BIECO Runtime Auditing Framework

Antonello Calabrò[1], Emilia Cioroaica[2], Said Daoudagh[1], and Eda Marchetti[1]

[1] ISTI-CNR, Pisa, Italy
{antonello.calabro, said.daoudagh, eda.marchetti}@isti.cnr.it
[2] Fraunhofer IESE Fraunhofer-Platz 1 Kaiserslautern, Germany
emilia.cioroaica@iese.fraunhofer.de

Abstract. *Context:* Within digital ecosystems avoiding the propagation of security and trust violations among interconnected parties is a mandatory requirement, especially when a new device, a software component, or a system component is integrated within the ecosystem. *Objective:* The aim is to define an auditing framework able to assess and evaluate the specific functional and non-functional properties of the ecosystems and their components. *Method:* In this paper, we present the concept of predictive simulation and runtime monitoring for detecting malicious behavior of ecosystem components. *Results and Conclusion:* We defined a reference architecture allowing the automation of the auditing process for the runtime behavior verification of ecosystems and their components. Validation of the proposal with real use-cases is part of the future BIECO's activities.

Keywords: Auditing, Predictive Simulation, Runtime Monitoring, BIECO

1 Introduction

Digital Ecosystems, which are the emerging extension of Systems of Systems (SoSs), are increasingly influencing our societies on multiple levels. Around the typical interconnected and collaborating system, ecosystems contain various actors (such as organizations, developers or, users) that may have different collaborative and competitive goals, which significantly influences the dynamics within these ecosystems. The emerging dynamism requires more attention in the integration and collaboration of the Information and Communication Technology (ICT) components and devices which exercise various functionalities at the operational level to fulfill tactical decisions and satisfy higher-level strategic goals of a business within an ecosystem. For example, in a vehicles platoon scenario, multiple software smart agents executed on a vehicle platform implement diverse *operational goals* (OG) such as: sharing of context information or activating speed limits. The correct implementation of OG further up satisfy the fulfilment of *tactical goals* (TG) such as: creation of vehicles platoon that consist of vehicles driving together in the same direction. By satisfying the TG of driving together in close proximity, due to reduced air friction, fuel consumption is reduced as well, and this contributed to the fulfilment of *strategic business goals* within automotive digital ecosystems.

When a dynamic hierarchical analysis of goals within an ecosystem is performed, the end operational goals represent the core asset for virtual technical evaluation and testing directed towards prompt discovering of security and safety issues. A general unpleasant situation is that systems within an ecosystem could be affected by intended hidden faults inserted into software components, leading to the expression of malicious behavior that propagates to other interconnected parties within an ecosystem. Particularly, when a new device, a software component, or system component is integrated into an ecosystem, techniques for efficiently and effectively assess and prevent anomalies and dangerous situations are requires. This is especially important in context of safety critical digital ecosystems, formed around safety critical system, as in the above example from the autonomous domain. Among these techniques, a commonly adopted one is the audit of components through a monitoring system [12]. This technique 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 this context, an auditing framework can collect events at different levels of dynamic goal evaluation (strategic-tactical-operational) and from various systems and system components (including sensors). It uses the collected data to infer complex patterns that indicate specific functional and non-functional properties. The derived complex patterns represent the observed normal or abnormal behavior of the monitored system and its components. For this, the auditing framework should be able to: i) collect and analyze data coming from the different SoS sources (e.g., sensors, software and hardware components or devices); ii) assess the run time behavior of these SoS (components or devices); iii) promptly rise up alarms in case of violations; and moreover, an intelligent, complex monitoring can iv) put in place countermeasures if necessary.

Within the BIECO (Building Trust in Ecosystems and Ecosystem Component) project, we are working for advancing the state of the art on runtime monitoring by addressing the emerging needs for auditing techniques specific for interconnected ICT systems (including Cyber-Physical Systems) within digital ecosystems. In what follows, Section 2 presents related work. Section 3 presents the conceptual idea that provides the description of the behavioral process of the auditing framework. Section 4 presents in details the auditing process with its main components: predictive simulation and monitoring, and Section 6 concludes the paper and reports future work.

2 Related Work

Various techniques for qualitative and quantitative monitoring of CPS exist in the literature, as it is reported in [2] which presents a review of techniques and tools. In particular, the technique reported in of [3] describes offline monitors for identifying violations against policies. Another approach is the monitoring for validity presented in [4]. Similar to our approach, it utilizes internal simulation models of the robot, interacting actors and the environment. Our approach, different than other approaches, does not rely on selection of a set of predefined actions, but dynamically generates the actions exploiting the results of decision events. In particular, our validity monitoring activities involve selection of most probable environmental changes and iterations through multiple internal simulation models of the environment.

3 Conceptual Idea

In this section, we describe the concept of auditing for malicious detection, considering the situation in which a new component or device needs to be included into an ecosystems as an integral part of a SoS. In Figure 1, we depict the basic idea of the proposed auditing framework. As in the figure, for assessing the behavior of the new device during runtime, we consider two different parallel executions: On the right side, the execution of the device or component in a Controlled Environment (CE), i.e., in a real or realistic environment that can be exploited for validation and testing; on the left side: the execution of abstractions of the software components within a Simulation Environment (SE) fed with real-time data. For this, we rely on the Digital Twin (DT) representation of the component as presented in [7] independently derived from the component specification. DTs are abstract, trusted representations of components that can be executed in a simulation environment.

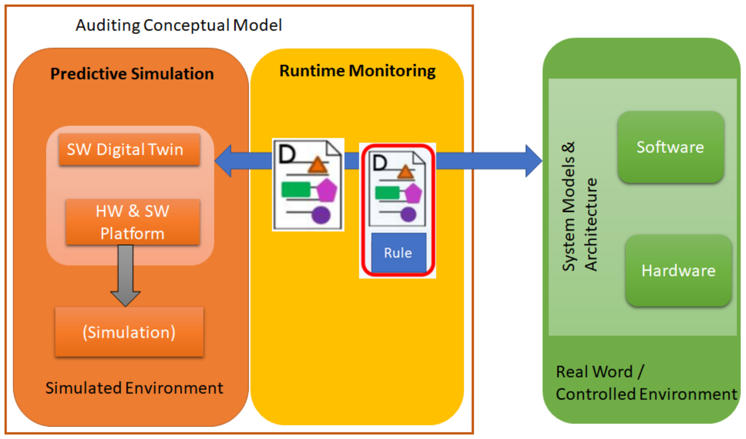

Fig. 1. Runtime Auditing Conceptual Model.

Following the prediction phase, the runtime monitor will collect both the predicted events computed by the simulated environment as well as real events coming from the CE and will compare them according to predefined rules with the scope of assessing the correctness of the behavior of the component itself against the behavior of the predictive simulation. If the behavior of the component executed in the real world is considered not trustworthy, then monitor can immediately rise up alarms so in order to trigger an operational fail-over behavior. For this, the behavior of the DT becomes the specification against which the behavior of the real world system is validated and therefore our solution represents a first attempt of using monitoring data coming from combined simulated and CE/real world software executions to promptly detect malicious behaviors and provide a timely reaction. Because the trustworthiness of the component is judged by executing the behavior of its DT in a simulated environment, the conformity between the DT's behavior and the specification needs to be assured pre-deployment. It is out of the scope of this paper going into detail of the conformity checking, and we refer to [5] for more information.

4 BIECO Auditing Process

In this section, we illustrate in more details the idea presented in the previous section, by explicitly referring to the auditing process implemented within the BIECO project.

As reported in Figure 2, two main participants are considered: the BIECO Framework and the Auditing component. In this section, however, we focus only on the description of the latter, which is in turn composed of two different roles: Predictive Simulation and Runtime Monitoring. Looking at the diagram, the user of Bieco framework can set up both the Predictive simulation and the runtime Monitoring. It could also set up the Controlled Environment, however because it is part of the BIECO Framework currently under development, we do not include it in this figure. Considering the Runtime monitoring, the procedural activities are: (1) `Monitor Set up`: during this phase the monitoring engine will establish all the communication needed from and to other components of the BIECO framework to which the information encapsulated in events will flow. In case of detected deviation, these events can be alarms or warnings that trigger a fail-over behavior. An event-based communication pattern guarantees a loosely coupled architecture with possibility of exchanging different components. (2) `Get Blueprints`: the information required for the monitoring activity are retrieved from the `Data Store`. The information includes the expected boundaries, i.e. admissible values and constraints for each event to be observed together with previously evaluated and trusted behavior of different ecosystem components. This information is used in deriving the initial rules to be monitored. (3) `Generate Monitoring Rules`: The blueprints are used for deriving a set of initial monitoring rules. In this activity the *Collection Data Object* generated from the task `Generate and Instrument Simulated environment and Digital Twin` contains possible names of the events that will be fired by the DT. This data together with blueprints are input for the `Generate monitoring rules` which generate the final rules needed for monitoring the runtime execution. The output of this phase is a set of rules against which the predicted behaviour of the DT will be evaluated together with the parameters provided within the blueprints. The blueprints contain design time evidence of trust that speeds up the runtime rule derivation process. (4) `Start Auditing Activity`: When the CE and the SE have been set up, the auditing activity can start. (5) `Listen for Events`: this activity collects and synchronizes the event coming from the CE and the SE (6) `Check Rules`: the event of the CE and SE are compared and the assessment of the Component performed (7) `Notify violation`: in case a rule violation an

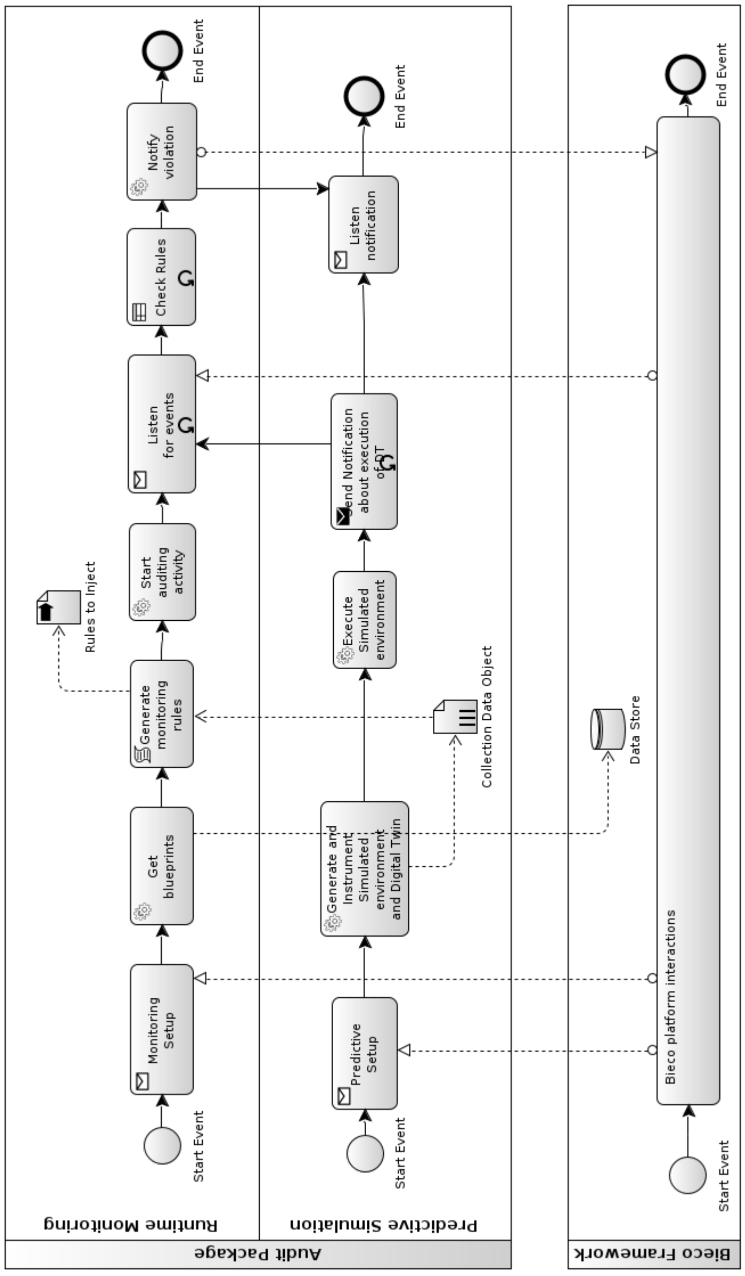

Fig. 2. Auditing Process.

alarm message is generated and sent to both the Controlled Environment (i.e.,) and the Predictive Simulation.

The conceived activities for the predictive simulation are: (1) `Set up the simulation scenario`: During this phase a virtual representation of a concrete technical situations in which behavior needs evaluation is created. The concrete technical situation is a composition of external environmental information and internal system configuration. During this phase, DTs are executed in collaboration with stubs that represent abstractions of the hardware components and/ or platform with which the software under evaluation interacts. (2) `Instrument the execution of Digital Twins (DT)`: DTs, which are abstract models fed with real time data are executed by a simulation environment at a faster speed than the wall clock. As a results, the predictive simulation provides an event signature that consists of the type, number and order of events who have been virtually validated. (3) `Send Notification about execution of DT` : during this phase an event signature is encapsulated into probes that the runtime monitoring can read.

5 Runtime Auditing Framework

In this section, we describe the auditing framework used within the BIECO project. This framework allows supporting the execution of the auditing process described in Section 4, and it is composed of two main modules: Predictive Simulation and Runtime Monitoring.

5.1 Predictive simulation

The predictive simulation component is in charge of running a virtual evaluation environment through execution of Digital Twins (DTs). DTs are abstract models representing expressed in executable abstractions of ecosystem components under evaluation (ICT systems, ICT system components such as software components). Because a software smart expressed can hide a malicious behavior, it is necessary to execute abstractions of these components instead. These abstractions are directed towards the scope of the evaluation, and for example: if functional interaction between components needs to be evaluated, DTs that represent the functional behavior of the components are executed. If on the other hand, scheduling behavior needs runtime evaluation, DTs representing time behavior of interacting components are evaluated instead. By restricting the logic of expressing behavior while keeping the overall actions and event decisions that cross the architectural boundaries, the components under virtual evaluation cannot detect whether they interact with real world or virtual world entities. Consequently, malicious behavior will be visible in the virtual world without affecting the real world.

In the linked predictive simulation in particular, the current state of the system is used to predict behavior in the future. For enabling detection of malicious behavior hidden within software components, a Domain Specific Language (DSL) that enables definition of control functions is currently under development. The predictive simulation works in collaboration with the Runtime Monitoring component.

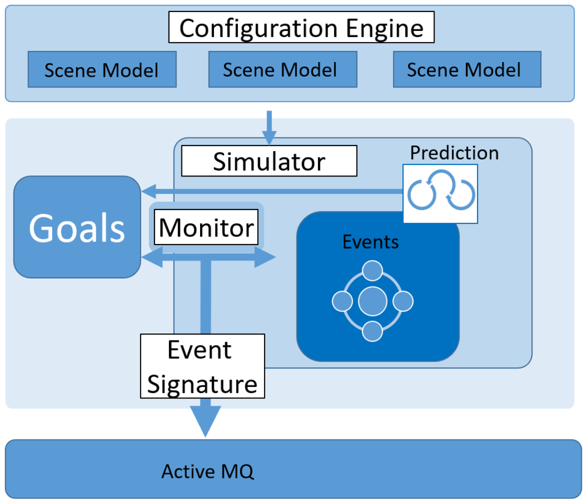

Fig. 3. Predictive Simulation.

The predictive simulation module is based on FERAL simulator [10] and comes as an extension of the platform presented in [6], which enables testing of automotive smart ecosystems in scenarios composed by virtual and real-world entities collaborating with each other. Typically used for the rapid development of architecture prototypes through coupling of simulators, simulation models, and high-level design models, FERAL enables the coupling of abstract simulation models with very detailed ones. In this way, the predictive simulation framework can integrate multiple abstraction versions of virtual agents and execute their behavior at different frequencies. Information from the predictive simulation is passed to the monitoring engine in the form of ActiveMQ messages.

Figure 3 depicts the main components of the predictive simulation environment. The instantiation of a technical setting for evaluation is performed through a configuration engine which sets scene models. The scene models contain simulation models that are executed by the FERAL simulator in a predictive manner. The results of the prediction are events that are monitored and validated w.r.t. goals. Internally, the prediction performs a validity monitoring of the raised events, and only based on evidence of trustworthy behavior, the signature events are sent to the monitoring engine described in the following section.

5.2 Runtime Monitoring

The component is in charge of setting up and managing monitoring component. The Runtime Monitoring is based on event messages. In particular, it enables the collection of specific events that flows during controlled environment, real execution and predictive simulation [12] among the different virtual and real entities (e.g., DT, sensors and ecosystem components) and it infers one or more complex events about the runtime execution (Complex Event Processing (CEP)).

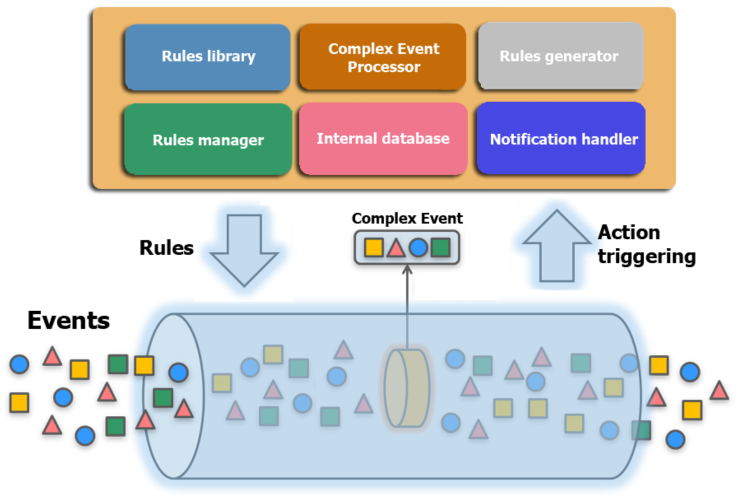

Fig. 4. Monitoring Infrastructure.

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. Thus, Runtime Monitoring includes a set of generic rules templates (meta-rules) that can be instantiated at runtime in concrete technical scenarios in which behavior is evaluated. The technical scenarios are a combination of both external environment and internal configuration of a system capable to accommodate different components over time. The events that trigger the execution of a rule are generated by a probe, i.e., a piece of code injected in the entities to be observed during the runtime execution able to notify the occurrence of the events to the monitor engine. At the technical level, the Runtime Monitoring is based on ActiveMQ [3] messaging protocol, and it also exposes a REST interface.

Figure 4 shows our reference monitoring architecture, whose main components are:

1) *Complex Event Processing (CEP):* i.e., a rule engine which analyzes the events generated by probes and correlates them to infer more complex events [1]. If the event triggers no rule, the event is just collected into the Event Stream of the CEP. Figure 5 depicts the structure of the events considered inside the monitoring component.

2) *Notification Handler:* i.e., a registry that keeps track of the requests for monitoring sent to the monitoring infrastructure. Once it receives the advice of a rule firing, pattern completion, or property violation from the CEP, the component sends the evaluation to the requester.

[3] https://activemq.apache.org/

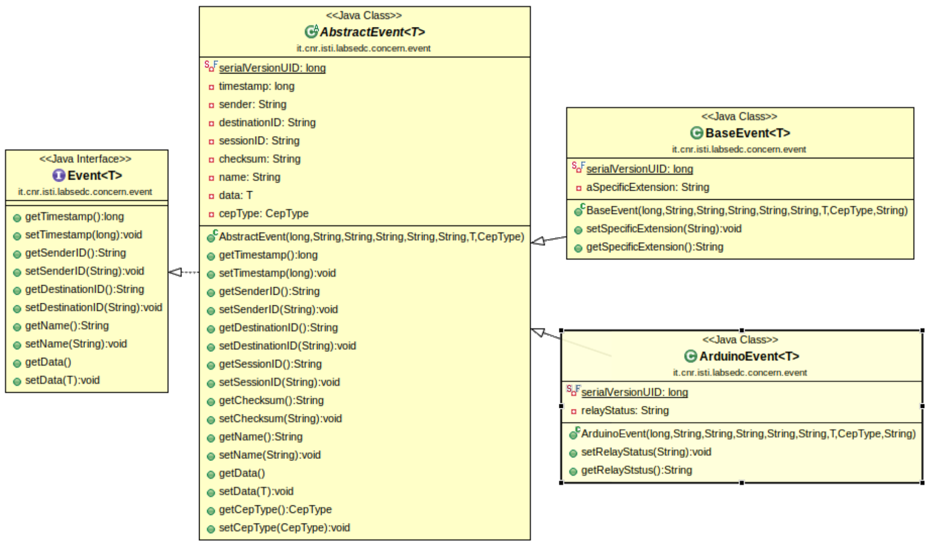

Fig. 5. UML Class Diagram of the Monitoring Event.

3) *Rules Generator:* i.e., the component in charge to generate the rules using the templates stored into the Rules Library. These rules are generated according to the specific properties to be monitored. A generic rule consists of two main parts: i) the events to be matched and the constraints to be verified are specified; and ii) the events/actions to be notified after the rule evaluation.

4) *Rules Library:* i.e., an archive of predetermined rules templates that will be instantiated by the Rules Generator when needed. A rule template is a rule skeleton, that is specified by the instantiation of a set of template-dependent placeholders. For the sake of completeness, the Rules Template Repository can also include sets of static rules.

5) *Rules Manager:* i.e., a component in charge to instruct the Complex Event Processor, to load and unload a set of rules. The Rule Generator will instantiate the template with appropriate values inferred from the specific properties to be monitored. Once a rule is instantiated this is loaded by the Rule Manager into the Complex Event Processor. The complex event detection process depends directly on the operations performed by the Rules Manager component.

6 Conclusions and Future Work

Detection of malicious behavior within digital ecosystems [9, 11] requires new techniques for runtime auditing [13, 8]. In this paper, we have introduced a reference auditing conceptual model for monitoring the behaviour of SoS within BIECO platform. The conceived auditing framework is composed of two main components: predictive simulation and runtime monitoring. As future work, we are planning to thoroughly validate our approach by considering three use cases within the BIECO project, coming from both academic and industrial contexts.

Acknowledgement

This work was partially supported by the EU H2020 BIECO project GA No. 952702 (www.bieco.org).

References

1. 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)
2. Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Lectures on Runtime Verification, pp. 135–175. Springer (2018)
3. Basin, D., Caronni, G., Ereth, S., Harvan, M., Klaedtke, F., Mantel, H.: Scalable offline monitoring. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification. pp. 31–47. Springer International Publishing, Cham (2014)
4. Blum, C., Winfield, A.F.T., Hafner, V.V.: Simulation-based internal models for safer robots. Frontiers Robotics AI 4, 74 (2017)
5. Brilliant, S.S., Knight, J.C., Leveson, N.G.: Analysis of faults in an n-version software experiment. IEEE Transactions on software engineering 16(2), 238–247 (1990)
6. Cioroaica, E., Kuhn, T., Bauer, T.: Prototyping automotive smart ecosystems. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE (2018)
7. Cioroaica, E., Kuhn, T., Buhnova, B.: (do not) trust in ecosystems. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). pp. 9–12. IEEE (2019)
8. 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, p. 117180A (2020)
9. Hidayanti, F.: Design and application of monitoring system for electrical energy based-on internet of things. Helix 10(01), 18–26 (2020)
10. Kuhn, T., Forster, T., Braun, T., Gotzhein, R.: Feral—framework for simulator coupling on requirements and architecture level. In: Formal Methods and Models for Codesign (MEMOCODE), 2013 Eleventh IEEE/ACM International Conference on. pp. 11–22. IEEE (2013)
11. 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)
12. Ullo, S.L., Sinha, G.R.: Advances in smart environment monitoring systems using iot and sensors. Sensors 20(11) (2020)
13. Won, M.: Intelligent traffic monitoring systems for vehicle classification: A survey. IEEE Access 8, 73340–73358 (2020)