SN Computer Science (2023) 4:632
https://doi.org/10.1007/s42979-023-01975-y

ORIGINAL RESEARCH

DAEMON: A Domain-Based Monitoring Ontology for IoT Systems

Said Daoudagh[1] · Eda Marchetti[1] · Antonello Calabrò[1] · Filipa Ferrada[2,3] · Ana Inês Oliveira[2,3] · José Barata[2,3] · Ricardo Peres[2,3] · Francisco Marques[2,3]

Received: 17 December 2022 / Accepted: 29 May 2023
© The Author(s) 2023

Abstract
Internet of Things (IoT) is an emerging technology used in several contexts and domains. The work aims to define a technological reference solution specifically conceived for monitoring and assessing the behavior of IoT systems from the cybersecurity perspective when a new device or component joins the system. We leverage semantic web technologies, such as ontologies, for defining DAEMON, a domain-based ontology that formally models monitoring, IoT, and System of Systems (SoS) domains' knowledge. We also propose a supporting architecture and describe the proof-of-concept implementing different components. We have validated and showcased our proposal by instantiating DAEMON into a multi-robot autonomous navigation scenario applied to the intralogistics domain.

Keywords Cyber security · Internet of Things (IoT) · Monitoring · Ontology · System of Systems (SoS)

Introduction

Quality, trustworthiness, and cybersecurity are essential attributes of Internet of Things systems and ecosystems. However, even if an important part of industrial and academic research activity is devoted to conceiving and implementing solutions able to guarantee, monitor, and assess these attributes during the overall development process, the time-to-market, the productivity, and the competitiveness impose quick releasing deadlines that may decrease the general quality level and increase the number of possible vulnerabilities and weaknesses. Therefore, in line with the development lifecycle, practical, integrated methodologies and mechanisms for leveraging the understanding and management of the functional and non-functional quality attributes, evaluating the development risks, and reducing the vulnerability threats are necessary.

As highlighted in [1], a commonly applied solution for satisfying these needs is using a monitoring system that enables the analysis of functional and non-functional properties

This article is part of the topical collection "Multidisciplinary Research Perspectives for IoT Systems" guest edited by Luis Camarinha-Matos, Luis Ribeiro, Paul Havinga and Srinivas Katkoori.

✉ Said Daoudagh
said.daoudagh@isti.cnr.it

Eda Marchetti
eda.marchetti@isti.cnr.it

Antonello Calabrò
antonello.calabro@isti.cnr.it

Filipa Ferrada
faf@uninova.pt

Ana Inês Oliveira
aio@uninova.pt

José Barata
jab@uninova.pt

Ricardo Peres
ricardo.peres@uninova.pt

Francisco Marques
fam@uninova.pt

[1] Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" - ISTI, CNR, Via G. Moruzzi 1, Pisa 56124, Pisa, Italy

[2] Uninova Institute, Centre of Technology and Systems (CTS) and Associated Lab of Intelligent Systems (LASI), Campus de Caparica, 2829-516 Caparica, Portugal

[3] NOVA School of Science and Technology, NOVA-SST, Campus de Caparica, 2829-516 Caparica, Portugal

Published online: 23 August 2023

during the Internet of Things execution.[1] Indeed, thanks to the analysis of the events produced by the systems, devices, or components during their online use, an assessment of specific properties can be performed. Additionally, the monitoring system promptly raises alarms or implements countermeasures in case of abnormal behavior detection. Despite the exposed efficacy, the main objection against adopting the monitoring system is the costs (in terms of effort and time) related to its design, implementation, and management. Indeed, monitoring activity needs to be integrated into the stages of the software development process and involves several stakeholders, such as Internet of Things domain experts, developers, or monitoring experts [1, 3]. Based on the initial proposal of [1, 4] in this paper, we leverage the methodology and architecture of the Domain bAsEd Monitoring ONtology (DAEMON) framework useful for realizing a manageable and user-friendly integrated monitoring system. DAEMON integrates concepts and definitions about the Internet of Things and monitoring into a unique manageable ontology-based representation. In particular, we leverage the recent proposal by:

1. revising the initial DAEMON ontology core elements to represent better the IoT systems and monitoring concepts and their relationships.
2. detailing a new reference architecture and its GUI, components' interaction, and roles.
3. detailing the validation inside a Multi-Robot Navigation use case scenario and its specific functional and non-functional properties.

In particular, we focused on the following research challenges:

CH1: Whitening the black-box assessment process

DAEMON adopts a specific mechanism for collecting internal execution data (white-box data) without knowing their source code structure (black-box data). Indeed, the implemented methodology makes the IoTs (and the execution environment) more "transparent" for functional and non-functional properties assessment and prediction without revealing their internals. Data are collected, preserving the principles of loose coupling and implementation neutrality.

CH2: Separating properties predictions and assessment

DAEMON has been conceived considering the principle of independence between the components. The internal and external DAEMON components have a specific role and contribute to the overall quality, usability, and effectiveness. Excluding a few mandatory collaborations (like, for instance, the engine dedicated to the monitoring activity), the DAEMON has been conceived to work with few adjustments in heterogeneous environments. By referring "DAEMON Ontology Modules" and our previous work [1] for more details, DAEMON peculiarities are as follows:

- Providing functional and non-functional monitoring properties as external data sets.
- Providing independent knowledge management processes for classifying and collecting the specific peculiarities, properties, and quality attributes of the different IoTs and their components and devices useful for the development and assessment activities.

CH3: Leveraging the existing monitoring solutions

DAEMON leverages the existing monitoring solutions considering several aspects. In particular, we refer to the following:

- The functional and non-functional properties collected through the knowledge management process be easily customized, enriched, or modified. The properties dataset is exploitable for further application or research activity.
- The implemented knowledge management process lets manual/automatic countermeasures be customized based on the risk analysis. This possibility allows for mitigating the vulnerability detection risks during the monitoring activities.

Outline. "Related Works" briefly puts our research in context by describing the most relevant literature concerning Cybersecurity Specification and Vulnerability, Ontology-based System of Systems, and Monitoring systems. "DAEMON Ontology Modules" describes the main modules of DAEMON ontology by highlighting concepts and their relationship that we identified and revised. "Ontology-Based Knowledge Management Architecture" introduces the DAEMON's reference architecture, reports its main components, and briefly shows how they interact. In "An Intralogistics Domain Use Case Scenario", we showcase our proposal by instantiating DAEMON into a multi-robot autonomous navigation scenario applied to the intralogistics domain. Finally, "Conclusions" concludes the paper and highlights our future works.

[1] In this paper, we rely on the definition provided in [2] that considers the IoT system as "the latest example of the System of Systems (SoS)."

Related Works

Cybersecurity Specification and Vulnerability

Cybersecurity implies a set of methods, technologies, and processes to protect against digital attacks that would seriously affect the integrity of data, networks, and programs. In recent years, much effort has been put into functional and non-functional (including security, safety, and privacy) requirements specification, which are essential for research and technology development. Some examples can be found in the latest European Projects documents [5, 6], in standards and specifications [7, 8], and in available backlog list containing structured security and privacy user stories [9, 10]. In this context, this article proposes a methodology for collecting and organizing the knowledge which is available in several wide-ranging sources into a unique reference ontology—the DAEMON ontology. In line with challenges *CH2* presented in the introduction ("Introduction"), the aim is to allow the selection of the most suitable functional and non-functional properties useful for monitoring and assessing the expected IoT/SoS/Ecosystem/components behavior purposes.

Ontology-based System of Systems (SoS)

In recent years, several research has been conducted on modeling System-of-Systems (SoS). Ontology is one SoS modelling approach that can be used to establish domain concepts and connect the SoS processes consistently using common language and semantics, which are essential in the planning and analysis processes of Systems-of-Systems Engineering (SoSE) [11–15]. Ontologies can have a more holistic nature where the representation of categories and relations are common to all domains such as BFO or DOLCE [16–19] or might be domain-specific intended to describe individual systems or specific domains of interest [15, 20].

Internet of Things (IoT) systems can be engineered from the perspective of SoS. As known, IoT applications involve the integrated operation of many subsystems (or Constituent Systems) that are physically and functionally heterogeneous, preserving their advanced cyber-physical functionalities. Therefore, it is important to develop ontologies to share semantic information between different IoT subsystems. Over the last few years, an effort has been made by the semantics and IoT research communities to develop ontologies that capture the related concepts and relationships between different entities. Some examples are the SSN ontology [21], the M3 Ontology [22], the oneM2M base ontology [23] and the IoT-Lite ontology [24].

In its turn, the ontology proposed in this paper aims to represent the monitoring of SoS knowledge, introducing new concepts useful for a better representation of knowledge, such as the rule hierarchy and skill. It is specifically conceived for being the connection between SoS and the monitoring ontologies that, to the best of the author's knowledge, is still not yet offered. In addition, existing IoT/SoS ontologies such as IoT-Lite or oneM2M Base Ontology can be integrated and reused in the proposal in the DAEMON "SoS module" (see "DAEMON Ontology Modules").

Monitoring Systems

Monitoring systems allow information collection, processing, analysis, and dissemination within a project or system, aiming at supporting decision-making. They can be found in applications spanning numerous domains, including traffic [25], automotive [26], avionic [27], healthcare [28], industry [29]. In almost all the application contexts, the existing monitoring proposals aim to i) provide powerful, concise, and unambiguous specification languages for the validation properties specification [30]; ii) defining mechanisms for the conformity assessment of the system against the selected properties [31].

With the increase of cyber attacks over the internet, it is essential to explore monitoring solutions to mitigate risks. Some promising solutions are *Security Information and Event Management* (SIEM), *Security Orchestration, Automation, and Response* (SOAR), *eXtended Detection and Response* (XDR), or *Endpoint Detection and Response (EDR)*. *SIEM* systems gather, aggregate, and normalize information from various events related to potential security violations that occurred within the system [32]. SOAR solutions, seen as the next generation of SIEM, add orchestration, automation, and response capabilities that enable dynamic interactions at all phases of the incident workflows to cope with existing and emerging (or unknown) threats [33]. On the other hand, XDR and EDR can boost the SIEM analysis by providing information and tools that will enhance the analysis executable through the SIEM. Usually, the collected data are stored in *Data Lake* [34], useful for advanced forensic analysis.

This work aims to leverage the existing proposals and provide a collaborative, easy-to-use, and effective solution for applying the monitoring activity inside target domains. In particular, focusing on challenges *CH1* and *CH3* presented in the introduction ("Introduction"), we provide

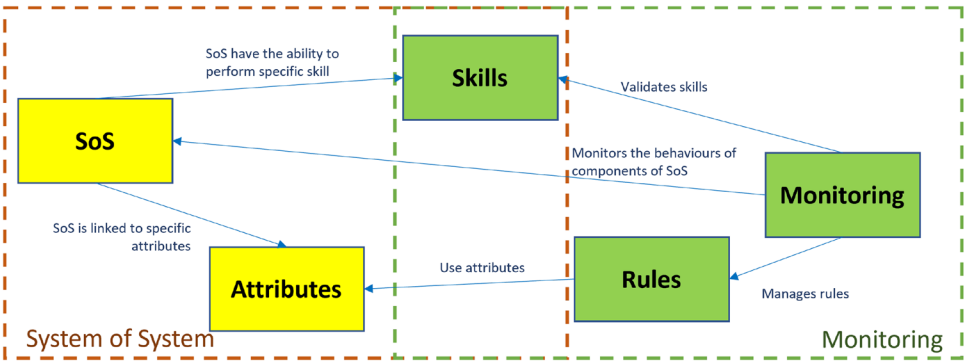

Fig. 1 DAEMON Ontology modules [1]

facilities for easily: (1) identifying the most suitable functional and non-functional properties that can be used during the monitoring activity to detect failures and vulnerabilities promptly; (2) detecting critical problems, failures, and security vulnerabilities; (3) classifying and expressing important warnings or system reconfiguration to assure a trustworthy execution.

DAEMON Ontology Modules

To enable a solution envisioned to support a monitoring process and evaluate the behaviour of the system each time, a new IoT device is changed or added, the DAEMON is proposed. DAEMON is, therefore, an extension of an initial core ontology, i.e., MONitoring onTOLOGY (MONTOLOGY) [1, 4], managed by the DAEMON Manager (see "Ontology-Based Knowledge Management Architecture"). The improvement comprises reinforcing concepts and reorganizing the content to be more manageable and modularly to permit interoperability and facilitate both extensibility and maintainability. Additionally, considering the existing IoT/SoS ontologies, such as the ones later on mentioned in "Related Works", DAEMON intends to provide means to integrate later and/or reuse them in the *SystemOfSystems* Module (please see description below).

In this line, DAEMON intends to mainly assist SoS stakeholders in gathering functional and non-functional properties related to the various parts of *IoT/SoS*. Therefore, it enables the definition of concrete monitoring rules associated with a specific property to demonstrate compliance (or non-compliance) with the selected properties. Therefore, the conceived enhanced ontology is intended to be modular, manageable, and comprehensive, being related to the challenges *CH1*, *CH2* and *CH3* described in "Introduction", allowing Knowledge derivation reasoning and inference of new knowledge. As depicted in Fig. 1, DAEMON is divided into five modules: *SoS*, *Attributes*, *Skills*, *Monitoring*, and *Rules*, being all of them described in the remainder of this section.

As such, the core ontology is composed of two main modules: the *SystemofSystems* (SoS) Module (containing eight concepts) and the *Monitoring* Module (which includes five concepts), with a total of 13 (thirteen) concepts.

SoS Module

Considering IoT systems from an SoS perspective, as described in "Related Works", it is important to maintain and share semantic information between its different subsystems. In this line and preserving physical and functional properties of the IoT/SoS systems, in DAEMON, as illustrated in Fig. 2, considering its operation and execution context, the *SystemOfSystems* is modeled as a composition of *System*, and it is influenced by a specific *Environment*. Moreover, being the target of the monitoring actions, the *System* is composed of a set of *Devices*.

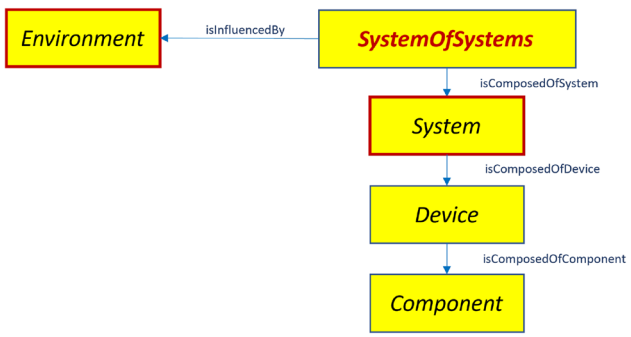

Fig. 2 DAEMON System of Systems (SoS) Module [1]

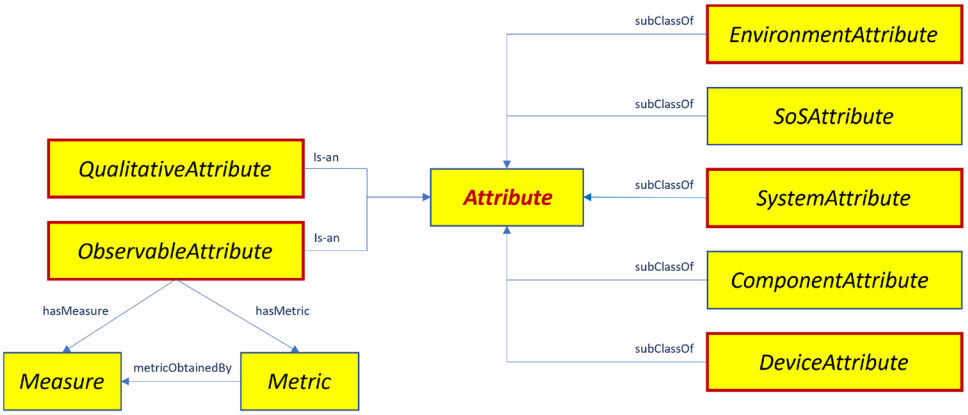

Fig. 3 DAEMON Attributes Module [1]

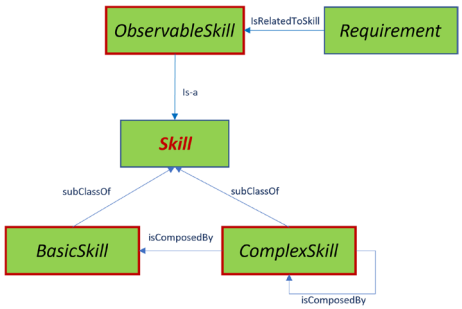

Fig. 4 DAEMON Skills Module [1]

Attributes Module

An *Attribute* is a functional and non-functional property related to a specific SoS concept. To enable the monitoring of the behavior through specific monitoring rules, the *Attribute* hierarchy is developed by combining three major sub-classes: *EnvironmentAttribute*, *SystemAttribute*, and *DeviceAttribute*. As shown in Fig. 3, this module also includes concepts related to the observable properties of the classes in the SoS module and, similarly, it comprises two specific concepts: *QualitativeAttribute* and *ObservableAttribute*, both applied for quantitative attributes used to define both the *Measure* and *Metric* part of the definition of the monitoring rules. In this context, among others, examples of *Attributes* can be: communication latency between components; or to avoid/detect DoS attacks if the average amount of messages is below a specific amount; or the total amount of the allowed/authorized connections.

Skills Module

The concept of *Skill* represents the ability of an agent (being it active or passive) to execute a particular activity. To appropriately model and leverage the *Skill* concept, it is composed of two specific sub-classes: *BasicSkill* and *ComplexSkill*, as perceived in Fig. 4. A *ComplexSkill* is composed, on the one hand, by a collection of *BasicSkill*, and on the other hand, can also be composed iteratively by a set of *ComplexSkill*. Furthermore, the concept of ObservableSkill is proposed as a means to observe the ability associated with the SoS concept that the monitoring facilities can assess. Examples of *Skill* can be, for instance, the ability of connection or movement. Moreover, directly related to *ObservableSkill*, through a *isRelatedToSkill*

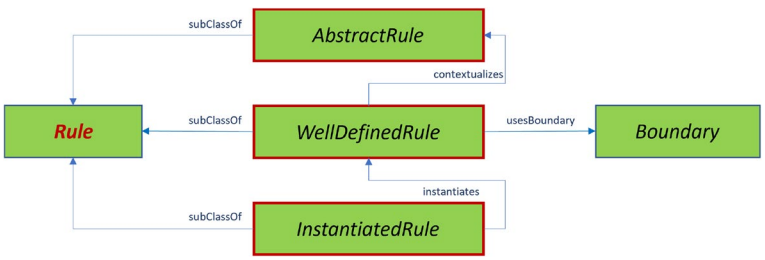

Fig. 5 DAEMON Rule Module [1]

Fig. 6 Rule Transformation Process [1]

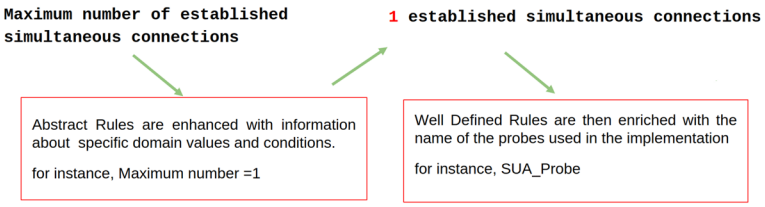

Fig. 7 From Abstract to Well-defined Rule Enrichment Process [1]

```
declare   // Optional
rule "rule name"
    // Attributes
    when
        // Conditions
    then
        // Actions
```

Fig. 8 Drools Rule Skeleton [1]

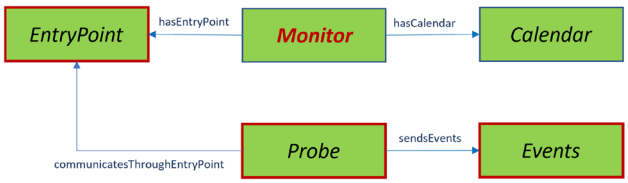

Fig. 9 DAEMON Monitoring Module [1]

association, is the *Requirement* concept. In this way, each *ObservableSkill*, specified as a collection of *Requirements*, can be validated through a specific *Rule*.

Rule Module

A *Rule* can be defined as a collection of guidelines for assessing the events of one or more issues in a flow or a cloud of events. Commonly, rules are defined as a set of if-then-else structures. As such, *Rules* are a core concept to allow the monitoring of certain *Skills*.

As illustrated in Fig. 5, in DAEMON, *Rule* is composed of three basic sub-classes: *AbstractRule*, *WellDefinedRule*, and *InstantiatedRule*, where: (i) *AbstractRule* is based on a generic rule, gathered from the navigation of DAEMON, that is still not instantiated within the execution context; (iii) *WellDefinedRule*, correlated to the monitoring of a precise *Device*, where the rule is prepared for translation to the target language of a Complex Event Processor (CEP); *InstantiatedRule* that is envisaged to be translated in a language being understandable by a monitor engine. Additionally, the *Boundary* concept includes the specific applicability value ranges in which the *Rule* is expressed.

Figure 6 illustrates the complex process flow of the evolution of rules: starting in *AbstractRule*, to *InstantiatedRule*, to obtain a processable rule. In this context, the *AbstractRule* can be seen as a generic natural language description of the purpose of the auditing activity that is simply comprehensible by non-expert users, such as the maximum number of connections between two components that are established simultaneously. Then, the *AbstractRule* is handled to *WellDefinedRule* as a semi-structured and implementable rule. The users need to add specific details about the context.

An example of *AbstractRule* and *WellDefinedRule* is depicted in Fig. 7, where the main difference between them is illustrated in the example of the maximum number of simultaneous connections.

Lastly, the *WellDefinedRule* can be enhanced with the name of the applied probes and automatically translated into an *InstantiatedRule* to the monitoring language used. This is then used during the execution by Monitoring System. Thus, a standard structure of an *InstantiatedRule* can be seen as follows in Fig. 8, where it is possible to include one or more rules that specify the rule conditions and actions (when and then):

Monitoring Module

The main aim of the Monitoring module is to provide the basic functionalities to model the basic concepts and their relationships. The central concept of this module is the *Monitor* that primarily follows rules that are organized in the *Calendar*, which includes a collection of rules. In turn, during run-time, the *Calendar* concept can confirm a given *ObservableSkill* in the Skills module. As illustrated in Fig. 9, the *Monitor* can communicate with a *Probe* through a specific *EntryPoint* used for communication.

As such, a *Probe* can be described as a piece of software code that is suitable to be injected into a monitored or observed *Component*, *Device*, or *System*. The existence of actions in an observed SoS entity produces relevant information in the form of *Events* that can be sent, by the Probes, in accordance with a specific format and at regular intervals or in a particular situation. Therefore, the term *Event* describes the state change within or of a *System*. This state change is caused when a method call is executed or internal action is authorized. In this case, the injected *Probe* packs the atomic activity into an *Event* and notifies the *Monitor* to perform the processing action on the event course. To be correctly managed by a concrete *Monitor*, the *Event* should contain several

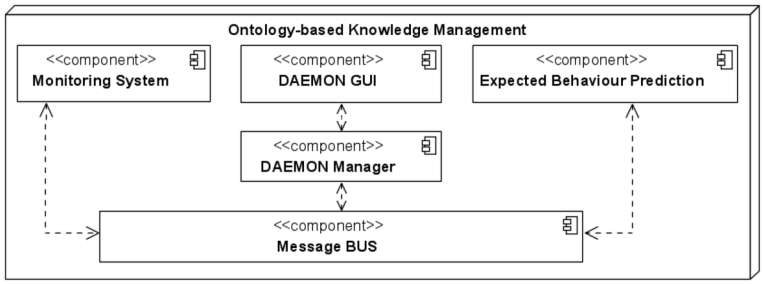

Fig. 10 DAEMON Reference Architecture

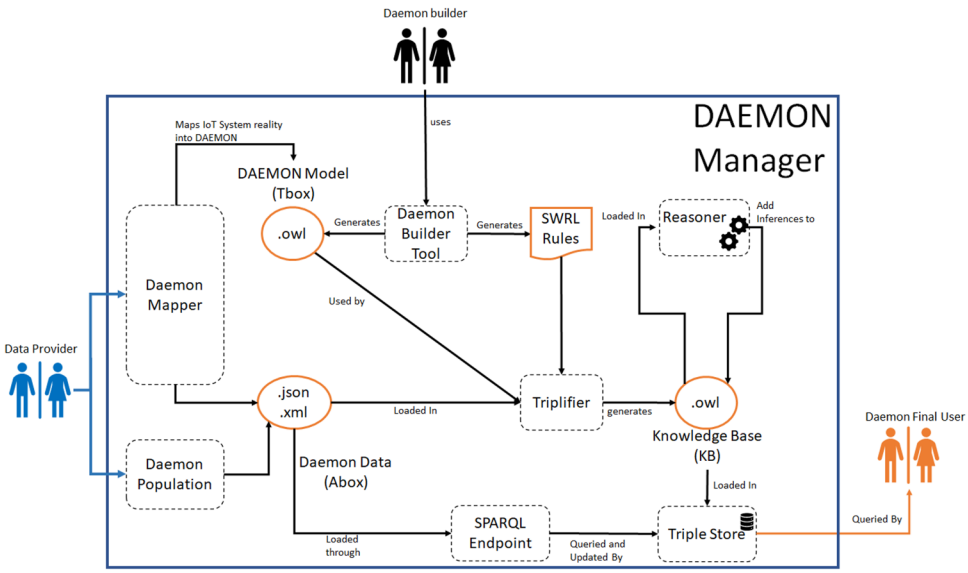

Fig. 11 DAEMON Manager Architecture

pieces of information needed for analyzing a snapshot of what is happening within the IoT system under monitoring.

Ontology-Based Knowledge Management Architecture

This section describes the architecture where the DAEMON ontology is supposed to be integrated. In line with challenges *CH2* and *CH3* presented in the introduction ("Introduction"), the DAEMON architecture has been conceived considering the principle of independence between the components and to be used in heterogeneous environments. Additionally, it provides the customization and management of functional and non-functional properties and exploitable dataset for further monitoring and validation activities.

By referring to "DAEMON Ontology Modules" and our previous work [1] for more details, DAEMON peculiarities are as follows: as reported in Fig. 10, DAEMON solution relies on two main components that collaborate to predict expected behavior and monitor functional and non-functional properties of a given IoT System. In particular, the proposed architecture includes the following components.

DAEMON GUI

This component lets the user interaction to perform all the phases related to setup and settings. It interacts with the DAEMON Manager, requesting information and providing user inputs. It also manages the current state, saving the user-provided information for later use.

Message BUS

This component manages the communication between all the framework parts. It, therefore, allows the flow of specific messages according to a predefined schema that enables interoperability and extensibility of the proposed framework.

Expected Behaviour Prediction

This component is responsible for predicting the behavior of the monitored IoT System. Usually, it is supposed to collaborate with the Monitoring system to predict future events.

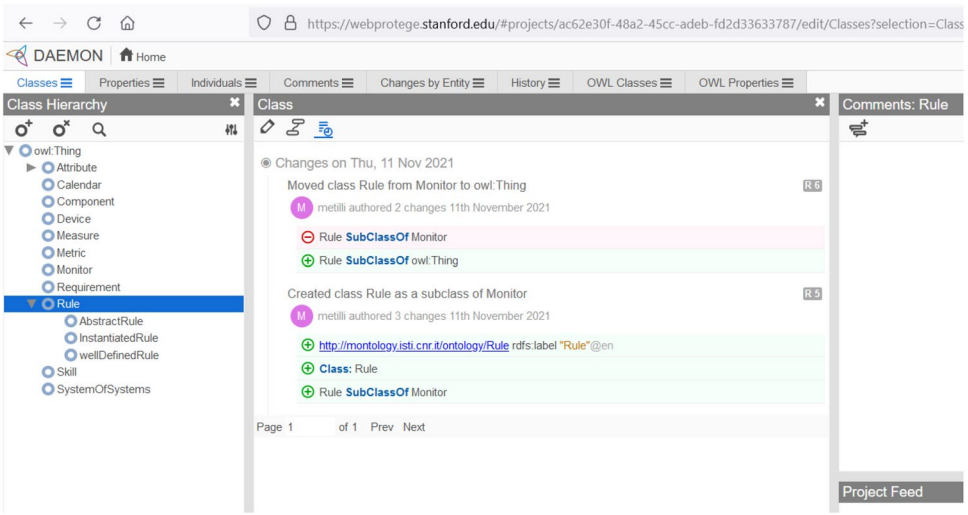

Fig. 12 DAEMON Ontology in WebProtégé

Monitoring System

This component is responsible for monitoring the functional and non-functional properties the IoT system should satisfy. It is supposed to collaborate with the DAEMON Manager to receive the set of properties (rules) to be monitored during the execution. It exposes channels for events on which a CEP listens to analyze them.

DAEMON Manager

This component implements the DAEMON proposal described in "DAEMON Ontology Modules". DAEMON Manager is also responsible for the management of the specification process of the monitoring rules: from the abstract to well-defined and finally to instantiated rules. A detailed description of the DAEMON Manager is provided in the next sections.

DAEMON Manager: Detailed Description

DAEMON Manager is the component responsible for managing and supporting the implementation and the DAEMON Ontology. Thus, its architecture is voluntarily conceived as abstract as possible to instantiate its components with available tools. DAEMON Manager is composed of different components collaborating to achieve a common goal, i.e., providing functionalities and means to share information and knowledge about IoT systems and their components. Figure 11 illustrates the updated standalone supporting architecture allowing the management of the data produced by different *Data Providers* and *DAEMON Builder*, allowing the *DAEMON Final User* to interact with DAEMON Manager for ontology navigation and management. In the following, more details about each component are provided.

DAEMON Mapper

This component is responsible for specifying ontology entities and presenting specific use case domains. Figure 11 shows that the Data Provider is this component's primary target end-user group.

DAEMON Builder Tool

It is used for creating, modifying, and visualizing the ontology according to the representation detailed in "DAEMON Ontology Modules". DAEMON Tool Builder is instantiated with Protégé because it provides a friendly Graphical User Interface (GUI) for defining ontologies; it can be adapted to build even complex ontology-based applications thanks to its modular architecture. The customization of the DAEMON Tool Builder can be used offline as a standalone solution and online as a web-based solution called WebProtégé [2]. This allows a more dynamic sharing of ontologies for collaborative viewing and editing. Figure 12 shows a screenshot of the adopted WebProtégé.

[2] WebProtégé software is available at: https://webprotege.stanford.edu/.

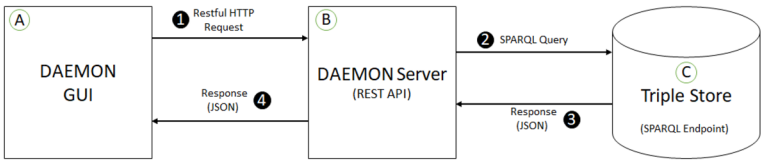

Fig. 13 DAEMON RESTful Server

```
/daemon/sos
/daemon/sos/{id}
/daemon/sos/{id}/environment
/daemon/sos/{id}/system
/daemon/sos/{id}/system/{id}
/daemon/sos/{id}/system/{id}/device
/daemon/sos/{id}/system/{id}/device/{id}
/daemon/sos/{id}/system/{id}/device/{id}/component
/daemon/sos/{id}/system/{id}/device/{id}/component/{id}
```

Fig. 14 DAEMON Server API endpoints for the SoS module

Triplifier

It is a triplifier based on OWL [3] (Web Ontology Language) developed in Java. Triplifier takes the ontology data as input, consisting of the individuals, and it is specified in JSON or XML format. It also takes the rules the DAEMON Builder Tool defines as input to allow the reasoning and inference of new knowledge.

Reasoner

This component is utilized to infer new knowledge, and the reasoning is conducted for consistency tests and inference definition. Openllet is used in the semantic Reasoner's current implementation for the following reasons:

- It is Java-based and readily integrates with the OWL API.
- It is open-source software continuously developed and offers, among other capabilities, the ability to examine the coherence of ontologies.

Triple Store

In the current implementation, GraphDB has been selected as the reference Triple Store, a free-to-use graph database and a tool for knowledge discovery compliant with RDF and SPARQL. It is available as a high-availability cluster. Technically, the DAEMON GUI and the DAEMON End Users interact with the DAEMON Manager by employing well-defined SPARQL queries.

SPARQL Endpoint

This component allows specifying and executing specific SPARQL queries to retrieve knowledge from the triple store and dynamically update the KB content. In the next subsection ("DAEMON Manager: Implementation Details and Exchanged Data"), details about both Triple Store and SPARQL Endpoint are also provided from the behavioral point of view.

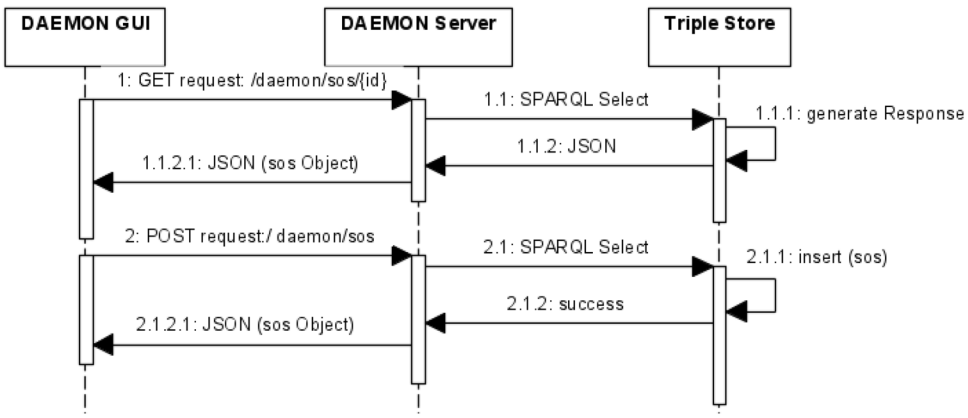

Fig. 15 DAEMON RESTful Server Sequence Diagram

[3] The W3C Web Ontology Language (OWL) is available at: https://www.w3.org/OWL/.

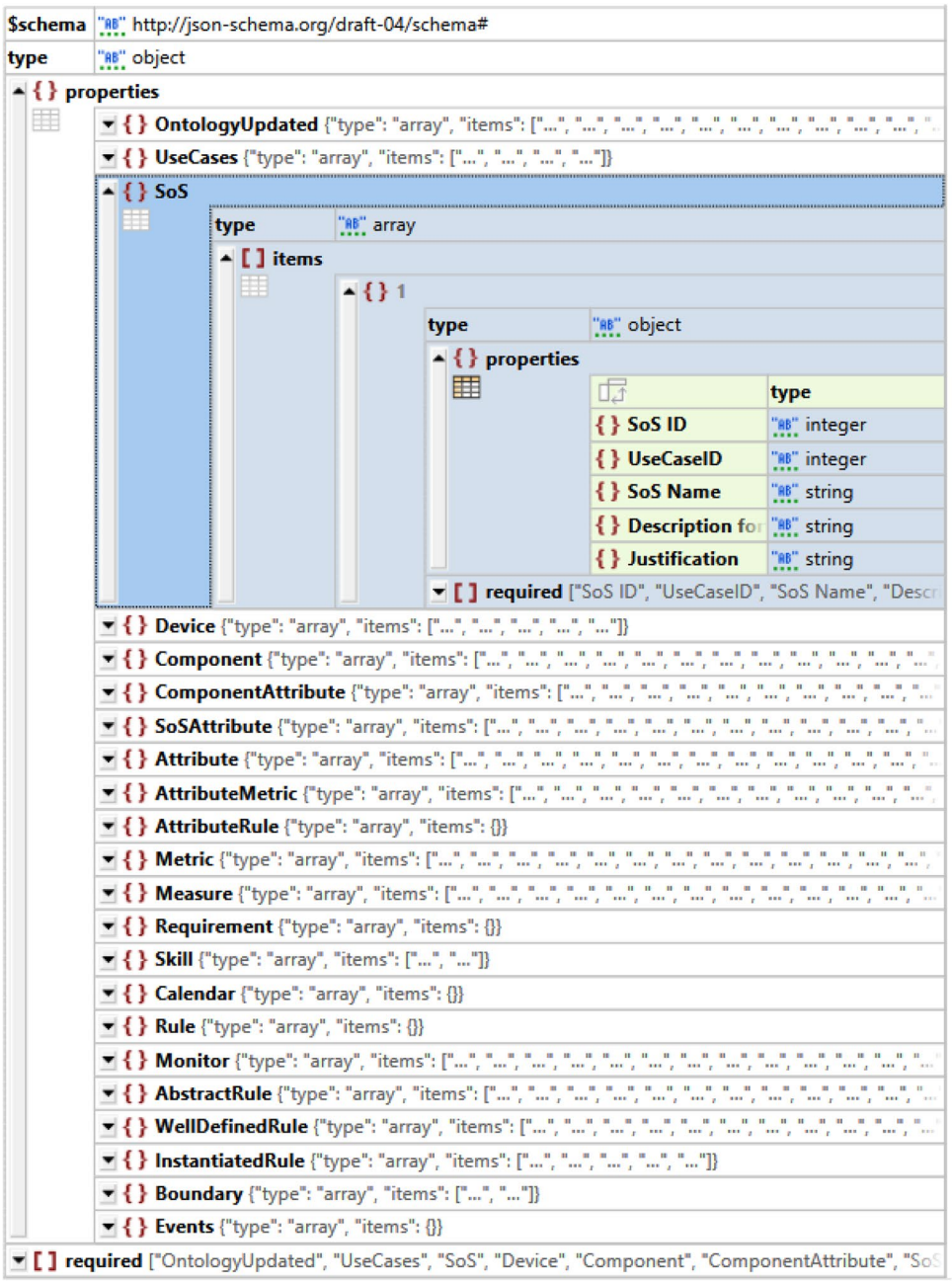

Fig. 16 DAEMON reference JSON schema

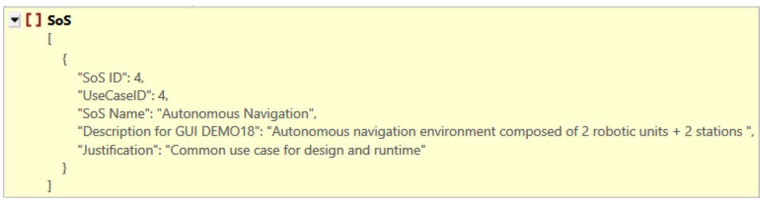

Fig. 17 Extract of an Instance of the DAEMON Reference JSON Schema

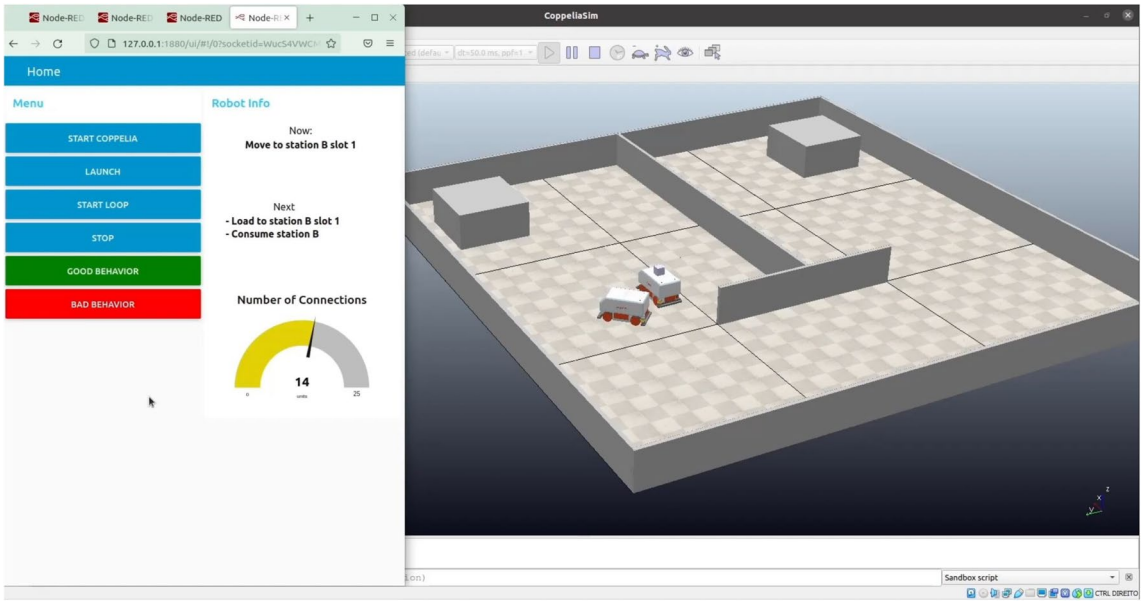

Fig. 18 Coppelia simulation for the use case

DAEMON Manager: Implementation Details and Exchanged Data

In this section, we provide some implementation details of DAEMON Manager by focusing only on managing the interaction and the communication between the DAEMON GUI and the DAEMON Manager component. As shown in Fig. 13, DAEMON provides services for querying/interacting with the ontology. The DAEMON GUI uses the provided information during user interaction. The server handles GET, POST, PUT, and DELETE requests, which are addressed with SELECT/CONSTRUCT, INSERT, UPDATE, and DELETE SPARQL queries, respectively.

The model for the specification of the available endpoints follows the OpenAPI Specification (OAS) .[4] Fig. 14 shows some examples of the proposed endpoints that the DAEMON server API exposes, in this case, for the SoS module of the developed ontology (see "DAEMON Ontology Modules" for more details).

The HTTP operations are defined for each endpoint (path) (GET, POST, PUT, and DELETE). A single path can support more than one operation, and an operation would have one path, except for the GET operation. For instance, 'GET /daemon/sos' returns all available systems of systems in the ontology, whereas 'GET /daemon/sos/{id} returns information about a particular system of systems. For the POST, PUT, and DELETE operations, the 'POST /daemon/sos,' 'PUT /daemon/sos/{id}' and 'DELETE /daemon/sos/{id}', are used, respectively. These last operations must be managed carefully, according to the daemon constraints.

Figure 15 shows the interactions between DAEMON GUI when it performs GET and POST requests for returning a particular System of Systems and inserting a new one. When the Ontology Server receives the requests, it creates the corresponding SPARQL query to retrieve/insert the results from/in the knowledge graph database. The results are retrieved in JSON format.

Exchanged Data Structure

DAEMON Manager is based on RESTful service and communicates with components by exchanging data. To facilitate that communication, we have defined a specific JSON schema.

It represents the data exchanged and allows interoperability with the current (and additional) components integrated within a given IoT system. Figure 16 reports the conceived JSON data structure schema. It highlights the structure of the SoS JSON object, which has, among others, an SoS ID, the SoS Name, and a Justification field containing the reason for choosing that SoS.

[4] API Development for Everyone specification and terminologies are provided at: https://swagger.io/docs/specification/paths-and-operations/.

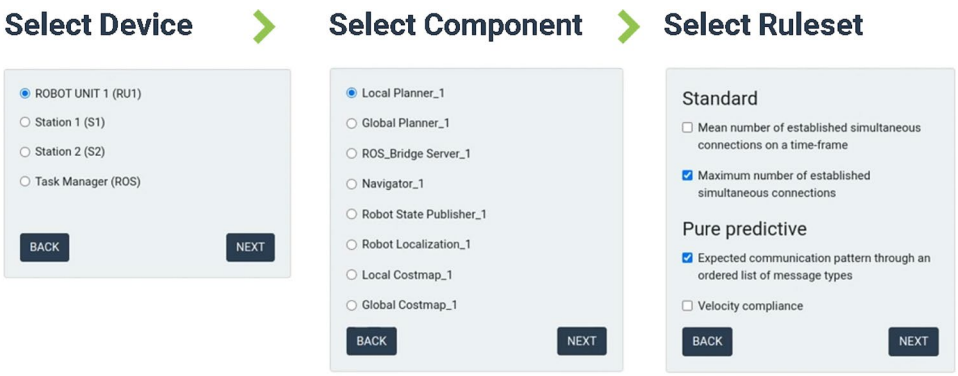

Fig. 19 Graphical user interface panels for the Pre-setup Phase flow. This represents a realization of the DAEMON ontology modules shown in Figs. 2 and 3

Example of Produced Data

All the data produced by DAEMON Manager must comply with the above-mentioned schema. An instance of that schema containing data related to the System of Systems associated with the use case example, described in "An Intralogistics Domain Use Case Scenario", is reported in Fig. 17.

An Intralogistics Domain Use Case Scenario

The present section focuses on describing how a DAEMON instance can be applied to a concrete application scenario, in this case, to support an auditing process within the context of the BIECO project [6] under the H2020 programme. Further details about the technical implementation can be found in [1]. As part of the BIECO project, the present work was instantiated for a multi-robot autonomous navigation scenario applied to the intralogistics domain. In the use case at hand, the execution environment consists of a CoppeliaSim simulation with two autonomous ground vehicles responsible for transporting material between two different workstations, as depicted in Fig. 18.

The DAEMON instance permits real-time observation and verification of the robot's operation and movement, ensuring that the robot can explore the area safely and efficiently. This is especially critical in both dynamic and complex surroundings, as there may be a number of possible obstacles and hazards for the vehicle to avoid, particularly those that involve operating around people. Runtime monitoring makes it possible to identify problems early and take quick corrective action, assuring that the vehicle may continue to operate with safe and trusted behavior at all times.

The following sections describe the DAEMON instance setup and execution for the multi-robot navigation use case in intralogistics. It focuses on the interaction between the four main components (see "Ontology-Based Knowledge Management Architecture"): the DAEMON GUI, the DAEMON Manager, the runtime Monitoring System, and the Expected Behaviour Prediction component.

Monitoring Setup for Autonomous Navigation

During runtime, the DAEMON instance targets the monitoring of functional and non-functional properties of both the robotic system and the execution environment. It involves the interactions between the System of Systems, the execution environment, and the new component or device to be introduced into the system, which in this case is considered to be the local planner component of the robot for navigation. The execution of the DAEMON instance can be carried out in three different situations:

1. Execution environment is simulated, i.e., a simulation model able to represent the real environment. In this case:
   - System or component under evaluation can be simulated or real.
   - Models or stubs can be used to simulate the environment components in which the system or component is executed.
2. Execution environment is running in a test-bed, i.e., it is a representation of the real environment but executed in a test-bed framework to have the possibility to control the internal status of each of its components and to manage violations safely. In this case:
   - The system or component under evaluation is a real component.
   - The execution environment components directly interacting with the component or system under evaluation can be (a) real components, (b) simulated models, or (c) executed using stubs.
3. Execution environment is running in a real context, i.e., the execution environment and its components are executed in a real (operational) environment. In this case:

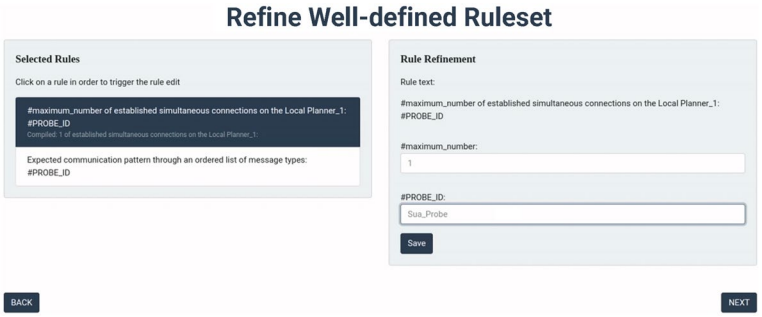

Fig. 20 Graphical user interface panel for the refinement of the well-defined ruleset, implementing the DAEMON ontology modules and process previously presented in Figs. 5 and 7

- The component or system under evaluation is a real component.
- The execution environment components, directly interacting with the component or system under evaluation, are real components.

In this specific case, the system under evaluation is a part of the robot. More specifically, it is the autonomous navigation that was built with the Robot Operating System (ROS) stack, and probes were inserted into the component that manages the local planner.

The user can start the DAEMON execution by interacting with the DAEMON GUI, beginning with selecting the DAEMON instance setup features. The following sections detail the execution of each phase, starting from the first step, i.e., the Pre-setup phase.

Pre-setup Phase

During this phase, the user explores the classification and categorization of the different Systems of Systems, their devices, and components (e.g., execution environment, target system). The Pre-setup Phase involves the collaboration between the DAEMON GUI and the DAEMON Manager components of the previously shown architecture. In particular, the DAEMON GUI provides the user an easy-to-use means for navigating the ontology. In fact, the user selection forces a suitable ontology query to guide the definition of the rules to be used during the monitoring stage.

Once these values are configured, the DAEMON GUI visualizes the components list to the user to let them select the suitable one. As before, through the collaboration between the DAEMON GUI and the DAEMON Manager, the visualization of the specific component skills is provided: connectivity and movement. In this use case, connectivity is the skill considered for experimentation, because it is regarded as the most critical from the security point of view.

Again, the collaboration between the DAEMON GUI and the DAEMON Manager provided the user with the lists of the most suitable functional and non-functional properties for the selected system and execution environment. The properties are presented as high-level specifications and correspond to the ontology abstract rules. According to the ontology representation, the abstract rules are classified as standard rules, i.e., non-functional properties that can be assessed through the Monitoring System, and Pure Predictive rules, i.e., functional properties that can be predicted using the Expected Behaviour Prediction component. An example of such a rule is, for instance, the "Maximum number of established simultaneous connections", which targets the mutual interaction between Local Planner_1 and Autonomous Navigation. Figure 19 provides an overview of these interfaces.

The rule boundaries can be established either using data collected during the design and development stages or provided by the user. In both cases, the values are managed through the ontology and remain valid for all runtime execution. Considering the pure predictive rules instead, one of the properties considered is the "Expected communication pattern through an ordered list of message types." This rule is visualized as a DAEMON Manager query and focuses on the behavior of the Local Planner_1. It requires that the predictive engine forecasts the specific message order that would be expected. The last interaction between the DAEMON GUI and the DAEMON Manager concerns the definition of boundaries. Indeed, if not provided by the Blueprint data analysis, the user needs to insert the boundaries for the selected abstract rules as the last step. This concludes this first part of the pre-setup and starts preparing the following steps described in the upcoming sections.

Offline Activities

When the initial Pre-setup Phase ends, the interaction between the DAEMON Manager and the DAEMON GUI provides the user with downloadable artefacts helpful in preparing for the following monitoring activities. More specifically, these artefacts include a jar file of the executable probe and a set of guidelines helpful in instrumenting the code of the target component.

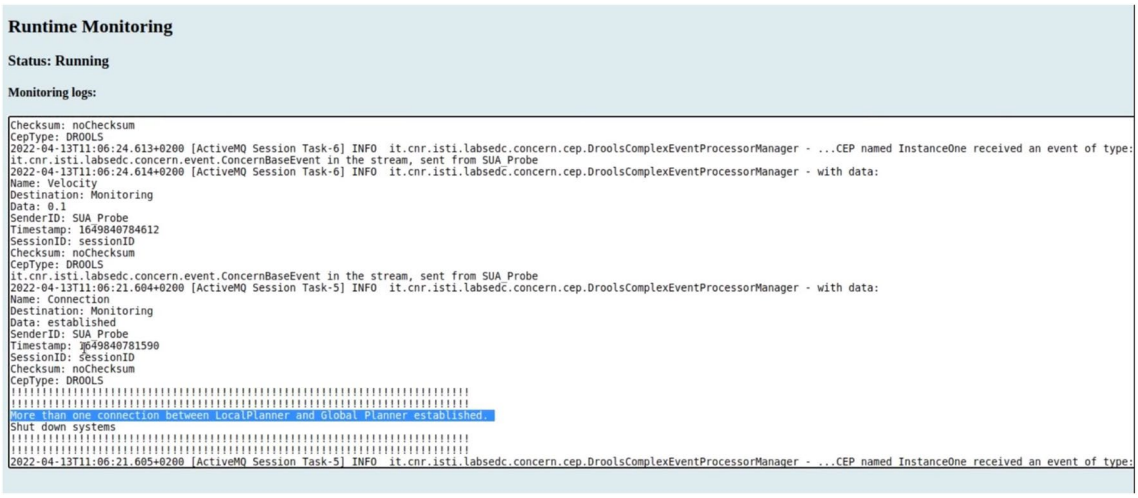

Fig. 21 Runtime Monitoring Logger: Trace of rule violation raised

Using this above information, the user can work offline to instrument the Local Planner_1 and execution environment with suitable probes.

Finish Pre-setup

The user can finalize the auditing activity pre-setup phase through DAEMON GUI by refining the well-defined rules. This activity involves the DAEMON GUI and the DAEMON Manager components. It focuses on the identifiers of the probes injected into the target component and the execution environment. An example for this use case is provided in Fig. 20. In this case, for the rule "Maximum number of established simultaneous connections," the user inserts the identifier "Sua_Probe."

Once finalized, the well-defined rules are translated into instantiated rules and provided to the Monitoring System for execution. This finally concludes the complete Pre-setup Phase and the control goes back to the DAEMON GUI to start the core of the monitoring activity.

Start DAEMON Monitoring

As soon as the user issues the start command from the DAEMON GUI, the DAEMON instance passes from the "online" state to the "running" status. In this state, the Monitoring System raises the Complex Event Processor (CEP) and accepts the instantiated rules defined in the Pre-setup Phase. Then, it compiles the rules into meta-rules and Instantiated rules. At this point, the monitoring starts listening to the events sent by the probes on a dedicated channel. In parallel, listening from the same channel, the Expected Behaviour Component, if previously activated, receives the events useful for the forecasting models. These are executed faster than the normal execution environment and can, therefore, provide predictions about trusted behavior. Considering, for instance, an "Expected communication pattern through an ordered list of message types" rule, a forecast event containing a prediction regarding a velocity and trajectory score event sequence can be monitored. Using the data of the forecast event, the monitor instantiates the meta-rule into a new rule and injects it into the CEP. The boundary value for the period validity of the new rule is also provided. For this use case, the boundary is set to 5 s, as established in the forecast event.

On the Monitoring System side, it continuously receives the Sua_Probe events containing Score, Velocity, and connection status and checks the instantiated rules. Expected Behaviour Prediction and Monitoring components continue collaboration until the user decides to stop the auditing activity or as soon as a rule violation is experienced.

Validation Scenario

In the use case execution, a malicious code attack has also been simulated for validation purposes. Thus, a malicious code injection has been performed through the UI shown in Fig. 18. As depicted, this caused an increase in the connections between the Local planner_1 and the Global planner. Consequently, through its monitoring, the DAEMON instance observed the violation of the "Maximum number of established simultaneous connections" rule, immediately notified the user, and triggered the associate countermeasure. If the DAEMON activity stops, the system's dynamic configuration can be performed offline. This detection is shown in the runtime monitoring logs provided in Fig. 21.

Once the system had been reconfigured, another round of DAEMON execution was performed, verifying the system returned to a safe and trusted condition.

Conclusions

This paper focuses on the advancements (technological and research) concerning previous work [1]. More precisely, it presents the improvements made to the DAEMON framework, which integrates knowledge about the IoT/SoS and monitoring domains from the cybersecurity perspective into a single ontology, the implementation of the proposed framework and its components, and the validation through its instantiation in a multi-robot autonomous navigation scenario applied to the intralogistics domain, one of the BIECO project's Use Cases .[5]

As future general works, the following activities will be considered:

- Implementation of the final version of the proposed framework and its components.
- Validation of the framework with all the BIECO Use Cases and considering also other contexts and IoT systems, such as Smart Cities and Healthcare Sectors.

As specific future works, the following will be considered:

For the Runtime Monitoring, we are planning to include features for using smart agents instead of the proposed probes. The new probe should be capable of sending formal events, receiving notifications from the CEP, and activating countermeasures. Therefore, the probe could be used to change its host's behavior while running to reduce the risk of the detected violation. Solutions to be analyzed are lowering a transmission rate to avoid collision or congestion or executing an alternative activity during the system running.

For DAEMON Manager, the planned future work concerns the finalization of the components' implementation by identifying the alternative open-source tools to be used and customized. This activity will also consider interoperability and extensibility to improve the overall implementation performance.

Acknowledgements This work was partially supported by the EU H2020 BIECO project Grant Agreement No. 952702, by the Portuguese FCT program, Center of Technology and Systems (CTS) UIDB/00066/2020 / UIDP/00066/2020, and by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

[5] More information about BIECO Use-Cases can be found at: https://www.bieco.org/use-cases/.

Declarations

Conflict of Interest Authors declare no conflict of interest exists.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Daoudagh S, Marchetti E, Calabrò A, Ferrada F, Oliveira AI, Barata J, Peres R, Marques F. An ontology-based solution for monitoring iot cybersecurity. In: Camarinha-Matos LM, Ribeiro L, Strous L (eds.) Internet of Things. IoT Through a Multi-disciplinary Perspective, 2022;p. 158–76. Springer, Cham.
2. Fortino G, Savaglio C, Spezzano G, Zhou M. Internet of things as system of systems: a review of methodologies, frameworks, platforms, and tools. IEEE Trans Syst Man Cybern. 2020;51(1):223–36.
3. Rastogi V, Srivastava S, Mishra M, Thukral R. Predictive maintenance for sme in industry 4.0. In: 2020 Global Smart Industry Conference (GloSIC), 2020;382–90.
4. Calabrò A, Daoudagh S, Marchetti E. MENTORS: Monitoring Environment for System of Systems. In: Proceedings of the 17th International Conference on Web Information Systems and Technologies - WEBIST, 2021; p. 291–8. SciTePress, Portugal. https://doi.org/10.5220/0010658900003058. INSTICC.
5. Sforzin A, Bobba R et al. D5.4-requirements analysis of demonstration cases phase 2. 2021. https://cybersec4europe.eu/publications/deliverables/.
6. Ricardo Silva Peres et al. The BIECO conceptual framework towards security and trust in ict ecosystems. In: Testing Software and Systems, Cham, 2022;230–2.
7. Skouloudi RC, Malatras A, Dede G. Guidelines for securing the internet of things. European Union Agency for Cybersecurity 2020.
8. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation). Official J Eur Union. 2016;L119:1–88.
9. Maier P, Ma Z, Bloem R. Towards a secure scrum process for agile web application development. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. ARES '17. Association for Computing Machinery, New York, NY, USA 2017. https://doi.org/10.1145/3098954.3103171.
10. Bartolini C, Daoudagh S, Lenzini G, Marchetti E. GDPR-based user stories in the access control perspective. In: International Conference on the Quality of Information and Communications Technology, 2019;3–17. Springer
11. DRIDI CE, BENZADRI Z, BELALA F. System of systems modelling: Recent work review and a path forward. In: 2020 International Conference on Advanced Aspects of Software Engineering (ICAASE), 2020;1–8.

12. Nilsson R, Dori D, Jayawant Y, Petnga L, Kohen H, Yokell M. Towards an ontology for collaboration in system of systems context. INCOSE Int Sympos. 2020;30:666–79.
13. Langford G, Langford T. The making of a system of systems: ontology reveals the true nature of emergence. In: 2017 12th System of Systems Engineering Conference (SoSE), 2017; 1–5. IEEE
14. Walden DD, Roedler GJ, Forsberg K, Hamelin RD, Shortell TM. International council on systems engineering. Systems engineering handbook: a guide for system life cycle processes and activities 2015.
15. Franzén LK, Staack I, Jouannet C, Krus P. An ontological approach to system of systems engineering in product development, 2019;35–44.
16. Arp R, Smith B, Spear AD. Building ontologies with basic formal ontology. Cambridge: The MIT Press; 2015. https://doi.org/10.7551/mitpress/9780262527811.001.0001.
17. Bajaj G, Agarwal R, Singh P, Georgantas N, Issarny V. A study of existing ontologies in the IoT-domain. arXiv preprint. 2017. arXiv:1707.00112.
18. Mascardi V, Cordì V, Rosso P. A comparison of upper ontologies. In: Woa, vol. 2007, 2007;55–64. Citeseer
19. Partridge C, Mitchell A, Cook A, Sullivan J, West M. A survey of top-level ontologies—to inform the ontological choices for a foundation data model. University of Cambridge; 2020.
20. Lynch K, Ramsey R, Ball G, Schmit M, Collins K. Conceptual design acceleration for cyber-physical systems. In: 2017 Annual IEEE International Systems Conference (SysCon), 2017;1–6. https://doi.org/10.1109/SYSCON.2017.7934771
21. W3C: Semantic Sensor Network Ontology 2017.
22. Gyrard A, Bonnet C, Boudaoud K, Serrano M. Lov4iot: a second life for ontology-based domain knowledge to build semantic web of things applications. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 2016;254–61.
23. oneM2M Partners Type 1: Base ontology. Technical report. 2019. https://www.onem2m.org/images/pdf/TS-0012-Base_Ontology-V3_7_3.pdf.
24. Bermudez-Edo M, Elsaleh T, Barnaghi P, Taylor K. Iot-lite: A lightweight semantic model for the internet of things. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016;90–7. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0035
25. Won M. Intelligent traffic monitoring systems for vehicle classification: a survey. IEEE Access. 2020;8:73340–58.
26. Fotescu R-P, Constantinescu R, Alexandrescu B, Burciu L-M. System for monitoring the parameters of vehicle. In: Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies X, vol. 11718, 2020;55–61. SPIE.
27. Hidayanti F. Design and application of monitoring system for electrical energy based-on internet of things. Helix. 2020;10(01):18–26.
28. Santos MA, Munoz R, Olivares R, Rebouças Filho PP, Del Ser J, de Albuquerque VHC. Online heart monitoring systems on the internet of health things environments: a survey, a reference model and an outlook. Inf Fusion. 2020;53:222–39.
29. Bhamare D, Zolanvari M, Erbad A, Jain R, Khan K, Meskin N. Cybersecurity for industrialcontrol systems: a survey. Comput Secur. 2020;89: 101677.
30. Khan S, Nazir S, García-Magariño I, Hussain A. Deep learning-based urban big data fusion in smart cities: towards traffic monitoring and flow-preserving fusion. Comput Electric Eng. 2021;89: 106906.
31. Burns M, Griffor E, Balduccini M, Vishik C, Huth M, Wollman D. Reasoning about smart city. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), 2018;381–6.
32. Bhatt SN, Manadhata PK, Zomlot L. The operational role of security information and event management systems. IEEE Secur Priv. 2014;12(5):35–41.
33. González-Granadillo G, González-Zarzosa S, Diaz R. Security information and event management (siem): analysis, trends, and usage in critical infrastructures. Sensors. 2021. https://doi.org/10.3390/s21144759.
34. Holubová I, Vavrek M, Scherzinger S. Evolution management in multi-model databases. Data Knowl Eng. 2021;136: 101932.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.