Towards enhanced monitoring framework with smart predictions

ANTONELLO CALABRÒ[*], *Institute of Information Science and Technologies (ISTI) - CNR - Pisa, Italy.*

SAID DAOUDAGH[**], *Institute of Information Science and Technologies (ISTI) - CNR - Pisa, Italy.*

EDA MARCHETTI[†], *Institute of Information Science and Technologies (ISTI) - CNR - Pisa, Italy.*

Abstract

Context: Predicting security and trust vulnerabilities and issues is crucial for IoT interconnected systems and ecosystems, especially when integrating new, third-party or open-source components. ***Objective:*** One way to ensure timely predictions is by using a smart monitoring framework to continuously verify functional and non-functional property violations during the executions of the systems and their components. ***Method:*** This paper presents a set of guidelines for the Smart Monitoring Framework definition and its application process. ***Results and Conclusion:*** The paper provides the reference architecture of the Smart Monitoring Framework and its possible implementation to promptly detect suspicious behavior or property violations. The paper also illustrates how the provided implementation satisfies the defined guidelines by design.

Keywords: Cybersecurity; ecosystem; malicious attack; predictive simulation; runtime monitoring; trust; vulnerability

1 Introduction

The current smart, agile and collaborative development processes and frameworks ask for Information and Communication Technology (ICT) solutions to integrate new or available third parties or open-source components and systems. Even though this way of proceeding is recognized as essential to maintaining productivity and competitiveness, it can introduce a high risk regarding security. Indeed, in most cases, there is no way to verify if these integrable solutions have vulnerabilities, have been built considering the best security practices or can cause suspicious/malicious behaviors. Consequently, a stringent need is to have smart and effective means for assessing and guaranteeing the security and privacy properties and predicting possible vulnerabilities and their propagation over the ICT ecosystems as soon as possible [16]. Indeed, monitoring approaches have been recognized in the literature as practical solutions that provide dynamic mechanisms for analyzing functional and non-functional properties against well-stated conditions, such as contractual conditions for trust [16].

[*]E-mail: antonello.calabro@isti.cnr.it
[**]E-mail: said.daoudagh@isti.cnr.it
[†]E-mail: eda.marchetti@isti.cnr.it

Vol. 32, No. 2, © The Author(s) 2024. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution NonCommercial-NoDerivs licence (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits non-commercial reproduction and distribution of the work, in any medium, provided the original work is not altered or transformed in any way, and that the work properly cited. For commercial re-use, please contact journals.permissions@oup.com
Advance Access published 16 March 2024 https://doi.org/10.1093/jigpal/jzae010

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

In this light, this paper focuses on designing and developing an integrated monitoring framework supported by techniques, methods and tools that can provide smart predictions about cybersecurity vulnerabilities and suspicious behaviors. In particular, the paper contributes to (i) providing a set of guidelines for the definition of a monitoring framework able to provide smart predictions about suspicious behavior or property violations; (ii) suggesting an application process for the execution of the monitoring activities; (iii) defining a reference architecture of the conceived framework; and (iv) describing a possible implementation that satisfies the defined guidelines by design.

The proposal of this paper extends the seminal work of [5] by providing means for intelligent predictions, adding new components able to manage different sources of knowledge and making user interaction and knowledge management easier. The leveraged framework, called hereafter Smart Monitoring Framework (SMF), provides a dynamic mechanism for the online analysis of functional and non-functional properties of an entity against well-stated conditions, such as contractual conditions for trust.

In detail, SMF collects events at different specification levels and from heterogeneous sources (such as applications, components, sensors or devices) and uses this data to infer complex patterns, each associated with a functional or non-functional property. Thus, each pattern represents an observed behavior that has to be compared with the trustable predicted one computed by the monitoring framework to detect anomalies, vulnerabilities or problems. As detailed in the remainder of this paper, the SMF can: (i) Collect and analyze data from different System of Systems (SoS) sources (e.g. applications, sensors, software and hardware components or devices). (ii) Assess the run time behavior of the SoSs (components or devices) based on the expected behavior rules. (iii) Promptly raise alarms in case of violations, anomalies or misbehaviors.

Considering the state of the practice, the SMF framework leverages it by:

- Providing the implementation of a dynamic, user-friendly and adaptable methodology for the specification of functional and non-functional properties. Therefore, it integrates and extends the proposals of [6] to enable holistic knowledge management and data sharing.
- Including a predicting mechanism for anticipating the behavior of a trustable ecosystem (components). Working with the monitoring facilities represents a dynamic means for anticipating trustable patterns.
- Providing an integrated mechanism for assessing the correctness of the run time executions of the ecosystem and its components against the collected prediction without knowing the source code structure;
- Providing a dynamic, user-friendly and adaptable methodology for managing the alarms, triggering the corresponding notifications and executing the associated countermeasures.

1.0.1 Outline

Section 2 introduces the main Research Guidelines (RGs) and the ambition of SMF. Section 3 presents the execution environment and the reference application process that enable satisfying the RGs. Section 4 describes the main components of SMF, whereas Section 5 provides details about two of them and briefly describes examples of SMF's usage. Finally, Section 6 concludes the paper, highlighting future work.

2 Research guidelines and ambition

The realization and development of the proposed SMF have been performed considering some general research guidelines. As a specific contribution of this paper in Section 2.1, the list of the

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

leading policies is presented because deemed helpful for developing any other similar proposals. Additionally, Section 2.2 discusses the purpose of the proposed SMF. More details about the guidelines' satisfaction will be provided later in this paper.

2.1 SMF Research Guidelines

The following Research Guidelines (RGs) have been identified:

RG1: Whitening the black-box assessment process. The realization of the SMF should rely on specific means for collecting internal execution data (white-box data) without knowing the source code structure (black-box data). Indeed, each SMF should focus on making the components and the ecosystems more 'transparent' for functional and non-functional properties assessment and prediction without revealing their internals. Data should be collected, preserving the principles of loose coupling and implementation neutrality.

RG2: Separating properties predictions and assessment. The implementation of the SMF should follow the principle of independence of the components. All the conceived elements should have a specific role and contribute to the overall quality, usability and effectiveness. However, the design of the SMF should let each of them work separately to better face different environments and application domains.

RG3: The proposed SMF should not be a 'yet-another-proposal.' The design of SMF leverages the current monitoring solutions by considering one or more of the following aspects: (i) including or leveraging some of the similar (open source) approaches; (ii) collaborating with other available monitoring solutions (possibly with minor adaptations); (iii) provides components that can be easily customized, enriched or modified; (iv) implementing the loose coupling principles.

RG4: Keeping the human in the loop. The SMF should allow customizing the monitoring activity and the implemented features for detecting violations or misbehavior. Thus, the provided components should be easily adaptable, leverageable or modifiable through user-friendly facilities. The implemented countermeasures should also be defined, keeping the human in the loop.

RG5: Including failure prediction methods. The SMF should include means for either smart prediction of malicious attacks or misbehavior or for anticipating evaluating functional or non-functional properties.

RG6: Common Vocabulary and Knowledge Management. The SMF should refer to specific terminology, vocabulary containing concepts and relationships. That allows any possible stakeholder to define, in every ICT domain, a set of practical and well-defined monitoring rules to reveal vulnerabilities and/or problems that a new device (or component) integration could arise.

2.2 SMF ambition and related work

At the state of the practice, there are three main trends for detecting or predicting run-time vulnerabilities or violations [16].

Using (previous) knowledge. In this case, previous data collections or past examples or failures are used to predict the output's quality. The proposals are usually based on either deep learning approaches as in [17] or rely on a separate system to monitor and predict a target model's failure (as in [14]) or on a perception system [15], or methods for prediction learning (as in [12]).

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

Using input data. This group includes monitoring methods based on the analysis of the stream of input data coming from different sources, such as sensors, components, devices, systems or models (as in [1–3]).

Using confidence estimations. In this case, confidence learning and uncertainty estimation are used for the output evaluation (as in [4, 8, 10])

Starting with the preliminary proposal of [5], the leveraged Smart Monitoring Framework (SMF) targets the second trend: using the input data for violation detection. The ambition is to implement the research guidelines listed in the previous section by providing facilities for the following main activities: (1) managing the heterogeneity of data (and events) producers; (2) interacting with different contexts and environments; (3) managing multiple different monitoring solutions (e.g. Esper[1] and Drools[2] instances); (4) dealing with multiple data storage (e.g. influxdb[3] or MySQL[4]); (5) providing communication facilities for speeding up the interaction and the composition (e.g. REST interfaces and JSON messages or JMS2JSON mediator); (6) scaling according to the components' availability and amount of data to be collected; (7) optimizing message processing and improving the quality of services by executing routing techniques or docker container deployment.

All those facilities have been developed according also to the principles specified in the Reactive manifesto,[5] which are (i) Responsiveness in guaranteeing a consistent quality of services; (ii) Resilience in trying to manage all possible exceptions and interruptions that may occur during the execution to provide a highly available system; (iii) Elasticity, in allowing the number of complex event processors and channels communication scales to avoid central bottlenecks; and (iv) Message-driven in letting all the messages asynchronous and loosely coupled between components involved in the evaluation.

3 Execution environment and application process

The following execution environment and application process of the SMF framework are defined to satisfy the research guidelines presented in Section 2.1. Before describing the environment in which the SMF can be executed, two specific terms need to be introduced: the System Under Monitoring (SUM), which indicates the device, the component or the system that is the target of the monitoring activity, and the Controlled Environment (CE) that refers to the environment in which the SUM is executed.

Following the *RG3: The proposed SMF should not be a 'yet-another-proposal'*, the SMF is developed independently from the CE and SUM. In particular, the monitoring activity can be executed in three different situations: (1) *Using simulation models:* the CE is executed using abstract models in a simulation context;(2) *Using a testing environment:* the CE is a testing environment where the components, directly interacting with the SUM, are either real or simulated or executed using stubs;(3) *Using a real context:* the CE and its components are executed in a real environment.

Figure 1 visualizes the environment in which the SMF can be executed, independent of the three described situations. Considering the *RG1: Whitening the black-box assessment process*, the required precondition for SMF execution is just the instrumentation of the CE and SUM with facilities (in

[1] Esper solution is available online at the following link: https://www.espertech.com/esper/
[2] DROOLS language can be found at: https://www.drools.org/
[3] influxdb database can be found at the following link: https://www.influxdata.com/
[4] The open source database MSQL is available at: https://www.mysql.com
[5] Reactive manifesto can be downloaded at the following link: https://www.reactivemanifesto.org/

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

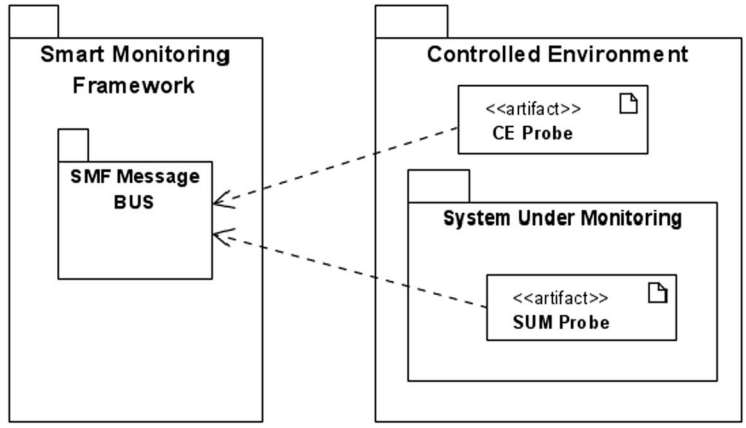

FIGURE 1. SMF Execution Environment.

Figure 1 and hereafter called probes [19]) for data (events) collection. The probes are pieces of code injected into the entities to send specific information about the execution.

In defining the SMF application process, the implementation of the *RG4: Keep the human in the loop* has been considered. In particular, the current SMF leverages the procedure proposed in [5] by offering a four-step application process that includes: Pre-setup, Artifacts Preparation, Finalization of the Pre-setup and Execution. The following details each step and its interaction.

Pre-setup. The Pre-setup phase deals with the activities necessary for setting up the execution environment. It includes the selection of functional and non-functional properties to be monitored. From the usability point of view, this phase also supports the definition of freezing/resuming actions and knowledge management.

Artifacts Preparation. This provides guidelines for instrumenting the CE and SUM with probes and facilities for managing prediction processes. This step includes facilities for probes definition and tools/artifacts for instrumenting CE and the SUM.

Finalization of the Pre-setup. The step consists of translating the functional and non-functional properties into executable monitoring rules and setting up the execution environment in which the instrumented CE and SUM can be run (or simulated) for monitoring purposes.

Execution. The execution step collects monitoring data to allow rules evaluation. It also manages the violation detection and the sending of the consequent alarms or notifications.

4 Smart Monitoring Framework

The SMF framework components are not strictly connected to the *RG2: Separating properties predictions and assessment*. Therefore, SMF offers the possibility to either deploy its components on different machines or clouds; or replace them with more performing components or services (as suggested by *RG3: The proposed SMF should not be a 'yet-another-proposal'*). This section briefly summarizes the main components of the Smart Monitoring Framework (SMF) introduced in [5]. As in Figure 2, SMF comprises five main components that collaborate to make behavioral or

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

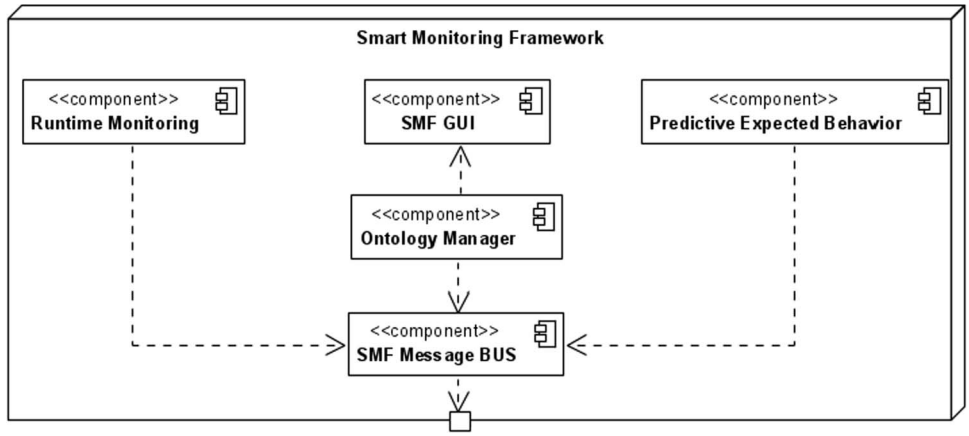

FIGURE 2. Smart Monitoring Framework (SMF).

vulnerability predictions: (1) Runtime Monitoring, (2) SMF GUI, (3) Predictive Expected Behavior, (4) Ontology Manager and (5) SMF Message BUS.

(1) Runtime Monitoring. It collects events from the probe injected into the SUM the CE and uses them to match the established functional and non-functional properties. In particular, Runtime Monitoring uses the predictions provided by the Predictive Simulation component to define specific rules about the expected CE or SUM behavior in the future. The Runtime Monitoring collaborates with the Ontology Manager to receive the functional and non-functional properties to be translated into rules and monitored during the execution. Finally, the Runtime Monitor receives the CE or SUM events through the SMF Message Bus. More details about Runtime Monitoring are provided in Section 5.2.

(2) SMF GUI. According to *GR4* and to keep the human in the loop, the GUI component lets the user interact with the SMF during the four phases of the application process described in Section 3). In particular, the SMF GUI lets the user interact with the Ontology Manager to request information and provide the necessary input. It also manages the stop, resume and saving of activity and data.

(3) Predictive Expected Behavior. This component targets *RG5: Including failure prediction methods* by offering features for predicting the behavior of the SUM or the CE. In particular, in line with the *RG3*, the predictions can be performed including different approaches: (1) using abstract models representing the executable abstractions of the ecosystem components (ICT systems, ICT system components and actors) and their interactions. This is often achieved by employing Digital Twins [5, 9], Business processes [7] or Smart Agents [13].(2) through approaches based on data mining, data visualization, algorithm clustering and neural networking or deep reinforcement learning [18]. These approaches focus on finding interesting patterns or trends in collected data or events that lead to a deviation from the expected behavior.[6]

[6]Dynatrace's Infrastructure Monitoring available at: https://www.dynatrace.com/

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

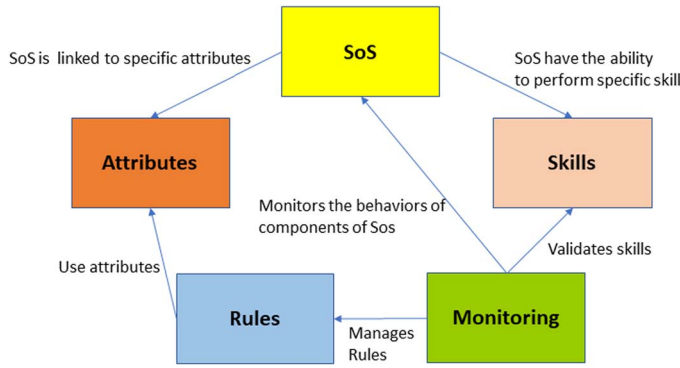

FIGURE 3. SMF Ontology Modules (reshaped from [11]).

(4) Ontology Manager. It manages the knowledge and provides facilities for classifying and categorizing the different Systems of Systems (SoSs), their devices and components, as well as the related skills and functional and non-functional properties. More details about Ontology Manager are in Section 5.1.

(5) SMF Message BUS. This component is the backbone of the SMF and manages all the communications between the framework components. Usually, two technologies can be considered: Java Message Service (JMS) Messages and REpresentational State Transfer (REST) interfaces exploited using Apache Artemis and Java Spring Boot with Thymeleaf.

5 Main SMF implementation aspects

This section focuses on the main improvements of the SMF concerning the preliminary version provided in [5]. In particular, it details the ontology used by Ontology Manager, the features of the Runtime Monitoring (introduced in Section 4) and some examples of usage.

5.1 Ontology

The ontology used by the Ontology Manager, hereafter called *SMF Ontology*, is depicted in Figure 3 and represents the evolution of an initial proposal, called MONTOLOGY (MONitoring onTOLOGY) described in [6]. The SMF Ontology focuses on making MONTOLOGY more modular, manageable and comprehensive to be suitable for different application domains and ecosystems and to satisfy better the guideline *RG6: Common Vocabulary and Knowledge Management*. As depicted in Figure 3, in the SMF Ontology, the MONTOLOGY initial proposal has been re-organized into five modules (SoS, Skills, Monitoring, Rules and Attributes), each containing a set of correlated concepts and relations between them. For a detailed description of this new proposal, we refer to [11].

Following the guideline *RG2: Separating properties predictions and assessment*, the SMF Ontology can be used in association with any prediction or monitoring facility. The SMF Ontology allows knowledge derivation reasoning and inference of new knowledge to help the different SoS stakeholders to gather focused and effective functional and non-functional properties of the CE or SUM or parts of them. In particular, the SMF Ontology connects the functional and non-functional properties provided in a non-formal language into a valuable formal specification for deriving

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

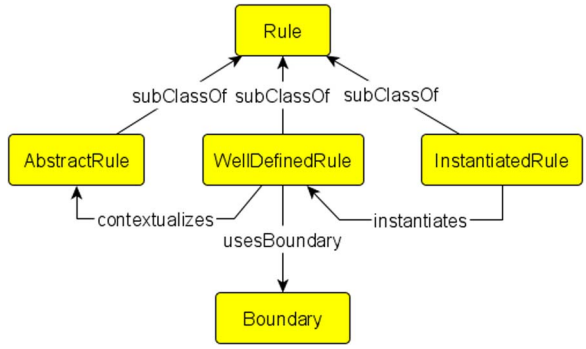

FIGURE 4. Rule Module (reshaped from [11]).

concrete monitoring rules. More details about this connection are provided in the remainder of this subsection by briefly describing the two mainly involved modules, namely *Rule* and *Monitoring* modules.

Rule Module. It provides a well-formed hierarchy useful for connecting non-formal functional and non-functional properties with formal ones. It includes the following sub-classes (see Figure 4): (1) *AbstractRule*,(2) *WellDefinedRule* and(3) *InstantiatedRule*.

The *AbstractRule* is a generic natural language description of the objective of the monitoring activity that is easily understandable by non-expert users. It is not yet instantiated within the execution context, domain data or specific constraints. An example of a functional property could be `the maximum number of established simultaneous connections between two components`. The abstract rule can be refined into the *WellDefinedRule*, i.e. a semi-structured rule ready to be translated to the destination language of the Runtime Monitoring component. Through the SMF GUI and in collaboration with the Ontology Manager (see Section 4), the users can provide specific values and boundaries about the execution context, CE or SUM. For instance, in the previous example, the user could set the boundary value for the maximum number of established simultaneous connections equal to 1. This information is collected into the *Boundary* class and expresses the applicability ranges of the rule. The *InstantiatedRule* is a rule translated into the language of the Runtime Monitoring component.

Monitoring Module. (Figure 5) The Monitoring module's structure includes four classes: *Calendar*, *Monitor*, *EntryPoint*, *Probes* and *Event*. Among them, the Monitor is the main class. It observes rules organized in the Calendar, i.e. an ordered set of rules (see Figure 5). Each Calendar can validate a specific skill defined in the Skills module at runtime. The Monitor has a specific EntryPoint used to communicate with the Probe. The Event represents the change of a state within a system. It is generated by the injected Probe and sent through a message call. According to RG1, it provides the information needed for monitoring activities without knowing what is happening inside the monitored entity and enables the whitening of the black-box assessment.

5.2 Runtime monitoring

The Runtime Monitoring component has been developed to target three of the six RGs mentioned in Section 2.2. In particular, it targets *RG1: Whitening the black-box assessment process*; *RG2:*

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

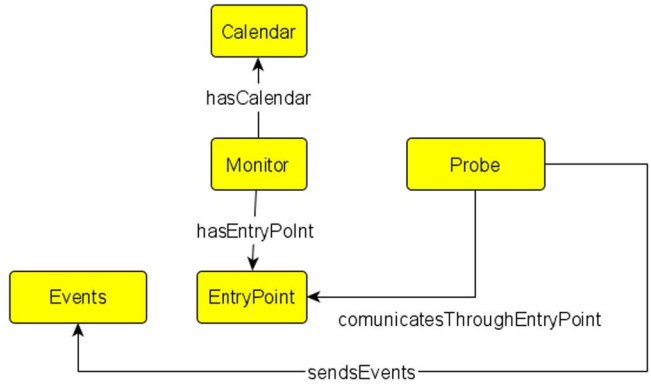

FIGURE 5. Monitoring Module (reshaped from [11]).

Separating properties predictions and assessment; and *RG3: The proposed SMF should not be a 'yet-another-proposal'*.

Runtime Monitoring has four main tasks: collecting specific events from the CE or SUM during the execution (as described in Section 3); receiving and translating into rules the predictions behavior of the various virtual or real CE or SUM entities; evaluating the rules on the bases of events receives during the execution; notifying possible rule violation or misbehaviors.

As depicted in Figure 6, Runtime Monitoring splits functionalities into independent modules that can be enacted according to the desired configuration/operational profile. The components can also be instantiated as stand-alone nodes or containers on top of a Docker architecture. This choice provides a more substantial decoupling and lets the components deploy on different machines for improving resource management. The **MonitorMain** is the launcher of the Runtime Monitoring. It oversees executing the components required for a specific profile in the correct order. It, therefore, allows managing the Runtime Monitoring lifecycle according to the Application Process described in Section 3. **ActiveMQ & MQTT Broker** executes an internal instance of a message broker and provides facilities for sending messages (events and requests) between the artifacts involved in a monitoring session. The current implementation supports two communication technologies: (1) ActiveMQ instance that implements the JMS Message Broker; and Mosquitto, i.e. an open-source message broker that implements MQTT protocol using publish/subscribe messaging techniques. **EventListener** component automatically executes the routing, and it listens on a specific channel by receiving event messages generated by probes injected in CE or SUM. The **Complex Event Processing (CEP)** is the core part of Runtime Monitoring and can be deployed in multiple instances on several nodes. CEP is a rule engine that analyzes the events generated by probes and correlates them to infer more complex events. If the event triggers no rule, the event is only collected into the event stream of the CEP. Complex events inference is based on a set of derived rules, i.e. 'if-then-else' grammar expressions that define sequences of attended or un-attended events patterns. We refer to [5] for details about the events' structure inside the monitoring component. **Rules Manager** includes generic rules templates (meta-rules) that can be instantiated during the execution of SMF for defining the monitored rules. The Rules Manager works with the CEP to load and unload the rules according to the events observed during the execution.

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

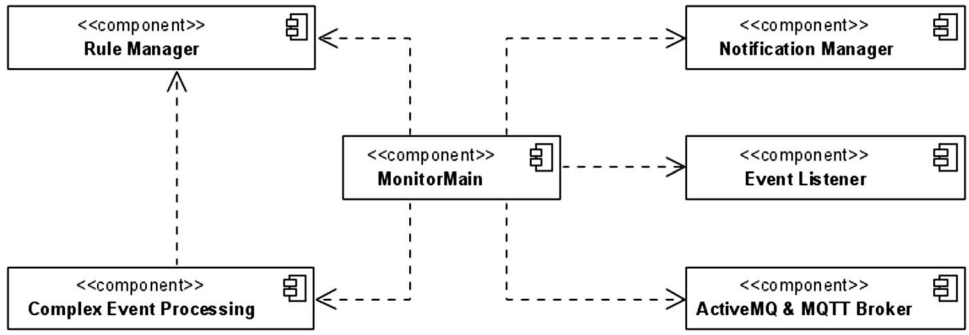

FIGURE 6. Runtime Monitoring Infrastructure.

5.3 Examples of usage of SMF

The proposed SMF is currently under development inside the BIECO project,[7] where we are using four actual use cases for SMF validation and assessment purposes. Specifically,, the adopted use cases will be on the following application domains: **UC1** represents a multi-robot autonomous navigation environment for intralogistics; **UC2** consists of an ICT Gateway for Smart Grids implemented; **UC3** deals with an AI Investment Platform; and finally, **UC4** models a Smart Microfactory.

For confidentiality reasons, we are not allowed to report details about the use cases evaluation here. However, in those experiments, the SMF's components have been instantiated using the following tools:

- For the Runtime Monitoring we use Concern monitor tool.[8]
- For the Ontology Manager, the implementation of the DAEMON tool [11].
- For Predictive Expected Behavior, the implementation of the Predictive Simulation tool as described in [9].

For more details, we refer to the collection of the public BIECO project deliverables.[9]

6 Conclusions

Pairing up Runtime Monitoring and Predictive Expected Behavior is pivotal for promoting trustworthiness in services and products that integrate new or available third parties or open-source components and systems to create value and guarantee business continuity. This paper contributed to this purpose in different aspects, first, by defining six Research Guidelines (RGs) to be used as a reference for designing and implementing a Smart Monitoring Framework (SMF) to detect suspicious behavior or property violations promptly. Second, by providing the execution process helpful in keeping the human in the loop during all the phases of SMF execution, and finally, by providing the architecture and the reference implementation of the SMF framework able to satisfy by design all the proposed RGs.

[7] More details about the BIECO project and the use cases can be found at: https://www.bieco.org/use-cases/
[8] Concern monitor tool is available under GPL3 at: https://github.com/ISTI-LABSEDC/ConcernMonitoringRest.git
[9] BIECO project Deliverables collection is available at: https://www.bieco.org/project-description/deliverables/.

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

	RG1	RG2	RG3	RG4	RG5	RG6
Environment & Process	✓		✓	✓		
SMF		✓	✓			
SMF: GUI				✓		
SMF: Ontology		✓				✓
SMF: Runtime Monitoring	✓	✓	✓			
SMF: Predictive					✓	

FIGURE 7. Mapping of RGs and SMF Proposal.

For completeness and clarity, Table 7 shows how the proposed SMF addressed the six RGs. In the table, the columns represent the conceived RGs, and the rows report the different contributions of our work. A mark shows when the SMF satisfies the RGs in the corresponding column.

As in the table, **RG1** (i.e. *Whitening the black-box assessment process*) has been satisfied by injecting specific probes in both the CE and SUM. Indeed, the injected probes provide the necessary data and information for monitoring activities without knowing what is happening inside the CE and the SUM. The SMF with modularity satisfies **RG2** (i.e. *Separating properties predictions and assessment*). Each SMF component has a specific responsibility and role within the monitoring execution, enabling the separation of concerns principle, i.e. decoupling prediction and assessment functionalities. Additionally, the SMF can be executed in three different situations (simulation models, a testing environment and a real context), guaranteeing execution flexibility. Further, the SMF Ontology can be used with any prediction or monitoring facility. Concerning **RG3** (i.e. *The proposed SMF should not be a 'yet-another-proposal'*), we conceived SMF to allow its customization with state-of-the-art solutions for each component. Furthermore, pairing up runtime monitoring and expected predictive behavior makes SMF unique and different from the currently adopted proposals. The SMF application process and the SMF GUI enabled **RG4**, i.e. *keeping the human in the loop*. In particular, the four-step application process and a user interface let the user interact with the SMF to perform all the envisioned phases. The **RG5** (i.e. *Including failure prediction methods*) has been addressed by including in the SMF framework specific facilities for predicting the expected behavior of the CE or SUM. Finally, **RG6** (i.e. *Common Vocabulary and Knowledge Management*) has been satisfied by defining the SMF Ontology that makes the previous MONTOLOGY proposal more modular, manageable and comprehensive.

However, future works are still possible despite the accuracy of investigating the defined research guidelines. Indeed, we are currently finalizing the implementation of the leveraged SMF and its components. Validation of the SMF is also an ongoing activity within the EU project BIECO to evaluate the proposal's effectiveness, collect improvements and consolidate the conceived RGs for further investigations and exploitation.

Acknowledgements

The authors thank the UNINOVA BIECO Partner's members for useful discussions about ontology enhancements. This work is partially supported by CyberSec4Europe H2020 Grant Agreement No. 830929 (https://cybersec4europe.eu/) and BIECO H2020 Grant Agreement No. 952702 (www.bieco.org).

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

References

[1] P. Antonante, D. I. Spivak and L. Carlone. Monitoring and diagnosability of perception systems. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 168–175. IEEE, 2021.

[2] P. Barsocchi, A. Calabrò, A. Crivello, S. Daoudagh, F. Furfari, M. Girolami and E. Marchetti. A privacy-by-design architecture for indoor localization systems. In *International Conference on the Quality of Information and Communications Technology*, pp. 358–366. Springer, 2020.

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

[4] A. Bertolino, A. Calabró, F. Lonetti and E. Marchetti. Towards business process execution adequacy criteria. In *International Conference on Software Quality*, pp. 37–48. Springer, 2016.

[5] A. Calabrò, E. Cioroaica, S. Daoudagh and E. Marchetti. BIECO runtime auditing framework. In *14th International Conf. On Computational Intelligence in Security for Information Systems and 12th International Conf. On European Transnational Educational (CISIS and ICEUTE), Bilbao, Spain, 22-24 September, 2021*, J. J. G. Prego, J. G. de la Puerta, P. G. Bringas, H. Quintián and E. Corchado., eds. Advances in Intel. Systems and Computing, vol. 1400, pp. 181–191. Springer, 2021.

[6] A. Calabrò, S. Daoudagh and E. Marchetti. MENTORS: monitoring environment for system of systems. In *Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021*, F. J. D. Mayo, M. Marchiori and J. Filipe., eds, pp. 291–298. SCITEPRESS, 2021.

[7] A. Calabro, F. Lonetti and E. Marchetti. Monitoring of business process execution based on performance indicators. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pp. 255–258. IEEE, 2015.

[8] A. Calabrò, F. Lonetti, E. Marchetti and G. O. Spagnolo. Enhancing business process performance analysis through coverage-based monitoring. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 35–43. IEEE, 2016.

[9] E. Cioroaica, S. Daoudagh and E. Marchetti. Predictive simulation for building trust within service-based ecosystems. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events, PerCom 2022 Workshops, Pisa, Italy, March 21–25, 2022*, pp. 34–37. IEEE, 2022.

[10] C. Corbière, N. Thome, A. Bar-Hen, M. Cord and P. Pérez. Addressing failure prediction by learning model confidence. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pp. 2898–2909. Curran Associates, Inc., 2019.

[11] S. Daoudagh, E. Marchetti, A. Calabrò, F. Ferrada, A. I. Oliveira, J. Barata, R. Peres and F. Marques. An ontology-based solution for monitoring iot cybersecurity. In *Internet of Things. IoT Through a Multi-Disciplinary Perspective*, L. M. Camarinha-Matos, L. Ribeiro and L. Strous., eds, pp. 158–176. Springer International Publishing, Cham, 2022.

[12] S. Hecker, D. Dai and L. Van Gool. Failure prediction for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1792–1799. IEEE, 2018.

[13] P. Jhunjhunwala, A. Zoitl, U. D. Atmojo and V. Vyatkin. Monitoring design pattern for distributed automation systems in iec 61499 and its formal modelling. In *IEEE 31st International Symposium on Industrial Electronics (ISIE)*, pp. 220–225, 2022.

[14] S. Mohseni, A. Jagadeesh and Z. Wang. *Predicting Model Failure Using Saliency Maps in Autonomous Driving Systems*, 2019.

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026

[15] S. Rabiee and J. Biswas. Ivoa: introspective vision for obstacle avoidance. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1230–1235. IEEE, 2019.

[16] Q. Rahman, P. Corke and F. Dayoub. Run-time monitoring of machine learning for robotic perception: a survey of emerging trends. *IEEE Access*, **9**, 20067–20075, 2021.

[17] J. Reich, D. Schneider, I. Sorokos, Y. Papadopoulos, T. Kelly, R. Wei, E. Armengaud and C. Kaypmaz. Engineering of runtime safety monitors for cyber-physical systems with digital dependability identities. In *Computer Safety, Reliability, and Security: 39th International Conference, SAFECOMP 2020, Lisbon, Portugal, September 16–18, 2020*, pp. 3–17. Springer, Berlin, Heidelberg, 2020.

[18] G. Stamatakis, N. Pappas, A. Fragkiadakis and A. Traganitis. Autonomous maintenance in iot networks via aoi-driven deep reinforcement learning. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–7. IEEE, 2021.

[19] A. Tundo, M. Mobilio, O. Riganelli and L. Mariani. Automated probe life-cycle management for monitoring-as-a-service. *IEEE Transactions on Services Computing*, 1–14, 2022.

Received 20 May 2022

Downloaded from https://academic.oup.com/jigpal/article/32/2/321/7627778 by rheumatology unit user on 27 January 2026