops, even in concurrency-intensive scenarios.

The performance evaluation of the TESA conflict resolution mechanism shows a notable degree of stability as structural complexity increases. The reasoning latency remains remarkably flat even as the total action count scales tenfold. In the first experimental suite, increasing the number of ConflictedActions from 5 to 50 resulted in a mean latency shift of only 161.7 ms, indicating sublinear growth and suggesting that the framework is well optimized for handling multiple discrete conflict instances simultaneously. The comparative analysis with the standard OWL+SWRL approach highlights the TESA framework's most significant architectural advantage. While the traditional approach initially outperforms TESA in low-complexity scenarios, it suffers from severe performance degradation as the scale increases. By the 50-conflict threshold, the OWL+SWRL approach reaches a prohibitive latency of 13 680.2 ms, whereas TESA remains highly efficient at 807.7 ms. By adopting a streamlined logic structure, TESA avoids the complexity of traditional methods, offering a scalable alternative that maintains near-linear performance even under significant load.

TESA also remains compatible with recognized foundational ontologies such as BFO, DOLCE, or SUMO because it layers its scheduling constructs atop existing references to intervals or events without requiring a redefinition of domain axioms. As a result, integrators can retain their usual time definitions while benefiting from TESA's triggers, priority-based overrides, and minimal overhead. By combining dynamic gating, lightweight expansions, and conflict-resilient event-condition-action design, TESA achieves concurrency handling on par with more expansive scheduling frameworks, all while reducing the coding burden and delivering stable, dynamic time-based event management.

In conclusion, the comparative results highlight the advantages of the new ontology in contexts that necessitate rapid reconfiguration, precise scheduling, and the coordination of overlapping event-action triggers. While foundational ontologies may offer greater conceptual depth, the proposed ontology successfully meets an essential requirement: integrating time-centric dynamics, real-time triggers, and conflict resolution into a cohesive semantic framework. Future research should investigate the potential of distributed or cloud-based implementations of this ontology across large-scale manufacturing networks. Future tasks in the field of TESA development include refining the ontology's modular structure for specialized domains, exploring automated alignment techniques with other foundational ontologies at a higher level, and investigating robust mechanisms for dynamic scalability in real-world, heterogeneous scheduling environments. Additionally, it is imperative to broaden the priority-based logic to include multi-factor optimizations, thereby enhancing its relevance and applicability to practical production systems.

Contributions of authors: conceptualization, problem statement, formulation of tasks, software implementation of ontologies and development of a system for evaluating them, experimental research, analysis of the obtained results, formulation of the novel approach, and conclusions, review and editing – **Oleksii Atamaniuk;** conceptualization, problem statement, analysis of the obtained results, formulation of the novel approach, review, editing – **Yevgeniya Sulema.**

All authors have read and approved the final manuscript for publication.

Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Data Availability

The TESA ontology (OWL/RDF file), the model of the test system, the reasoner and benchmark scripts, and the generators of the conflict scenarios developed within this study are publicly available at: https://github.com/FireAndBlood12/tesa-ontology

Use of Artificial Intelligence

Artificial intelligence tools were used solely to improve the language, grammar, and readability of this manuscript. No scientific content, results, data, or conclusions were generated by artificial intelligence. The authors have reviewed and edited all suggested changes and take full responsibility for the entire content of this

publication.

References

1. Abayadeera, M. R. Digital Twin Technology: A Comprehensive Review. *International Journal of Scientific Research and Engineering Trends*, 2024, vol. 10, no. 4, pp. 1485–1504. DOI: 10.61137/ijsret.vol.10.issue4.199.

2. El-Agamy, R. F., Sayed, H. A., AL Akhatatneh, A. M., Aljohani, M., & Elhosseini, M. Comprehensive analysis of digital twins in smart cities: a 4200-paper bibliometric study. *Artificial Intelligence Review*, 2024, vol. 57, article no. 154. DOI: 10.1007/s10462-024-10781-8.

3. Muirhead, B. K., & Thomas, D. The Art and Science of Systems Engineering Tightly Coupled Programs. *SAE International Journal of Passenger Cars – Electronic and Electrical Systems*, 2010, vol. 3, no. 2, pp. 117–130. DOI: 10.4271/2010-01-2321.

4. Steinmetz, C., Rettberg, A., Ribeiro, F. G. C., Schroeder, G., & Pereira, C. E. Internet of Things Ontology for Digital Twin in Cyber Physical Systems. *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, IEEE, 2018, pp. 154–159. DOI: 10.1109/sbesc.2018.00030.

5. Curé, O., Faye, D., & Blin, G. Towards a better insight of RDF triples Ontology-guided Storage system abilities. *arXiv preprint*, arXiv:1306.6670, 2013. DOI: 10.48550/ARXIV.1306.6670.

6. Karabulut, E., Pileggi, S. F., Groth, P., & Degeler, V. Ontologies in digital twins: A systematic literature review. *Future Generation Computer Systems*, 2024, vol. 153, pp. 442–456. DOI: 10.1016/j.future.2023.12.013.

7. Tooth, J. M., Tuptuk, N., & Watson, J. D. M. A Systematic Survey of the Gemini Principles for Digital Twin Ontologies. *arXiv preprint*, arXiv:2404.10754, 2024. Available at: https://arxiv.org/abs/2404.10754 (accessed 18 January 2026).

8. Ermolayev, V., Batsakis, S., Keberle, N., Tatarintseva, O., & Antoniou, G. Ontologies of time: review and trends. *International Journal of Computer Science and Applications*, 2014, vol. 11, no. 3, pp. 57–115. Available at: https://www.researchgate.net/publication/271906680_Ontologies_of_Time_Review_and_Trends (accessed 18 January 2026).

9. W3C. Time Ontology in OWL. W3C Candidate Recommendation, 2022. Available at: https://www.w3.org/TR/owl-time (accessed 18 January 2026).

10. Schmidt, D., Trojahn, C., & Vieira, R. Matching BFO, DOLCE, GFO and SUMO: an evaluation of OAEI 2018 matching systems. *Proceedings of the XII Seminar on Ontology Research in Brazil and III Doctoral and Masters Consortium on Ontologies*, Porto Alegre, Brazil, 2019, vol. 2519. Available at: https://ceur-ws.org/Vol-2519/paper7.pdf (accessed 18 January 2026).

11. de Farias, T. M., Roxin, A., & Nicolle, C. SWRL rule-selection methodology for ontology interoperability. *Data & Knowledge Engineering*, 2016, vol. 105, pp. 53–72. DOI: 10.1016/j.datak.2015.09.001.

12. Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E. M., & Vieu, L. DOLCE: A descriptive ontology for linguistic and cognitive engineering. *Applied Ontology*, 2022, vol. 17, no. 1, pp. 45–69. DOI: 10.3233/ao-210259.

13. Otte, J. N., Beverley, J., & Ruttenberg, A. BFO: Basic Formal Ontology. *Applied Ontology*, 2022, vol. 17, no. 1, pp. 17–43. DOI: 10.3233/ao-220262.

14. Silva Muñoz, L., & Grüninger, M. Mapping and Verification of the Time Ontology in SUMO. *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2016, vol. 283, pp. 109–122. DOI: 10.3233/978-1-61499-660-6-109.

15. Patiniott, N., Borg, J., Farrugia, P., Mercieca, A., Gatt, A., & Casha, O. Design and Implementation of an Ontology-Driven Cyber–Physical Prosthesis Service System for Personalised and Adaptive Care. *Applied Sciences*, 2025, vol. 15, no. 23, article no. 12637. DOI: 10.3390/app152312637.

16. Guizzardi, G., Benevides, A. B., Fonseca, C. M., Porello, D., Almeida, J. P. A., & Sales, T. P. UFO: Unified Foundational Ontology. *Applied Ontology*, 2022, vol. 17, no. 1, pp. 167–210. DOI: 10.3233/ao-210256.

17. Trojahn, C., Vieira, R., Schmidt, D., Pease, A., & Guizzardi, G. Foundational ontologies meet ontology matching: A survey. *Semantic Web*, 2022, vol. 13, no. 4, pp. 685–704. DOI: 10.3233/sw-210447.

18. Khan, Z. C., & Keet, C. M. The Foundational Ontology Library ROMULUS. *Model and Data Engineering (MEDI 2013), Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, vol. 8216, pp. 200–211. DOI: 10.1007/978-3-642-41366-7_17.

19. Grüninger, M., & Menzel, C. The process specification language (PSL) theory and applications. *AI Magazine*, 2003, vol. 24, no. 3, pp. 63–74. DOI: 10.1609/aimag.v24i3.1719.

20. Rajpathak, D., Mota, E., & Roy, R. A Generic Task Ontology for Scheduling Applications. 2012. Available at: https://www.researchgate.net/publication/242444791_A_Generic_Task_Ontology_for_Scheduling_Applications (accessed 18 January 2026).

21. Miller, L., & Connolly, D. RDFiCal: iCalendar in RDF. *W3C Interest Group*, 2005. Available at: https://www.researchgate.net/publication/247337367_RDFiCal_iCalendar_in_RDF (accessed 18 January 2026).

22. Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 1995, vol. 43, no. 5–6, pp. 907–928. DOI: 10.1006/ijhc.1995.1081.

23. Macer, D. B., Jennions, I. K., & Avdelidis, N. P. A Review of an Ontology-Based Digital Twin to Enable Condition-Based Maintenance for Aircraft Operations. *Applied Sciences*, 2025, vol. 15, no. 20, article no. 11136. DOI: 10.3390/app152011136.

24. Uschold, M., & Gruninger, M. Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 1996, vol. 11, no. 2, pp. 93–136. DOI: 10.1017/S0269888900007797.

25. Sulema, Y., Dychka, I., & Sulema, O. Multimodal Data Representation Models for Virtual, Remote, and Mixed Laboratories Development. *Smart Industry & Smart Education: Proceedings of the 15th International Conference on Remote Engineering and Virtual Instrumentation (REV), Lecture Notes in Networks and Systems*, Springer, Cham, 2018, pp. 559–569. DOI: 10.1007/978-3-319-95678-7_62.

26. Sulema, Y., & Kerre, E. Multimodal Data Representation and Processing Based on Algebraic System of Aggregates. *Mathematical Methods in Interdisciplinary Sciences*, Wiley, USA, 2020, pp. 63–97. DOI: 10.1002/9781119585640.ch5.

27. Allen, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 1983, vol. 26, no. 11, pp. 832–843. DOI: 10.1145/182.358434.

28. W3C. Spatial Data on the Web Use Cases & Requirements. W3C Working Group Note, 2016. Available at: https://www.w3.org/TR/sdw-ucr/#SpatialRelationships (accessed 18 January 2026).

29. Randell, D., Cui, Z., & Cohn, A. A spatial logic based on regions and connection. *Proceedings of the 3rd In-*

ternational Conference on Knowledge Representation and Reasoning, 1992, pp. 165–176. Available at: https://www.researchgate.net/publication/221393453_A_Spatial_Logic_based_on_Regions_and_Connection (accessed 18 January 2026).

30. W3C RDF Working Group. RDF 1.1 Turtle: Terse RDF Triple Language. W3C Recommendation, 2014. Available at: https://www.w3.org/TR/turtle (accessed 18 January 2026).

31. Sinha, P. K., & Dutta, B. A Study of Various Dimensions of Ontology Development Methodologies. *Journal of Information and Knowledge*, 2025, vol. 62, no. 2, pp. 119–146. DOI: 10.17821/srels/2025/v62i2/171809.

32. Bjørnskov, J., Badhwar, A., Shikhar Singh, D., Sehgal, M., Åkesson, R., & Jradi, M. Development and demonstration of a digital twin platform leveraging ontologies and data-driven simulation models. *Journal of Building Performance Simulation*, 2025, vol. 18, pp. 1–21. DOI: 10.1080/19401493.2025.2504005.

33. Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2007, vol. 5, no. 2, pp. 51–53. DOI: 10.1016/j.websem.2007.03.004.

34. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 2014, vol. 53, no. 3, pp. 245–269. DOI: 10.1007/s10817-014-9305-1.

Received 29.01.2026, Received in revised form 19.03.2026
Accepted date 18.07.2026, Published date 31.07.2026

ОНТОЛОГІЯ ЧАСОВИХ ПОДІЙНО-КЕРОВАНИХ ЗАПЛАНОВАНИХ ДІЙ (TESA) З МЕХАНІЗМОМ ПРІОРИТЕТІВ ДЛЯ КІБЕРФІЗИЧНИХ СИСТЕМ

О. В. Атаманюк, Є. С. Сулема

Предметом *вивчення в статті є розроблення та оцінювання легковагових онтологій з механізмами планування та розв'язання конфліктів для моделювання тісно пов'язаних складних систем у кіберфізичних системах, таких як цифрові двійники.* ***Метою*** *є розроблення розширення онтології OWL-Time засобами планування, подієво-керованих дій та розв'язання конфліктів, а також підтвердження його ефективності за чотирма вимірюваними критеріями: зменшенням зусиль за метрикою Голстеда порівняно з ISO-стандартом PSL; отримання затримки логічного виводу близької до лінійної для 5–50 груп конфліктів; відсутністю зупинок різонера; сумісністю з фундаментальними онтологіями, такими як BFO, SUMO, UFO та DOLCE.* ***Завдання****: збір вимог до нової онтології; опис основних елементів системи, що моделюються онтологією, із забезпеченням сумісності з фундаментальними онтологіями; розроблення механізму пріоритетів для розв'язання конфліктних дій; опис та оцінювання нової онтології з особливим фокусом на обмеженні зупинок, викликаних конфліктними діями під час роботи механізму логічного виведення онтології.* ***Методи****, що використовуються: проєктування тестової системи для експериментальної перевірки; формалізація системи для моделювання з використанням запропонованої та існуючих онтологій; методи програмної інженерії для запуску експериментів і збору даних про продуктивність; метрики зусиль Холстеда для вимірювання когнітивних зусиль та зусиль на розробку. Отримано наступні* ***результати****: запропонована онтологія TESA дозволяє досягти зниження метрик зусиль Холстеда на 70% порівняно зі стандартом ISO PSL, що вказує на значне зменшення когнітивних зусиль та зусиль на розробку; TESA запобігає виникненню зупинок механізму логічного виводу; у сценаріях з високою щільністю конфліктів TESA демонструє чудову часову стабільність із затримкою логічного виведення менше секунди, тоді як традиційні методології зазнають експоненціального зниження продуктивності; онтологія демонструє легковагову архітектуру та безперешкодну сумісність із фундаментальними онтологіями; модульна структура TESA та зменшене семантичне навантаження дозволяють спростити впровадження та інтеграцію.* ***Висновки****. Наукова новизна отриманих результатів полягає в наступному: вперше розроблено легковагову онтологію для планування подій і запобігання конфліктам, яка забезпечує баланс між детальною семантикою та простотою інтеграції з фундаментальними онтологіями; модульна структура дозволяє адаптувати її до спеціалізованих доменів, зберігаючи сумісність із фундаментальними онтологіями вищого рівня; механізм розв'язання конфліктів на основі пріоритетів запобігає зупинкам механізму логічного виводу та підтримує стабільну продуктивність у сценаріях із великою кількістю конфліктів; архітектура онтології зменшує когнітивне навантаження, зберігаючи потужні можливості логіки планування.*

Ключові *слова:* *онтологія подійного планування; вирішення конфліктів; подійно-орієнтована онтологія; кіберфізична система; цифровий двійник; інженерія програмного забезпечення.*

Атаманюк Олексій Віталійович – аспірант кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Сулема Євгенія Станіславівна – д-р техн. наук, доц., завідувач кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Oleksii Atamaniuk – PhD Student of the Department of Computer Systems Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine,
e-mail: alexata1204@gmail.com, ORCID: 0000-0002-9038-9691.

Yevgeniya Sulema – Doctor of Technical Sciences, Associate Professor, Head of the Department of Computer Systems Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine,
e-mail: sulema@pzks.fpm.kpi.ua, ORCID: 0000-0001-7871-9806, Scopus Author ID: 36601536300.