MENTORS: Monitoring Environment for System of Systems

Antonello Calabrò[a], Said Daoudagh[b] and Eda Marchetti[c]
ISTI-CNR, Pisa, Italy

Keywords: Ontology, Run-time Monitoring, Vulnerability Detection.

Abstract: *Context:* Systems Of Systems (SoSs) are becoming a widespread emerging architecture, and they are used in several daily life contexts. Therefore, when a new device is integrated into an existing SoS, facilities able to efficaciously assess and prevent anomalous and dangerous situations are necessary. *Objective:* The aim is to define a reference environment conceived for monitoring and assessing the behavior of SoS when a new device is added. *Method:* In this paper, we present MENTORS, a monitoring environment for SoS. MENTORS is based on semantic web technologies to formally represent SoS and Monitoring knowledge through a core ontology, called MONTOLOGY. *Results and Conclusion:* We defined the conceptual model of MENTORS, which is composed of two phases: Off-line and On-line, supported by a reference architecture that allows its (semi-)automation. Validation of the proposal with real use-cases is part of future activities.

1 INTRODUCTION

Nowadays, Systems of Systems (SoSs) are becoming an emerging widespread architecture. SoSs are used in several daily life contexts: from our homes through to the industrial assembly lines. Often the development of SoS requires the integration and collaboration of different ICT components or devices typically developed by several third parties. If from the one hand this process assures high productivity and competitiveness, from the one other it exposes the SoS to important vulnerabilities and risks. A typical consequence is that each of the integrated components could be affected by, and involuntarily propagate, the vulnerabilities of the others included in the SoS, with potential drastically consequences. In this situation, when a new device or component is integrated into an existing SoS, facilities able to efficaciously assess and prevent anomalous and dangerous situations are therefore necessary. Among them, one commonly adopted is the use of a monitoring system (Ullo and Sinha, 2020), i.e., a means for the on-line analysis of functional and non-functional properties.

Indeed, a monitor engine collects events from different levels from the different system components (or sensors) and uses these data to infer complex patterns that could represent a specific functional or non functional property. The derived complex patterns represent the observed normal (or abnormal) behavior of the monitored system (or its components). Therefore, the monitor is able to: i) collect and analyze data coming from the different SoS sources (e.g., sensors, components or devices); ii) assess the run time SoS (components or devices) behaviour; iii) promptly rise up alarms in case of violations; and, iv) put in place countermeasures if necessary.

However, even if notably efficient and effective, the design, implementation, and management of a monitoring system inside an SoS environment may involve the participation of different stakeholders such as SoS domain experts, device developers, or monitoring experts. Thus, due to the intrinsic complexity of the scenario, monitoring the integration of new devices (or components) in different SoS environments can result in a complex and costly activity (Rastogi et al., 2020). One of its main criticalities is the lack of a common understanding: experts of SoS domain could ignore the peculiarities of monitoring activity and vice versa. Thus, this paper wants to move a step ahead in this direction, by joining the different definitions of the SoS and monitoring context into a unique structural representation. For this, we conceived a smart and efficient monitoring processes for SoS, called MENTORS - Monitoring ENviromenT fOR Sos. MENTORS can provide a support for all the stakeholders involved in the monitoring process with the purpose of: i) lowering

[a] https://orcid.org/0000-0001-5502-303X
[b] https://orcid.org/0000-0002-3073-6217
[c] https://orcid.org/0000-0003-4223-8036

Calabrò, A., Daoudagh, S. and Marchetti, E.
MENTORS: Monitoring Environment for System of Systems.
DOI: 10.5220/0010658900003058
In *Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST 2021)*, pages 291-298
ISBN: 978-989-758-536-4; ISSN: 2184-3252

down costs of developing and setting up the monitoring environment, i.e., allowing the use of available monitor engines (e.g., GLIMPSE (Bertolino et al., 2011; Barsocchi et al., 2018)) without the necessity to implement ad hoc solutions; ii) improving quality control by smart and effective rules specification and encoding; iii) increasing flexibility and productivity by automating the monitoring process through a reference architecture so as to make the monitor designers agnostic of the domain specific challenges (see for instance (Palumbo, 2018)); and iv) increase the interoperability by using standardized and domain independent specification technologies (e.g., OWL (Motik et al., 2012) for the Ontology description, and RuleML (Boley et al., 2001) or Drools [1] for instantiating rules).

Outline: Section 2 reports related work, whereas the conceived monitoring process for the SoS, composed of two main phases (off-line and On-line) is introduced in Section 3. Section 4 illustrates the definition of standardized and domain independent specification of the SoS and monitoring concepts. A preliminary structure of the reference architecture allowing a (semi-)automation of the proposed process is reported in Section 5. Finally, Section 6 concludes the paper and reports future work.

2 RELATED WORK

At state of the practice, different run time solutions are currently available in many domains: traffic monitoring systems (Won, 2020), indicators on the control panel of a car (Fotescu et al., 2020), airplane or electronic device (Hidayanti, 2020), monitors for physiological conditions of patients in a hospital (Santos et al., 2020) or systems for controlling complex large-scale systems such as airports, railways or industrial plants (Bhamare et al., 2020). Usually, all the monitoring solutions address two main challenges: i) finding very powerful, concise and unambiguous specification languages able to express the properties to be validated (Khan et al., 2021); ii) defining efficient mechanisms to assess conformity of the system against these properties or contracts (Burns et al., 2018). In line with the current research, the aim of this work is to advance the state of practice of the monitoring process, by conceiving an easy to use and effective solution for the application of the monitoring activity inside SoS. Indeed, MENTORS can enhances the existing monitoring approaches for: 1) promptly rising warnings and detecting failures, and for enabling system reconfiguration, reliable and trustworthy execution; 2) capturing, inferring and analysing complex events, so to allow the detection of critical problems, failures and security vulnerabilities; and 3) validating the trust and security level of run-time behavior of SoS and its devices.

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

3 CONCEPTUAL MODEL AND PROCESS

From a conceptual point of view, the monitoring process presented in this paper is based on a structural representation of the concepts and knowledge about the SoS, devices and monitoring environments.

As schematized in Figure 1, the MENTORS monitoring process is structured into two different phases: i) **Off-line phase**, in which the concepts and the knowledge about the different SoS and the available devices are organized and integrated in a structural specification, and monitoring rules defined; ii) **On-line phase**, in which the monitoring activities are executed, results collected and inferences about the observed behavior and possible rules improvements defined.

Specifically, as reported in Figure 1, during the former phase the conceptual steps are:

- **SoS Specification & Monitoring Specification:** SoSs are widespread adopted in several application domain, each having its specific needs and challenges (Raman and D'Souza, 2019). However, at the state of the practice there is not a common knowledge-base for collecting into a unique environment the SoS and monitoring requirements, specifications, architectural design, and the set of existing or conceivable monitoring rules useful for the assessment or quality improvement (Vierhauser et al., 2016). This kind of information is still on the hands of a few experts and it is not easily shareable between the different stakeholders. As reported Figure 1, MENTORS provides a solution for this lack, i.e., a continuously updatable structural representation of the research and practical knowledge about SoS and the monitoring anemometers.
- **SoS & Monitoring Ontology:** At the state of the practice, one of the recognized means for knowledge representation and environmental data modeling is the use of ontologies (Motik et al., 2012). They provide a common vocabulary useful for modeling and understanding specific domains by capturing knowledge in a structured and

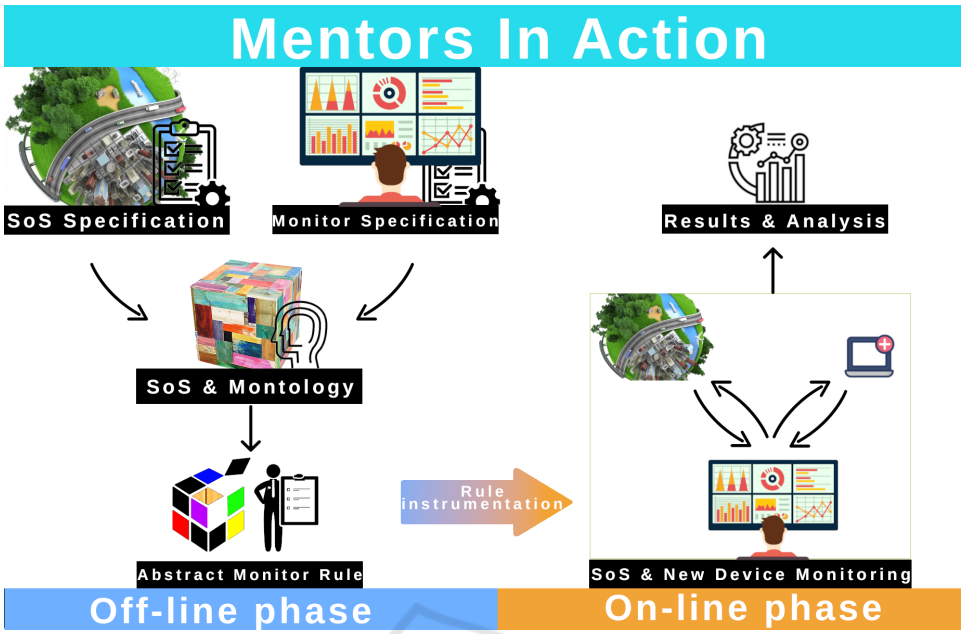

Figure 1: MENTORS: Conceptual Model.

formal way. As in Figure 1, MENTORS organizes, revises and structures the available knowledge by means of an ontology (Berners-Lee et al., 2001) called MONTOLOGY - MONitoring ontTOLOGY (see Section 4).

- **Abstract Monitoring Rules:** In the context of SoS, and considering in particular the rules definition, even if traditional performance indicators could be still relevant, the dynamic evolution and complexity of SoS require the availability of suitable rules, able to take into account also events risen by the SoS context in which the device (or its components) is included. Consequently, successful rules definition requires a deep knowledge of the SoS domain and its environment as well as monitoring behavior and management. By exploiting the MONTOLOGY structure, the proposal of this paper provides both: i) a common evolving rules definition, able to improving quality control and increasing flexibility and productivity of the monitoring process; ii) means for easily instrumenting the selected set of rules according to the selected SoS domain needs and attributes.

Crossing from Off-line phase to On-line phase, the `Abstract Monitoring Rules`, generated through MONTOLOGY, will be instantiated for a specific rule language understandable by the selected Monitoring component, and enacted for monitoring the *SoS and New Device* during the On-line phase. Therefore, during the On-line phase, the conceptual steps are:

SoS & New Device Monitoring: The instantiated rules are used to monitor the new device behavior inside a target SoS. During this activity, possible rule violations are detected and possibly managed.
Results & Analysis: The monitored data are then analyzed so as to infer new knowledge about the SoS or Device behavior, or to enrich the MONTOLOGY contents.
In the remainder of this paper, details about MONTOLOGY and the MENTORS reference architecture will be provided in Section 4 and Section 5.3, respectively. It is out of the scope of the paper discussing the validation of the proposal, which is part of our future work.

4 MONTOLOGY: MONITORING ONTOLOGY

In this section, we introduce our core ontology (called MONTOLOGY) for supporting the realization of MENTORS. We firstly present the development process in Section 4.1, and then we briefly describe its main modules in Section 4.2.

4.1 Process for Developing MONTOLOGY

The process adopted for developing MONTOLOGY is based on (Gharib et al., 2021), and it follows the

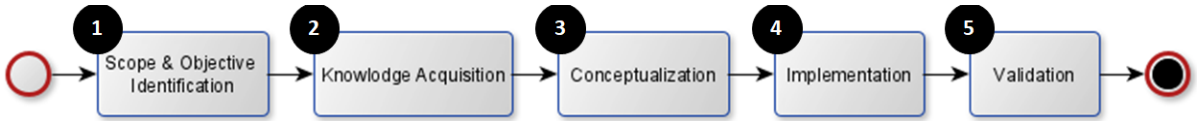

Figure 2: Process for Developing MONTOLOGY (adapted from (Gharib et al., 2021)).

five principles proposed in (Gruber, 1995) (i.e., clarity, coherence, extendibility, minimal encoding bias, and minimal ontological commitment).

As depicted in Figure 2, the process is composed of five main steps:

1. Scope & Objective Identification (Step ①) where the scope and the purposes of the ontology and its intended end-users are identified. In our case, the purpose is to allow any possible stakeholder to define, for every specific SoS domain, a set of effective and well-defined monitoring rules able to reveal the vulnerabilities and or problems that a new device integration could arise.
2. Knowledge Acquisition (Step ②) where the knowledge needed for the realization of the intended ontology is collected. During this step, monitoring and SoS experts have been involved both for the identification of the concepts and their relationships, as well as for capturing monitoring and SoSs requirements. Analysis of the available literature concerning SoS and monitoring has been also performed (de Almeida et al., 2020; Silva et al., 2015).
3. Conceptualization (Step ③) where the core ontology (i.e., MONTOLOGY) for supporting monitoring process is derived. MONTOLOGY is composed of 13 concepts: 8 coming from SoS domain, and 5 coming from monitoring domain (see Section 4.2 for more details).
4. Implementation (Step ④) where the conceptualized ontology is codified into a formal language. This requires an environment that guarantees (1) checking lexical and syntactic errors so as to implement an error-free ontology, and (2) an automated reasoner for detecting inconsistencies and redundant knowledge. For MONTOLOGY implementation, Protégé [2] has been selected because: it is an open-source environment allowing creating, modifying, visualizing and checking the consistency; and it relies on SPARQL [3] (Protocol and RDF Query Language) for knowledge extraction.
5. Validation (Step ⑤) where the validation of MONTOLOGY is performed to assure that it meets the needs of its usage. For this purpose, specific SPARQL queries are being defined to evaluate the MONTOLOGY level of detail.

4.2 MONTOLOGY Modules

MONTOLOGY (MONitoring onTOLOGY) is composed of two modules: System of Systems (SoS) module (yellow classes in Figure 3) and Monitoring module (green classes in Figure 3). Due to space limitation, we just give some details about the two modules.

System of Systems (SoS) Module. It models SoS and its components, as well as the relationship among them. As in Figure 3, a *SystemOfSystems* is a collection of Devices that represent the object of the monitoring activities. Each *Device* is composed of a specific set of *Components*. SoS and Components are related to as set *Attributes*: SoS is connected to *SoSAttribute* through the *hasSoSAtribute* relationship, whereas *Component* is connected to *ComponentAttribute* through the *haCompAttribute* relationship. As in the figure, *Attribute* is connected to both *Measure* and *Metric* concepts, and it is a super-class of SoSAttribute and ComponentAttribute. Through this hierarchy, these sub-classes can be connected to Measure and Metric concepts as well.

Monitoring Module. It aims at modelling monitoring concepts and relationship between them. The core class of Monitoring module is the *Rule*, which is the atomic observable concept within MENTORS. Indeed, the *Monitor* observes rules organized in *Calendar*, which is an ordered set of rules. Each *Calendar* is able to validate a specific *Skill* at run-time, i.e., during the on-line phase depicted in Figure 1. *Skill* is specified as a set of *Requirements*, each verified through a specific *Rule* (see the *verifiesRequirement* relationship).

The two main modules are then connected each other as following: (1) each SystemOfSystems defines a set of Skill, each related to a specific Component; (2) the core element of Monitoring module, i.e., Rule concept, is defined through a set of Attributes.

[2]https://protege.stanford.edu/

[3]https://www.w3.org/TR/rdf-sparql-query/

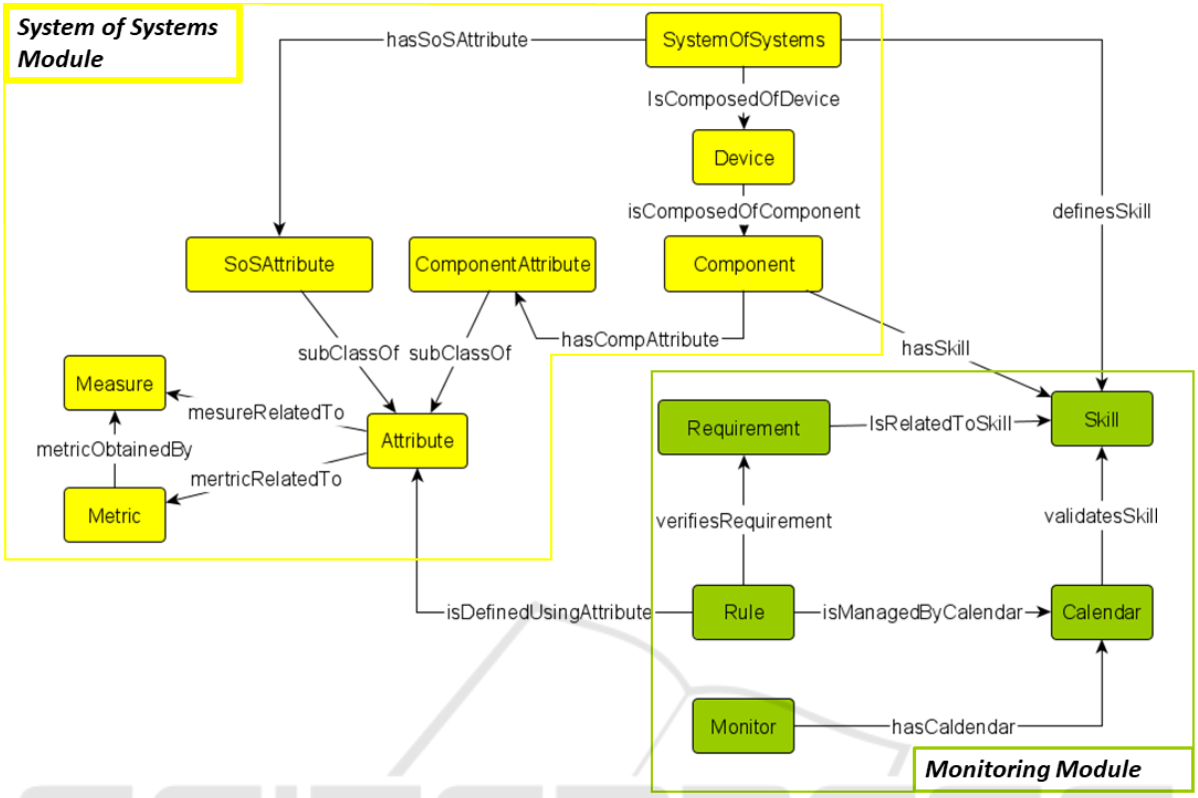

Figure 3: Overview of MONTOLOGY.

5 MENTORS: ACTORS, USE CASES AND ARCHITECTURE

In this section, we describe MENTORS, a monitoring environment for System of Systems (SoS). We firstly illustrate the main actors involved in MENTORS; then, we describe the supported uses cases; finally, we present the conceived MENTORS reference Architecture.

5.1 Actors

As anticipated in the introduction, the monitoring process may involves several stakeholders as schematized in Figure 4 Box (A). In particular, the main actor is *MENTORS user* who is the generic actor interacting with MENTORS framework. This actor can be in turn specialized into different actors. The *Business Manager*, who is the user of the monitoring system. She or he is in charge of the specification of the monitoring needs (e.g., rules) as well as the administration and management of the monitoring data analysis and alerts. The *Learning System* that represents an autonomous system able to analyze the collected monitoring data, so as to infer new knowledge (e.g., new rules, concepts or relationships) for improving MONTOLOGY. The *CEP Programmer* who is an expert of a specific monitoring system, and she or he is in charge of instantiating the rules subset into a precise rules language. The *SoS Expert* who is expert of the SoS domain, its constituent parts and their attributes. The *Ontology User* who has the knowledge of the ontology concepts and management, and she or he is the responsible to conceptualize, to implement and to maintain MONTOLOGY. This actor can be specialized into *Device Expert*. They are experts of a specific device, and he or she knows the Device constituent parts, their needs and their attributes.

5.2 Use Cases

The use cases were guided by the conceptual model introduced in Section 3. To better clarify the roles of each actor, in Figure 4 Box (B) the main functionalities provided by MENTORS are reported. In particular, *Ontology User* and *Learning System* use the *Populate Ontology* feature for specifying his or her knowledge of the SoS or the device. More precisely, *Ontology User* can populate MONTOLOGY with in-

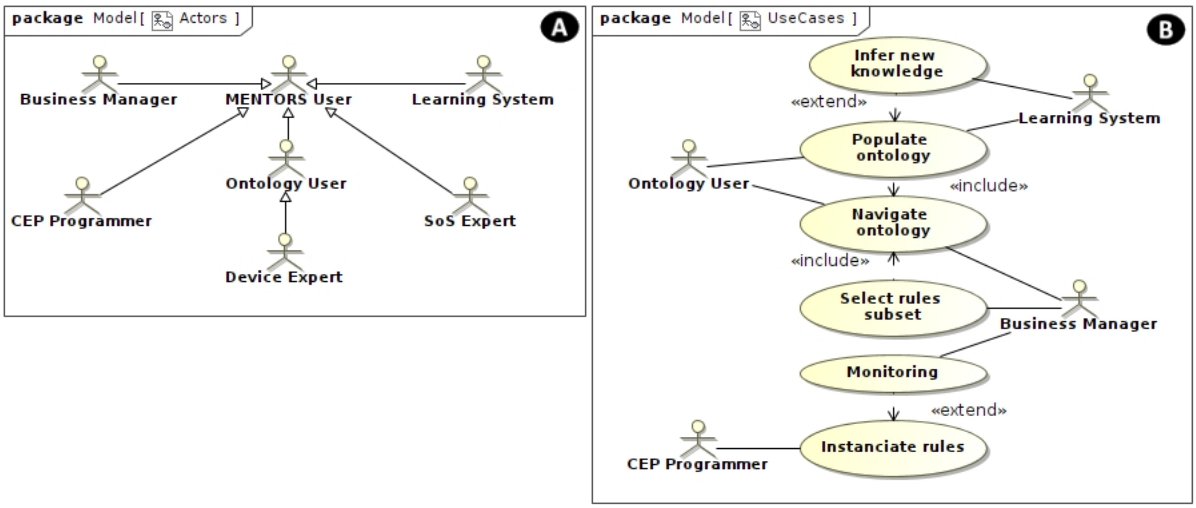

Figure 4: MENTORS: Actors and Use Cases.

dividuals and assertions about the SoS or the device. This feature includes also the possibility to modify the content of MONTOLOGY or extend it with already existing ontologies. The *Ontology User* and the *Business Manager* have then the possibility to *Navigate ontology*, so as to explore the already existing knowledge and to perform specific queries. Additionally, *Business Manager*, through *Select rules subset* can navigate the MONTOLOGY and can specify the most suitable rules to be monitored. The selected rules will then be instantiated by the *CEP programmer* through *Instantiate rule* feature so as to be observed by selected monitoring infrastructure. In particular, *Monitoring* feature is in charge of checking if the instantiated rules are respected or not. This functionality is also in charge of the storage of the monitoring data so as to let the *Business Manager* to analyze them.

5.3 Proposed Architecture

Our reference architecture is depicted in Figure 5, which is an abstract architecture that can be customized with different real tools. MENTORS reference architecture is composed of the following components:

(1) **GUI** provides a user-centric & user-friendly graphical interface that allows *MENTORS Users* to interact with MENTORS for performing all the available functionalities previously introduced and reported in Figure 4 Box (B).

(2) **Monitoring** component is the core of the analysis of the system execution implementing the *Monitoring* and *Instantiate rules* features described in Section 5.2. It includes the Complex Event Processor (CEP) (de Almeida et al., 2020) subcomponent that is in charge of inferring simple and complex pattern from the events captured during the run time. An event, is a snapshot of a change of the status within a system. Events are captured by Probes, which are piece of code injected within the device to monitor. Probes are n charge to put on an envelope the change of status occurred within the component monitored and sent it to the monitoring. The received data will be matched with the instantiated rules and results will be stored into the *Executed Data DB* for further analysis.

(3) **Instantiated Rules DB** contains the rules selected by *Business Manager* through the *GUI*, and instantiated by the *CEP Programmer*. The rules are translated into the language of the CEP. In our case, the Drools Language [4] has been used for the scope.

(4) **Executed Data DB** stores data derived from system execution, and it makes those data available for further analysis.

(5) **KB Manager** implements *Populate ontology*, *Navigate Ontology* and *Select rules subset* features (see Section 5.2). Specifically, users can manage the ontology through the *GUI*, while the *Learning System* can access through specific API. As in Figure 5, *KB Manager* interacts with the *Knowledge Base* component for updating and managing MONTOLOGY. *KB Manager* can be considered as a middle-ware between: the known *Knowledge Base*, the observed *Monitoring* and

[4]Details about Drools Language can be found at: https://www.drools.org/

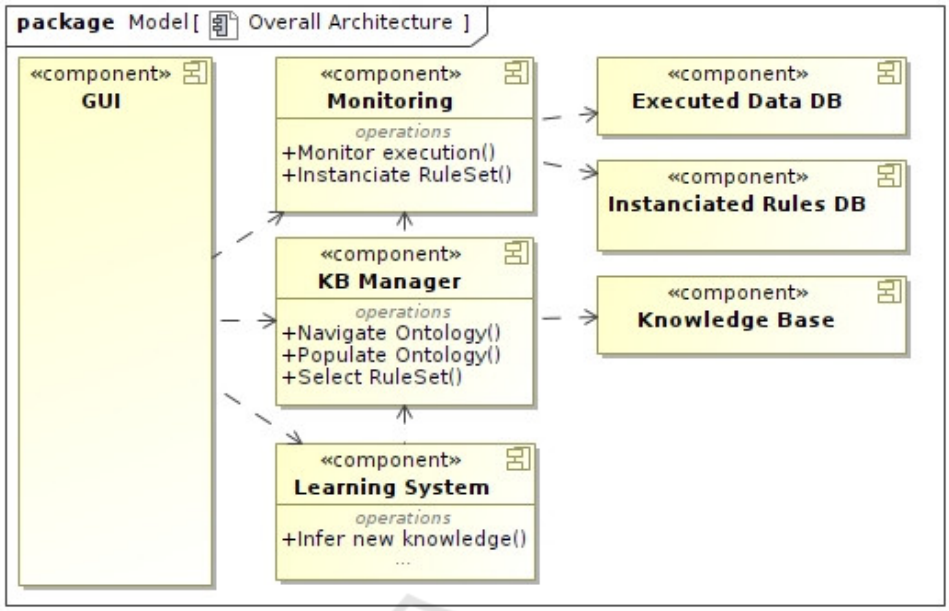

Figure 5: MENTORS: Supporting Predictive Monitoring Architecture.

the inferred *Learning System*.

(6) **Knowledge Base** allows the MONTOLOGY specification and update. In particular, MONTOLOGY is represented as an RDF (Manola and Miller, 2004) graph, which is saved in a triple store on which it is possible to perform queries.

(7) **Learning System** implements the *Infer new knowledge* and contributes to the *Populate Ontology* feature (see Section 5.2). Specifically, this component, through a reasoner, analyzes the *Knowledge Base* data and the information collected in the *Executed Data DB*, for both performing consistency checks and inferring new emerging knowledge. The results of this analysis are provided to *Knowledge Base* component through *KB Manager*. This component can be instantiated with different reasoners available in literature [5], such as Openllet [6].

6 CONCLUSIONS

In this paper, we have introduced MENTORS, a reference monitoring environment specifically conceived for SoS. In particular, MENTORS is targeting the situation in which a new device or component is integrated into an existing SoS, so as to assess and prevent anomalous and dangerous behaviour of both the device and the SoS. We firstly conceptualized MENTORS as process consisting of two main parts, Off-line and On-line. The former is supported by an ontological representation of the SoS and Monitoring domains through MONTOLOGY ontology. This is a core ontology useful for developing and deriving meaningful rules that can be monitored, and it can be exploited also for inferring new knowledge about the SoS. The latter consists of monitoring the conceived rules so as to promptly identify misbehaviour in terms of run-time violations. To this purpose, we have also presented a reference architecture that can be customized with different tools. As future work, we are planning to thoroughly validate MENTORS within an ongoing European Project. This validation could improve both MONTOLOGY and MENTORS reference architecture for supporting new functionalities.

ACKNOWLEDGEMENTS

This work was partially supported by CyberSec4Europe H2020 Grant Agreement No. 830929 and the EU H2020 BIECO project Grant Agreement No. 952702.

[5] An updated list of the currently available reasoners can be found in: https://www.w3.org/2001/sw/wiki/OWL/Implementations

[6] https://github.com/Galigator/openllet

REFERENCES

Barsocchi, P., Calabrò, A., Ferro, E., Gennaro, C., Marchetti, E., and Vairo, C. (2018). Boosting a low-cost smart home environment with usage and access control rules. *Sensors*, 18(6):1886.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):28–37.

Bertolino, A., Calabrò, A., Lonetti, F., and Sabetta, A. (2011). GLIMPSE: a generic and flexible monitoring infrastructure. In Giandomenico, F. D., editor, *Proceedings of the 13th European Workshop on Dependable Computing, EWDC '11, Pisa, Italy, May 11-12, 2011*, pages 73–78. ACM.

Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., and Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. *computers & security*, 89:101677.

Boley, H., Tabet, S., and Wagner, G. (2001). Design rationale for ruleml: A markup language for semantic web rules. In Cruz, I. F., Decker, S., Euzenat, J., and McGuinness, D. L., editors, *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*, pages 381–401.

Burns, M., Griffor, E., Balduccini, M., Vishik, C., Huth, M., and Wollman, D. (2018). Reasoning about smart city. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 381–386.

de Almeida, V. P., Bhowmik, S., Lima, G., Endler, M., and Rothermel, K. (2020). Dscep: An infrastructure for decentralized semantic complex event processing. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 391–398. IEEE.

Fotescu, R.-P., Constantinescu, R., Alexandrescu, B., and Burciu, L.-M. (2020). System for monitoring the parameters of vehicle. In *Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies X*, volume 11718, page 117180A. International Society for Optics and Photonics.

Gharib, M., Giorgini, P., and Mylopoulos, J. (2021). Copri v.2 — a core ontology for privacy requirements. *Data & Knowledge Engineering*, 133:101888.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907–928.

Hidayanti, F. (2020). Design and application of monitoring system for electrical energy based-on internet of things. *Helix*, 10(01):18–26.

Khan, S., Nazir, S., García-Magariño, I., and Hussain, A. (2021). Deep learning-based urban big data fusion in smart cities: Towards traffic monitoring and flow-preserving fusion. *Computers & Electrical Engineering*, 89:106906.

Manola, F. and Miller, E. (2004). RDF Primer. W3C Recommendation, WWW Consortium. http://www.w3.org/TR/rdf-primer/.

Motik, B., Patel-Schneider, P. F., and Parsia, B. (2012). OWL 2 Web Ontology Language structural specification and functional-style syntax (second edition). W3C recommendation, World Wide Web Consortium.

Palumbo, F. (2018). 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*, volume 302, page 171. Springer.

Raman, R. and D'Souza, M. (2019). Decision learning framework for architecture design decisions of complex systems and system-of-systems. *Systems Engineering*, 22(6):538–560.

Rastogi, V., Srivastava, S., Mishra, M., and Thukral, R. (2020). Predictive maintenance for sme in industry 4.0. In *2020 Global Smart Industry Conference (GloSIC)*, pages 382–390.

Santos, M. A., Munoz, R., Olivares, R., Rebouças Filho, P. P., Del Ser, J., and de Albuquerque, V. H. C. (2020). Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook. *Information Fusion*, 53:222–239.

Silva, E., Batista, T., and Oquendo, F. (2015). A mission-oriented approach for designing system-of-systems. In *2015 10th System of Systems Engineering Conference (SoSE)*, pages 346–351. IEEE.

Ullo, S. L. and Sinha, G. R. (2020). Advances in smart environment monitoring systems using iot and sensors. *Sensors*, 20(11).

Vierhauser, M., Rabiser, R., Grünbacher, P., Seyerlehner, K., Wallner, S., and Zeisel, H. (2016). Reminds : A flexible runtime monitoring framework for systems of systems. *Journal of Systems and Software*, 112:123–136.

Won, M. (2020). Intelligent traffic monitoring systems for vehicle classification: A survey. *IEEE Access*, 8:73340–73358.