An Ontology-based Solution for Monitoring IoT Cybersecurity

Said Daoudagh[1][0000-0002-3073-6217], Eda Marchetti[1][0000-0003-4223-8036], Antonello Calabrò[1][0000-0001-5502-303X], Filipa Ferrada[2,3][0000-0002-9328-852X], Ana Inês Oliveira[2,3][0000-0003-1716-846X], José Barata[2,3][0000-0002-6348-1847], Ricardo Peres[2,3][0000-0003-3777-1346], and Francisco Marques[2][0000-0003-0409-6380]

[1] ISTI-CNR, Pisa, Italy
{said.daoudagh, eda.marchetti, antonello.calabro}@isti.cnr.it
[2] Uninova Institute, Centre of Technology and Systems (CTS), Portugal
[3] NOVA School of Science and Technology, FCT-NOVA, Caparica, Portugal
{faf, aio, jab, ricardo.peres, fam}@uninova.pt

Abstract. *Context:* Systems of Systems (SoSs) are becoming an emerging architecture, and they are used in several daily life contexts. *Objective:* The aim is to define a reference environment conceived for monitoring and assessing the behavior from the cybersecurity point of view of SoS when a new IoT device is added. *Method:* In this paper, we propose the Domain bAsEd Monitoring ONtology (DAEMON), an ontology that formally models knowledge about monitoring and System of Systems (SoS) domains. We also conceived a reference supporting architecture, and we provided the first proof-of-concept by implementing different components. *Results and Conclusion:* For the feasibility purpose, we have validated our proof-of-concept in the context of the EU BIECO project by considering a Robot Navigation use-case scenario.

Keywords: Cyber Security · Internet of Things (IoT) · Monitoring · Ontology · System of Systems (SoS).

1 Introduction

Nowadays, most ICT systems and applications rely on the integration and interaction with other (third parties) components or devices because it is a valid means for increasing productivity and reducing, at the same time, the overall development costs. However, even if effective, this practice can expose the ICT systems to high risks regarding security, privacy, and safety. Additionally, in most cases, there are difficult and costly procedures for verifying if these solutions have vulnerabilities or if they have been built, taking into account the best security and privacy practices. However, detecting vulnerabilities accurately in ICT components and understanding how they can propagate over the supply chain is extremely important for the ICT ecosystems. To find a suitable and effective compromise, one commonly adopted solution for vulnerability detection is using a monitoring system, i.e., a means for the online analysis of functional and non-functional properties. Usually, this system relies on the collection of events produced by the systems, devices, or components during the execution and uses complex event patterns for assessing a specific property. Indeed, the patterns are associated with observable normal (or abnormal) behavior and can be exploited to raise alarms or implement countermeasures promptly. Nevertheless, even if notably efficient and effective, a monitoring system's design, implementation, and management can be an effort and time-consuming activity. In this paper, according to the definition provided in [18] that considers the IoT system as "the latest example of the System of Systems (SoS), demanding for both innovative and evolutionary approaches to tame its multifaceted aspects" we focus mainly on the System of Systems (SoS). Monitoring SoS involves all the stages of the software development process and different stakeholders, such as SoS domain experts, device developers, or monitoring experts. The complexity of the monitoring activity could increase when new devices (or components) are dynamically included in the SoS environments [35].

To overcome this issue, in [14] some of the authors of this paper presented an initial solution. It focused on the definition of a common framework for collecting together, in a manageable and user-friendly way, the knowledge coming from different sources: SoS domain experts, standards, guidelines, monitoring, and developers experts. The purpose was to join concepts and definitions about the SoS and monitoring into a unique manageable ontology-based representation [14]. In this paper, we leverage this recent proposal by:

1. extensively modifying the initial core ontology (i.e., MONTOLOGY) and deriving a new one called Domain bAsEd Monitoring ONtology (DAEMON). In particular, we introduce modules that make the ontology more manageable and comprehensive, and we add new concepts to the modules to better represent the monitoring of a SoS knowledge;
2. we customize the proposed reference architecture by revising the components' interaction and roles and introducing new ones for better managing the new elements;
3. we validate the proposed customization using a real case study provided within an ongoing European project. In particular, we consider the Multi-Robot Navigation use case scenario and its specific set of functional and non-functional properties.

Outline. Section 2 discusses the related works concerning the ontologies, cybersecurity specification and vulnerabilities in the context of SoSs, and Monitoring systems. We introduce DAEMON ontology in Section 3 by describing its main modules, concepts and the relationship between them. Section 4 describes DAEMON's reference architecture and how the components interact. We illustrate, in Section 5, the validation of both DAEMON and its reference architecture through a Multi-Robot Navigation use-case scenario within an ongoing EU Project. Section 6 concludes the paper by also highlighting our current and future works.

2 Related Works

Ontology-based System of Systems (SoS). In recent years, much research has been conducted on modeling System of Systems (SoS). Dridi et al. [16] classified SoS modeling into seven main classes: Model-Driven Architecture, Model-Driven, Services-Oriented Architecture, Ontology, Architecture Description Language, Bigraph, and Hybrid. To properly model SoS, one cannot separate their inherent engineering processes, which involve planning, analyzing, organizing, and integrating constituent/component systems (CS), i.e., the System of Systems Engineering (SoSE). In this context, ontology-based approaches for modeling SoS are needed to establish domain concepts and link the SoS processes consistently using common language and semantics, which are essential in the planning and analysis processes [16, 31, 25, 40, 20]. Ontologies can be used to deal with the interoperability and scalability of SoS, supporting the creation of new domain ontologies as well as the reusability of existing ones. A top-level ontology/upper ontology, such as BFO, DOLCE, among others [3, 5, 29, 34], is a highly general representation of categories and relations common to all domains. In addition, efforts have been made to create meta-models for representing SoS/SoSE ontologies: Dridi et al. [16] use a model or set of models to document and communicate from the system requirements level down to the software implementation level, Nilsson et al. [31] proposed an ontology for SoS that uses Object Process Methodology (OPM) ISO 19450 to facilitate collaboration among organizations with focus on safety aspects, Baek et al. [4] developed a conceptual meta-model for representing SoS ontology and Langford et al. [25] created a framework that embraces ontology of systems and SoS to expose the true nature of emergence. On the other side, ontologies might be domain-specific intended to describe individual systems or domains of interest. In response to this, some work has been done to develop ontologies for SoS/SoSE in different domains [20, 27].

In light of this, Internet-of-Things (IoT) systems can be engineered from the perspective of SoS. IoT applications involve the integrated operation of many subsystems, or constituent systems (CS) that are physically and functionally heterogeneous maintaining their advanced cyber-physical functionalities. Similarly, it is important to develop ontologies to share semantic information between IoT subsystems. In response to this challenge, in recent years several proposals for ontology have emerged from the semantics and IoT research communities aiming at describing concepts and relationships between different entities. In 2009, Scioscia and Ruta proposed to use the technologies of semantic web with IoT and developed the semantic web of things (SWoT) [37]. The W3C Semantic Sensor Network Incubator Group has developed the SSN ontology [4], to describe the sensors, observations, and related concepts in the sensor network. Gyrard et al. proposed the M3 Ontology [21] framework that assists users in reusing the domain knowledge and interpreting sensor measurements to build IoT applications. The oneM2M base ontology [33], developed within the oneM2M global open standard for M2M communications and the IoT, is a top-level ontology

[4] The W3C SSN is available at: https://www.w3.org/TR/vocab-ssn/

specifying the minimal ontology that is required such that other ontologies can be mapped into oneM2M as the example of Smart Appliances Reference Ontology (SAREF) [15]. More recently, Bermudez-Edo et al. proposed the IoT-Lite ontology [8] which is a light ontology that represents the resources, entities and services of the IoT. It is an instantiation of the W3C SSN Ontology and is also a base ontology that can be extended to represent IoT concepts in a more detailed way in different domains. A comprehensive ontology catalog is available online and maintained by LOV4IoT (Linked Open Vocabularies for IoT) [26], aiming at encouraging the reuse of domain knowledge already designed and available on the WWW.

The DAEMON ontology, proposed in this paper, aims at representing the monitoring of SoS knowledge. It leverages the previous MONTOLOGY by extending and introducing new concepts useful for a better knowledge representation, such as the rule hierarchy and skill (see Section 3). DAEMON is specifically conceived for being the connection between SoS and the monitoring ontologies, that in the best of our knowledge is still not yet offered. Nevertheless, SoS ontologies like IoT-Lite or oneM2M Base Ontology can be integrated and reused in our proposal. DAEMON intention is not to substitute but use integrate exiting knowledge. Therefore, the SoS module (see Section 3) aims at representing the point of integration where the different ontologies concepts can be used and integrated into DAEMON.

Cybersecurity Specification and Vulnerability. In literature, several comprehensive sets of functional and non-functional (including security, safety and privacy) requirements that can be used for guiding research, technology development, and design are currently available. These sets can be found in recent European Projects documentations, such as [38, 1], or standards and specifications such as [39, 17], or available backlog list containing structured security and privacy user stories [7]. These available heterogeneous sources, while very useful from a cybersecurity and vulnerability specification point of view, are generally kept generic and domain-specific agnostic. The proposal presented in this paper intends to provide a methodology for collecting and organizing together the generic and specific knowledge about a target application domain into a unique reference ontology. The aim is to focus only on the most suitable functional and non-functional properties of each specific context to better focus the monitoring and assessing activity on the expected behavior of IoT/SoS/Ecosystem/Components. In Section 5, a specific example in the robotic domain is provided.

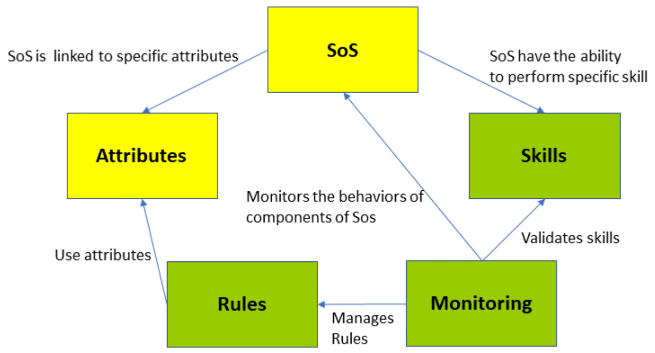
Monitoring Systems. Monitoring systems have been applied in several domains such as: traffic [41], automotive [19], avionic [22], healthcare [36], industry [10]. In almost all the application contexts, the existing monitoring proposals aim to : i) providing a powerful, concise and unambiguous specification language for the validation properties specification [24]; ii) defining mechanisms for the conformity assessment of the system against the selected properties [13].
Recently, the massive and extensive usage of the internet promotes the adoption of monitoring approaches able to mitigate the risk of cyber-attacks. Among them

the most promising solutions are: *Security Information and Event Management* (SIEM), *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 violation occurred within the system [11] whereas, XDR and EDR can boost the SIEM analysis providing a set of information and tools that will enhance the analysis executable through the SIEM. Usually, the collected data are stored in *Data Lake* [23] 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, we provide 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) instantiating the selected properties into monitoring rules able to capture, infer and analyze complex events; (3) allowing the detection of critical problems, failures, and security vulnerabilities; (4) validating the trust and security level of the run-time behavior of SoS and its devices or components; (5) rising warnings and enabling (non-blocking) system reconfiguration to assure a trustworthy execution.

3 DAEMON Ontology

The aim of Domain bAsEd Monitoring ONtology (DAEMON) is to help the different SoS stakeholders gather functional and non-functional properties related to the different part of SoS, and consequently enabling the definition of concrete monitoring rules each related to a specific property. As a result, we can define a reference set of meaningful rules to be monitored during the SoS execution so as to automatically demonstrate the compliance (non-compliance) with the selected properties.


SoS
SoS is linked to specific attributes
SoS have the ability to perform specific skill
Attributes
Skills
Monitors the behaviors of components of Sos
Use attributes
Validates skills
Rules
Monitoring
Manages Rules
Fig. 1: DAEMON Ontology Modules.

The starting point of DAEMON ontology is the MONitoring onTOLOGY (MONTOLOGY) ontology [14], which models the SoS and monitoring main concepts and defines the relationships between them. MONTOLOGY is a core component of MENTORS (i.e., a monitoring environment for SoS) [14] and it is composed of two main modules: System of Systems (SoS) Module (containing eight concepts) and Monitoring Module (which contains five concepts), with a total of 13 (thirteen) concepts.

Therefore, the basic idea of DAEMON is to extend MONTOLOGY by adding new concepts, and to reorganize the content into a more manageable and modular way, so as to enable interoperability and facilitate both the extensibility and maintainability. More precisely, as reported in Figure 1, we divide DAEMON in five modules: (1) SoS (Figure 2); (2) Attributes (Figure 3); (3) Skills (Figure 4); (4) Rules (Figure 5); and (5) Monitoring (Figure 8).

The remainder of this section provides more details about how we derived DAEMON by reusing the most relevant parts of MONTOLOGY and how we reconstructed the new content in a more comprehensive set of modules. For the aim of readability, in the following figures, we report the new concepts/classes introduced in DAEMON by coloring the shape outline in red.

SoS Module. The SoS module aims at representing the most relevant concepts related to the System of Systems (SoS) domain and the relationship between them. Differently from MONTOLOGY, we model the *SystemOfSytems* as a composition of *System*, and it is influenced by a specific *Environment* in which it operates and it is executed. Therefore, a *System* is a collection of *Devices* that represent the object of the monitoring activities. As in MONTOLOGY, each *Device* is composed of a specific set of *Components*.

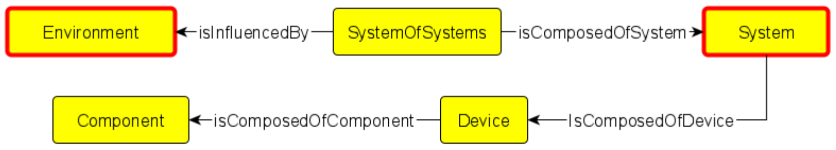


Fig. 2: System of Systems (SoS) module.

Attributes Module. An *Attribute* is a functional and non-functional property related to a specific SoS concept. Examples of attributes could be (1) the communication latency between the components; (2) the average amount of the messages is under a certain level, so as to avoid or detect DoS attacks; or (3) the number of the allowed/authorized connections.

Therefore, this module contains all the concepts related to the observable properties of the concepts in the SoS module. As in Figure 3, we extend this module with the two specific concepts; *QualititaveAttribute*, and *ObservableAttribute*, which is a quantitative attribute used for defining both the *Measure* and *Metric* used for defining monitoring rules [5]. We also expand the *Attribute* hier-

[5] Note that, in MONTOLOGY *Measure* and *Metric* are directly connected with the *Attribute* class.

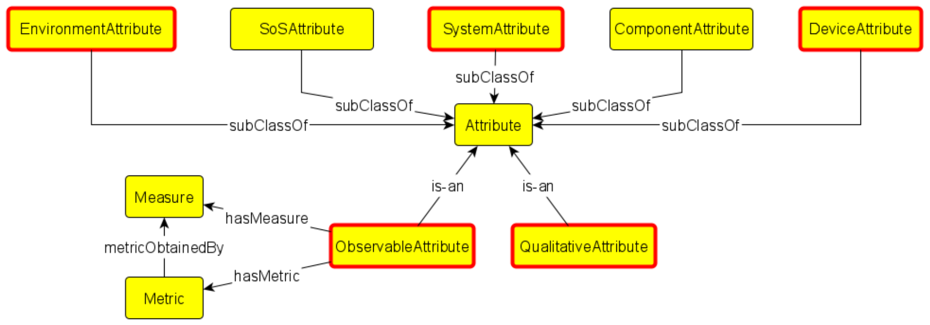


Fig. 3: Attributes Module.

archy by adding three sub-classes: *EnvironmentAttribute*, *SystemAttribute* and *DeviceAttribute*. The purpose is to have specific attributes for each of the SoS's concepts/classes, so as to enable the monitoring of their behavior throughout specific monitoring rules.

Skills Module. The skills module allows modeling the skills related to the different concepts in SoS module. A *Skill* represents an ability of an agent (active or passive) to perform a specific action, such as the ability of connection or the ability of movement. The original concept of *Skill* modelled in MONTOLOGY has been extended into two different ways. Firstly, we create a skill hierarchy by leveraging the original concept *Skill* as super-class of the hierarchy, and we add two specific sub-classes (*BasicSkill* and *ComplexSkill*) that are connected with

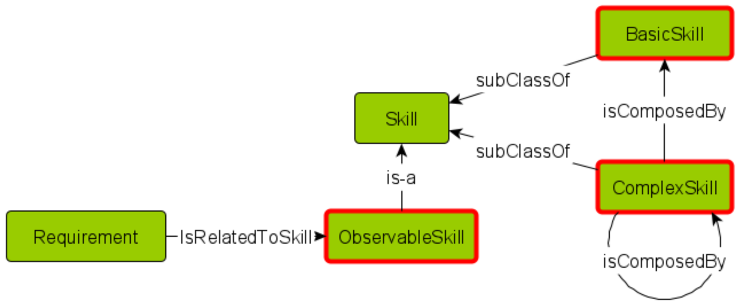


Fig. 4: Skills Module.

each other through the relation *isComposedBy*. As in the Figure, a *ComplexSkill* can be composed both through a set of *BasicSkill*, or/and iteratively throughout a set of *ComplexSkill*. Secondly, we introduce the concept of *ObservableSkill*, i.e., the observed ability related to the SoS concept that can be validated through the monitoring facilities. Differently from MONTOLOGY, we connect the *Requirement* class directly to *ObservableSkill* through the *isRelatedToSkill* association. Therefore, each *ObservableSkill*, specified as a set of *Requirements*, can be verified through a specific *Rule*.

Rule Module. This module contains all the concepts related to rule at different level of specification. A rule is a set of instructions related to analysis of the occurrences of one or more events in a stream or a cloud of events. Usually, rules are structured as a set of if-then-else sequences. In particular, DAEMON leverages the concept of rule by providing a well-formed hierarchy with the following subclasses (see Figure 5): AbstractRule, WellDefinedRule, and InstantiatedRule.

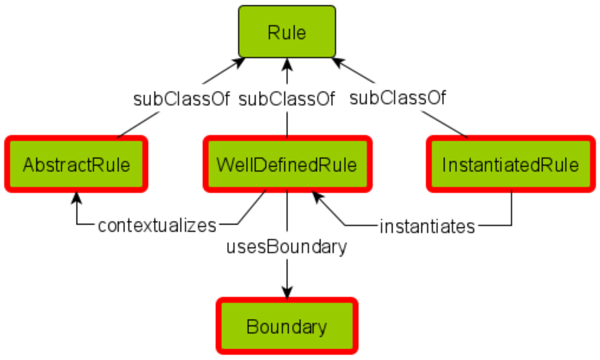


Fig. 5: Rule Module.

AbstractRule points out a rule that is generic, not yet instantiated within the execution context and it has been simply gathered from the navigation of the ontology. *WellDefinedRule* refers to a rule ready for being translated to the destination language of the Complex Event Processor and related to the monitoring of a specific device. It is also expressed in terms a set of boundaries (see *Boundary* concept in Figure 5) that contains specific values that express the applicability ranges of the rule. The last case is related to the *InstantiatedRule*, which is a rule written using the language understandable by the monitoring engine.

To better clarify the complexity of the process involved in obtaining a processable rule, in Figure 6 we report a graphical representation of the evolution of the rule: from an abstract to an instantiated one.

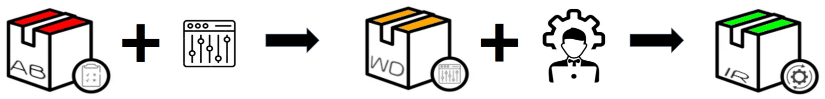


Fig. 6: Rules Transformation Process.

In particular, an abstract rule is a very generic natural language description of the objective of the auditing activity that is easily understandable by non-expert users. For instance, the maximum number of established simultaneous connections between two components. The abstract rule is then refined into the well-defined rule, that is a semi-structured and implementable rule, where the

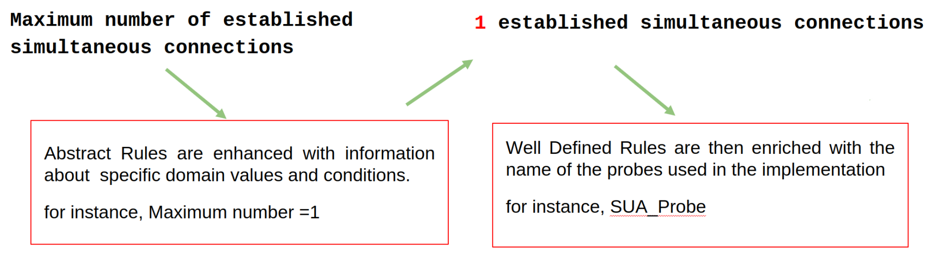

Maximum number of established simultaneous connections
1 established simultaneous connections
Abstract Rules are enhanced with information about specific domain values and conditions.
for instance, Maximum number =1
Well Defined Rules are then enriched with the name of the probes used in the implementation
for instance, SUA_Probe

Fig. 7: From Abstract to Well-Defined Rule Enrichment Process.

users need to add few specific details about the context. For instance, the maximum number of established simultaneous connections. In Figure 7, we report an example of abstract and well-defined rules.

Finally, the well-defined rule, enriched with the name of the probes used by the user, will be automatically translated into an instantiated rule according to the monitoring language used. This will be used by the run-time monitor during the auditing framework execution.

A typical structure of an instantiated rule can be summarized as follows:

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

It contain one or more rules that define at a minimum the rule conditions (when) and actions (then).

Monitoring Module. Monitoring Module aims at modeling monitoring concepts and relationships between them. The core class of the Monitoring module is the *Monitor*, which observes rules organized in *Calendar*, i.e., an ordered set of rules. Each *Calendar* is able to validate a specific *ObservableSkill* at run-time defined in the Skills module. The *Monitor* has a specific *EntryPoint* that is used to communicate with the *Probe*.

A *Probe* is a piece of software code, that can be injected into the observed/-monitored component, device, or system, and is capable of sending *Events* according to a specific format.The probes can send events at regular intervals or every time a specific situation occurs. The sent events contain information related to the occurrence of actions on the observed SoS entity.

The term *Event* defines the change of a state within a system. This change of state is generated when a function is invoked within the system under auditing. The injected *Probe* will pack this atomic action into an event and notify it to the *Monitor* for executing the processing action on the event stream. For being correctly managed by a concrete monitor, the event should contain several pieces of information needed for analyzing a snapshot of what is happening within the System Under Test.

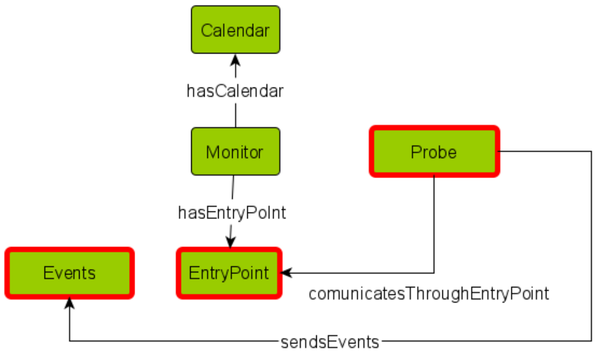


Fig. 8: Monitoring Module.

4 Reference Architecture

In this section, we present the reference architecture of the DAEMON methodology introduced in the previous section. This architecture revises the proposal of [14] by introducing new components, interaction and actors. The new proposal is schematized in the Figure 9 and described as in the following:

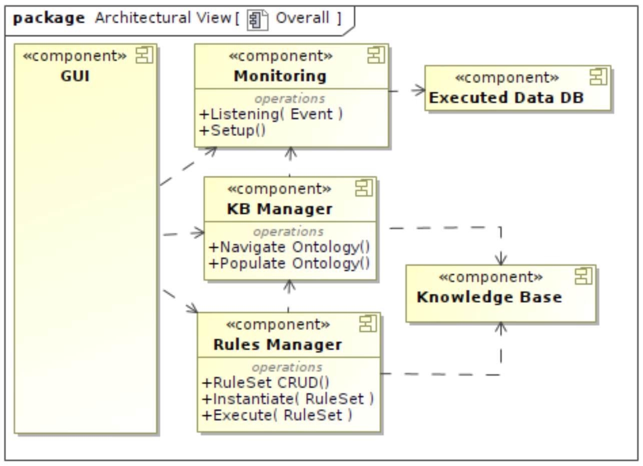


Fig. 9: DAEMON Reference Architecture.

GUI. This component provides the graphical user interface and the DAEMON facilities to different stakeholders such as: *Business Manager*, who is in charge of the selection and refinement of the functional and non-functional properties and the management of the possible notification of violations; the *Rules-Maker* who is in charge of providing an implementation of the well-defined rules into the target monitoring language; *Ontology Expert* who is the responsible of the ontology management that includes its conceptualization, implementation and maintenance; and the *Developer Expert* in charge of instrumenting the code with probes.

Monitoring. This component is the core of the analysis of the system execution. It relies on an event-driven publish-subscribe architecture. Different monitoring tools can be used for implementing this component as mentioned in Section 2. One of the main sub-component is the Complex Event Processor (CEP) [2], that is in charge of inferring simple and complex pattern according to well-defined rules selected by the DAEMON stakeholder. Through the operation called *Setup*, reported in Figure 9, the monitor component can be prepared for rule execution and events listening. During the execution, through the operation *Listening(Event)*, reported in Figure 9, the monitoring component is able to start the evaluation of the different instantiated rules. Indeed, the monitoring activity of the life-cycle of each instantiated rule can be into three different stages:

- inactive, i.e., if first event of the rule condition is not received yet;
- active, i.e., if first event has been received and the monitoring is listening further ones;
- satisfied, i.e., when the rule condition is satisfied and the action associated to the rule is executed. In this case the rule passes again to the inactive stage.

Executed Data DB. This component stores data derived from system execution and analysis done by the *Monitoring* component. The data can be used later on for further analysis.

Knowledge Base (KB) Manager. This component is in charge of the DAEMON ontology management by means of the following operations: *Navigate Ontology* that allows the management of the ontology through the *GUI*, and, the selection of the desired ruleset; *Populate Ontology* that allows the definition of individuals for each ontology concept. These can be also used by the *Rules Manager* for further analysis and refinements.

Knowledge Base. This component stores the DAEMON specification and its individuals. DAEMON is represented as a Resource Definition Framework (RDF) [28] graph, it is saved in a triple store. In particular this component contains the sets of abstract, well-defined and instantiated rules and it is invoked by **Rule Manager** for updating or instantiating the single rule or a ruleset. The component provides facility for performing suitable queries.

Rules Manager. Rules Manager is the component that takes care of the management of the rules created, updated and used by the user during the usage of the system. This component supervises the evolution of the rules from *Abstract* to *Well-Defined* and finally to *Instantiated* as depicted in Figure 6. The main operations are: *RuleSet CRUD*, that defines the creation, reading, updating or deleting of a RuleSet; *Instantiate (RuleSet)*, that allows the instantiation of the well-defined rules into instantiated ones; *Execute(RuleSet)*, that loads a RuleSet of instantiated rules into the *Monitoring* component.

Considering the DAEMON architecture usage, usually the *Ontology Manager* by means of the *Populate Ontology* operation can populate DAEMON with individuals and assertions about the SoS or the device. The *Business Manager* can use the operation *Navigate ontology* to explore the already existing knowledge and to perform specific queries about suitable rules to be monitored. The selected rules are then instantiated into the CEP through the *Instantiate Rule-Set* operation, so that the *Monitoring* component could check during the SoS execution, if they are satisfied. This component is also in charge of storing the monitoring data for subsequent analysis performed by the *Business Manager*.

5 BIECO Use Case: Multi-Robot Navigation

With the main aim of ensuring trust within ICT supply chains, a holistic security framework is proposed by the EU BIECO Project [6]. The framework comprises a set of tools and methodologies for vulnerability assessment, auditing, risk analysis, determining the best mitigation strategies, ensuring resilience and certifying the security and privacy properties of the ICT components and the complete supply chain. The BIECO framework is then validated by four different use case scenarios for distinct sectors, namely: the ICT Gateway (smart grid / energy), the AI Investments platform (financial), the Smart Microfactory (industry), and the Autonomous Navigation. For the work presented in this paper, we will focus on the last use case, specifically a multi-robot autonomous navigation system.

5.1 Use Case Scenario: Multi-Robot Navigation

The use case scenario is a *CoppeliaSim* simulation with multi-robot navigation scenario for intralogistics, where software monitoring the behavior of the autonomous mobile robot in runtime, tries to detect situations in which safety concerns might be encountered.

The environment in which the multi-robot operates is a simulation of the shopfloor that includes a representation of the costmap layers used for navigation applying Robot Operating system (ROS) and its visualization tool (Rviz). Besides the navigation plans, the navigation costmap also includes a set of obstacles that can be found in the environment as illustrated in Figure 10. Therefore, under normal conditions all robots should follow the given trajectory and avoid each other and additional obstacles.

Despite the numerous components of the architectural structure of the autonomous navigation use case, namely in terms of hardware, navigation, and supervision, we will focus on the interactions that the autonomous navigation robot components might have for navigation, namely for the local planner module in the navigation environment, where:

- The navigator component controls the execution of navigation goals based on the task sent by the task manager, acting like an orchestrator, and
- The local planner component that, given a global plan to follow and a costmap including goals, final position and direct sensory input obstacle information, produces motion commands to send to a mobile base (locomotion controller).

[6] EU H2020 BIECO project Grant Agreement No. 952702, https://www.bieco.org/

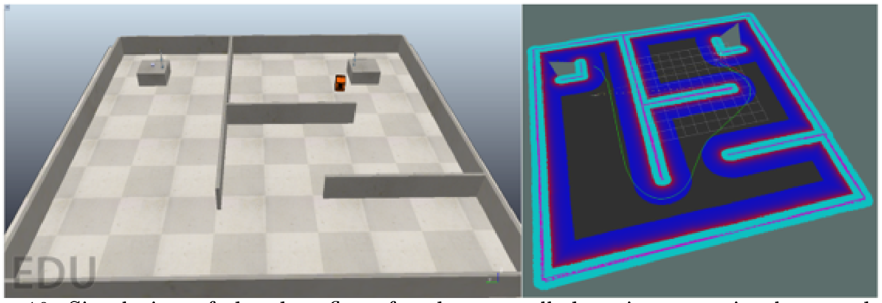


Fig. 10: Simulation of the shop-floor for the controlled environment implemented in CoppeliaSim (left). Visualization in RViz of the costmap layers for autonomous navigation, including obstacles and the navigation plans (right).

Aligned with the DAEMON Ontology described in Section 3, this use case can be represented as being influenced by the environment where it operates and including several systems present in the shop floor, being one of them the autonomous navigation multi-robot system. This system includes several devices, such as: *Robot_ Unit_ 1*, *Robot_ Unit_ 2*, *Station_ 1*, *Station_ 2*, *Task_ Manager*, *etc.*, as shown in Figure 11. Supported by DAEMON, each device is therefore composed of a set of components, as is the case of the *Robot_ Unit_ 1* device, that is composed of for example *Local_ Planner_ 1*, *Global_ Planner_ 1*, *ROS_ Bridge*, *Navigator_ 1*, *etc.*

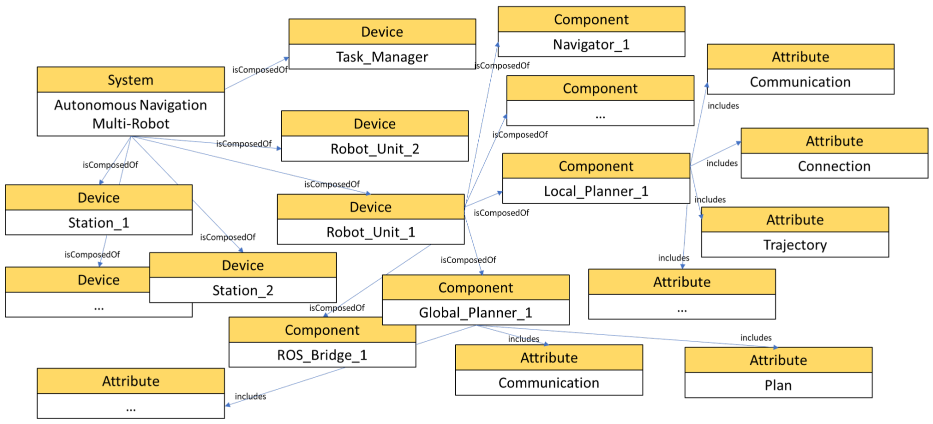


Fig. 11: System Use-Case in DAEMON (A Small Example).

For the functional and non-functional properties of the components, each device includes several attributes, as also depicted in Figure 11.

The mapping of the use case with DAEMON proceeds to the other modules, as is the case of attribute *trajectory* of the *Local_ Planner_ 1* that is composed of *linear_ velocity* and *angular_ velocity* as the *observable attribute* metric, being measured in *euler_ angles* and *length*.

Considering the case where the insertion of malicious code into one of the components of the autonomous navigation occurs, it would severely impair the system causing not only security concerns but also probably safety issues. As the mobile robots are operating in a shopfloor shared with human workers a possible malicious code insertion in their navigation system could lead to collisions and injuries.

Moreover, among several cyber-threats that can be considered in such scenario, we would like to highlight the following with catastrophic severity:

- Malicious code installed or being installed to replace the genuine software component;
- Accidental code errors that can be introduced during software development or maintenance activities;
- Viruses or worms can penetrate the ICT network from the internet and corrupt data in the system;
- Data Injection where the attacker modifies sensible data published by the system.

With the main aim at monitoring and collecting data from the environment, the *Local_Planner* and *Global_Planner* of the autonomous navigation robots have been instrumented with probes to listen to the events related to their execution. Considering the provided feedback on events, the DAEMON execution includes two different conceptual steps: the navigation and the analysis. The navigation includes the selection of the proper rules and boundaries so that it can ensure the safety and trustworthy behavior in the context of addition or update of a new module within the local planner of the Multi-Robot Navigation environment. At this stage, the refinement of the initial auditing rules will take place. After the instrumentation of the system under auditing with the probes, the analysis phase can start, as described in the following subsection.

5.2 Analysis stage

As mentioned in Section 2, different monitoring facilities can be used for instantiating the monitoring component. The implementation considered in this paper relies on the Drools [7] as rule language and Apache Artemis [8] as messages broker. During the Analysis stage, the set of well-defined rules are leveraged into executable monitoring rules through *Instantiate(RuleSet)* operation. Considering the well-defined rule shown in Figure 7, the result of the instantiation is shown in the Listing 1.1.

As in the listing, the rule is divided into two parts: the former (lines 16-21) checks if a connection is established between *Local_Planner* and *Global_Planner*; the latter (lines 23-28) checks if any additional connection is established in the meanwhile. In particular, within the *aEvent*, that is waiting for an occurrence of an event of type *ConcernBaseEvent* (line 16), sent by *SUAProbe* (line 18) to

[7] https://www.drools.org/
[8] https://activemq.apache.org/

the *Monitoring* (line 19) with an attribute *Name* set to *Connection* (line 17) and payload *Data* set to *established* (line 20). The two parts of the rule (lines 16-21 and lines 23-28 in Listing 1.1) differentiate by the *getConsumed* parameter (lines 21 and 28) that is set to true as soon as the first connection is notified. In case of more than one connection is established, a notification is sent (line 30, Listing 1.1).

```
1  package it.cnr.isti.labsedc.concern.event;
2  import it.cnr.isti.labsedc.concern.event.ConcernBaseEvent;
3  import it.cnr.isti.labsedc.concern.notification.Manager;
4
5  dialect "java"
6  declare ConcernBaseEvent
7      @role( event )
8      @timestamp( timestamp )
9  end
10
11 rule "check that only one connection is active"
12 no-loop
13 salience 200
14 dialect "java"
15  when
16   $aEvent : ConcernBaseEvent(
17   this.getName == "Connection",
18   this.getSenderID == "SUA_Probe",
19   this.getDestinationID == "Monitoring",
20   this.getData == "established",
21   this.getConsumed == true);
22
23   $bEvent : ConcernBaseEvent(
24   this.getName == "Connection",
25   this.getSenderID == "SUA_Probe",
26   this.getDestinationID == "Monitoring",
27   this.getData == "established",
28   this.getConsumed == false);
29  then
30   Manager.print("Connection amount violation on Local Planner");
31   retract($aEvent);
32   retract($bEvent);
33 end
```

Listing 1.1: Instantiated Rule Example.

The instantiate rule is then injected into the CEP of the Monitoring Component so that the monitoring activity can start. This includes running the *Local planner* and *Global Planner* within the Multi-Robot Navigation environment and the listening of events related to their execution sent by the relative probes. In the execution of the presented Use Case, a malicious code attack has been simulated. For this purpose, the malicious code injection is performed through the UI provided with the controlled environment. Due to this malicious behavior, the number of connections between the *Local planner* and *Global planner* increases over the threshold that has been set to 1. On the monitoring side this causes the violation of the *check that only one connection is active* rule shown in Listing 1.1. In this case a notification is generated and sent by the Monitoring component (line 30, Listing 1.1). This untrusted behavior detected can be alternatively managed by the monitoring providing a dynamic reconfiguration suggestion to the Controlled Environment or putting in place an instant countermeasure returning the system to a safe and trusted condition.

6 Conclusion and Future Work

In this paper, we introduced DAEMON, an ontology that models SoS and monitoring concepts uniquely and comprehensively to help SoS stakeholders monitor the behavior of SoS at runtime. DAEMON is supported by a reference architecture that enables: i) lowering down costs of developing and setting up the monitoring environment, i.e., allowing the use of available monitor engines (e.g., GLIMPSE [9, 6]); ii) improving quality control by smart and effective rules specification and encoding; iii) increasing flexibility and productivity and making the monitor designers agnostic of the domain-specific challenges (see for instance [32]); and iv) increasing the interoperability by using standardized and domain- independent specification technologies (e.g., OWL [30] for the Ontology description, and RuleML [12] or Drools [9] for instantiating rules).

We have validated our proposal through the Multi-Robot Navigation use-case scenario within the EU BIECO project, demonstrating the feasibility of both DAEMON and its reference architecture. Inside the BIECO project, we are finalizing a customized implementation of the DAEMON architecture. As a future work, we plan to use the BIECO release to evaluate the effectiveness, and the overhead of the DAEMON application. In particular, we are currently working on validating our proposal within the ICT Gateway (smart grid/energy) and the Smart Microfactory (industrial environment) use case scenarios. This will provide important data for the DAEMON performance not only in terms of processing and communication overhead, but also of development and usage effort. Additionally, comparison against related solutions will be also provided as well as the contextualization of DAEMON in other technological domains such as IoT for smart cities.

Acknowledgements This work was partially supported by the EU H2020 BIECO project Grant Agreement No. 952702, by the CyberSec4Europe H2020 Grant Agreement No. 830929, and by the Portuguese "Fundação para a Ciência e Tecnologia" "Strategic program UIDB/00066/2020" (UNINOVA-CTS project).

References

1. et al., R.S.P.: The BIECO conceptual framework towards security and trust in ict ecosystems. In: Testing Software and Systems. pp. 230–232. Cham (2022)
2. de Almeida, V.P., Bhowmik, S., Lima, G., Endler, M., Rothermel, K.: Dscep: An infrastructure for decentralized semantic complex event processing. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 391–398. IEEE (2020)
3. Arp, R., Smith, B., Spear, A.D.: Building Ontologies with Basic Formal Ontology. The MIT Press (2015)
4. Baek, Y.M., Song, J., Shin, Y.J., Park, S., Bae, D.H.: A meta-model for representing system-of-systems ontologies. pp. 1–7. ACM (5 2018)

[9] https://www.drools.org/

5. Bajaj, G., Agarwal, R., Singh, P., Georgantas, N., Issarny, V.: A study of existing ontologies in the iot-domain. arXiv preprint arXiv:1707.00112 (2017)
6. Barsocchi, P., Calabrò, A., Ferro, E., Gennaro, C., Marchetti, E., Vairo, C.: Boosting a low-cost smart home environment with usage and access control rules. Sensors **18**(6), 1886 (2018)
7. 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. pp. 3–17. Springer (2019)
8. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: Iot-lite: A lightweight semantic model for the internet of things. pp. 90–97. IEEE (7 2016)
9. Bertolino, A., Calabrò, A., Lonetti, F., Sabetta, A.: GLIMPSE: a generic and flexible monitoring infrastructure. In: Giandomenico, F.D. (ed.) Proceedings of the 13th European Workshop on Dependable Computing, EWDC '11, Pisa, Italy, May 11-12, 2011. pp. 73–78. ACM (2011)
10. Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., Meskin, N.: Cybersecurity for industrial control systems: A survey. computers & security **89**, 101677 (2020)
11. Bhatt, S.N., Manadhata, P.K., Zomlot, L.: The operational role of security information and event management systems. IEEE Secur. Priv. **12**(5), 35–41 (2014)
12. Boley, H., Tabet, S., Wagner, G.: Design rationale for ruleml: A markup language for semantic web rules. In: Proc. of The 1st Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001. pp. 381–401 (2001)
13. 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). pp. 381–386 (2018)
14. Calabrò, A., Daoudagh, S., Marchetti, E.: MENTORS: monitoring environment for system of systems. In: Mayo, F.J.D., Marchiori, M., Filipe, J. (eds.) Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021. pp. 291–298. SCITEPRESS (2021)
15. Daniele, L., den Hartog, F., Roes, J.: Created in close interaction with the industry: The smart appliances reference (saref) ontology. In: Formal Ontologies Meet Industry. pp. 100–112. Springer International Publishing, Cham (2015)
16. DRIDI, C.E., 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). pp. 1–8 (2020)
17. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation). Official Journal of the European Union **L119**, 1–88 (May 2016)
18. Fortino, G., Savaglio, C., Spezzano, G., Zhou, M.: Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools. IEEE Transactions on Systems, Man, and Cybernetics: Systems **51**(1), 223–236 (2020)
19. 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, pp. 55–61. SPIE (2020)
20. Franzén, L.K., Staack, I., Jouannet, C., Krus, P.: An Ontological Approach to System of Systems Engineering in Product Development (10 2019)
21. 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). pp. 254–261 (2016)

22. Hidayanti, F.: Design and application of monitoring system for electrical energy based-on internet of things. Helix **10**(01), 18–26 (2020)
23. Holubová, I., Vavrek, M., Scherzinger, S.: Evolution management in multi-model databases. Data & Knowledge Engineering **136**, 101932 (2021)
24. 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. Computers & Electrical Engineering **89**, 106906 (2021)
25. Langford, G., Langford, T.: The making of a system of systems: Ontology reveals the true nature of emergence. pp. 1–5. IEEE (6 2017)
26. LOV4IoT: Lov4iot-iot ontology catalog: Reusing domain knowledge expertise, http://ww.lov4iot.appspot.com/?p=lov4iot-iot (Last Access 15/08/2022)
27. Lynch, K., Ramsey, R., Ball, G., Schmit, M., Collins, K.: Conceptual design acceleration for cyber-physical systems. pp. 1–6. IEEE (4 2017)
28. Manola, F., Miller, E.: RDF Primer. W3C Recommendation, WWW Consortium (2004), http://www.w3.org/TR/rdf-primer/
29. Mascardi, V., Cordì, V., Rosso, P.: A comparison of upper ontologies. In: Woa. vol. 2007, pp. 55–64. Citeseer (2007)
30. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language structural specification and functional-style syntax (second edition). W3C recommendation, World Wide Web Consortium (2012)
31. Nilsson, R., Dori, D., Jayawant, Y., Petnga, L., Kohen, H., Yokell, M.: Towards an ontology for collaboration in system of systems context. INCOSE International Symposium **30**, 666–679 (7 2020)
32. Palumbo, F.: Leveraging smart environments for runtime resources management. In: Software Quality: Methods and Tools for Better Software and Systems: 10th International Conference, SWQD 2018, Vienna, Austria, January 16–19, 2018, Proceedings. vol. 302, p. 171. Springer (2018)
33. oneM2M Partners Type 1: Base ontology (5 2019), https://www.onem2m.org/images/pdf/TS-0012-Base_Ontology-V3_7_3.pdf
34. 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)
35. Rastogi, V., Srivastava, S., Mishra, M., Thukral, R.: Predictive maintenance for sme in industry 4.0. In: 2020 Global Smart Industry Conference (GloSIC). pp. 382–390 (2020)
36. Santos, M.A., Munoz, R., Olivares, R., Rebouças Filho, P.P., Del Ser, J., de Albuquerque, V.H.C.: Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook. Information Fusion **53**, 222–239 (2020)
37. Scioscia, F., Ruta, M.: Building a semantic web of things: Issues and perspectives in information compression. pp. 589–594. IEEE (9 2009)
38. Sforzin A., Bobba R. et al.: D5.4–Requirements Analysis of Demonstration Cases Phase 2. https://cybersec4europe.eu/publications/deliverables/ (2021)
39. Skouloudi, R.C., Malatras, A., Dede, G.: Guidelines for securing the internet of things. European Union Agency for Cybersecurity (2020)
40. Walden, D.D., Roedler, G.J., Forsberg, K.J., Hamelin, D.R., Shortell, T.M.: Systems Engineering Handbook. Wiley, 4 edn. (2015)
41. Won, M.: Intelligent traffic monitoring systems for vehicle classification: A survey. IEEE Access **8**, 73340–73358 (2020)