# PatrIoT: IoT Automated Interoperability and Integration Testing Framework

Miroslav Bures
*Dept. of Computer Science, FEE*
*Czech Technical University in Prague*
Prague, Czechia
miroslav.bures@fel.cvut.cz

Bestoun S. Ahmed
*Dept. of Math. & Computer Science*
*Karlstad University*
Karlstad, Sweden
bestoun@kau.se

Vaclav Rechtberger
*Dept. of Computer Science, FEE*
*Czech Technical University in Prague*
Prague, Czechia
rechtva1@fel.cvut.cz

Matej Klima
*Dept. of Computer Science, FEE*
*Czech Technical University in Prague*
Prague, Czechia
klimama7@fel.cvut.cz

Michal Trnka
*Dept. of Computer Science, FEE*
*Czech Technical University in Prague*
Prague, Czechia
trnkamich@gmail.com

Miroslav Jaros
*Red Hat Czech s.r.o.*
Brno, Czechia
mjaros@redhat.com

Xavier Bellekens
*Dept. of Electronic and Electrical Engineering*
*University of Strathclyde*
Glasgow, United Kingdom
xavier.bellekens@strath.ac.uk

Dani Almog
*Software Engineering Dept.*
*Shamoon College of Engineering*
Be'er Sheva, Israel
almog.dani@gmail.com

Pavel Herout
*Dept. of CS and Engineering, FAS*
*University of West Bohemia*
Plzen, Czechia
herout@kiv.zcu.cz

***Abstract*—With the rapid growth of the contemporary Internet of Things (IoT) market, the established systems raise a number of concerns regarding the reliability and the potential presence of critical integration defects. In this paper, we present a PatrIoT framework that aims to provide flexible support to construct an effective IoT system testbed to implement automated interoperability and integration testing. The framework allows scaling from a pure physical testbed to a simulated environment using a number of predefined modules and elements to simulate an IoT device or part of the tested infrastructure. PatrIoT also contains a set of reference example testbeds and several sets of example automated tests for a smart street use case.**

***Index Terms*—test automation, automated testing framework, software testing, internet of things, model-based testing, interoperability testing, integration testing, simulation**

## I. Introduction

Paper accepted at **IEEE International Conference on Software Testing, Verification and Validation 2021, Testing Tools Track**, virtual conference, April 12–16, 2021.

**https://icst2021.icmc.usp.br/track/icst-2021-Testing-Tool-Track**

This research is conducted as a part of the project TACR TH02010296 Quality Assurance System for the Internet of Things Technology. The authors acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019 /0000765 "Research Center for Informatics". Bestoun S. Ahmed has been supported by the Knowledge Foundation of Sweden (KKS) through the Synergi Project AIDA - A Holistic AI-driven Networking and Processing Framework for Industrial IoT (Rek:20200067).

The current dynamic development of IoT systems requires the development of proper quality assurance and testing methods to ensure a sustainable level of quality [1]. While a number of testing frameworks and testbeds have been developed recently (e.g., [2]–[5]), IoT testing is characterized by specific problems, which raise the demand for a universal and open solution to allow scalable interoperability and integration testing of IoT systems.

In this paper, we present PatrIoT[1], an open automated testing framework that allows a flexible composition of testbeds from both simulated and physical devices in addition to parts of the infrastructure. The framework is developed in a joint academia-industry collaboration project between the Czech Technical University (CTU) in Prague and Red Hat, as the industry partner, with external consultancy cooperation of the University of West Bohemia, University of Strathclyde, UK, and Shamoon College of Engineering, Israel. The PatrIoT framework is accompanied by a set of example testbeds and automated tests created for modeled smart street IoT applications, which eases its applicability for industry engineers in various IoT projects.

## II. Related Work

Regarding the construction of IoT automated testing frameworks, the greatest effort is currently dedicated to the security and privacy aspects [6]. Several testing frameworks have been

Bestoun S. Ahmed is also with Dept. of Computer Science, Czech Technical University in Prague, Czech Republic

[1]https://patriot-framework.io/

This is a peer-reviewed, accepted author manuscript of the following article: Bures, M., Ahmed, BS., Rechtberger, V., Klima, M., Trnka, M., Jaros, M., Bellekens, X., Almog, D., & Herout, P. (2021). PatrIoT: IoT automated interoperability and integration testing framework. In 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 454-459) https://doi.org/10.1109/ICST49551.2021.00059

developed recently in these areas, e.g., [2], [3]. In addition, other studies focused on domain-specific frameworks for industrial IoT systems [7], healthcare systems [4], or smart cities [5], [8]. Additionally, Kim and Ziegler [9] addressed the challenges of IoT testing in the area of interoperability and conformance. For several industrial projects today, extensive testing strategies of devices and their interoperability and interactions are required. The magnitude of the problem increases with the number of various devices that communicate with each other, the versions of their firmware, and the various communication protocols used. This variety creates an extensive number of combinations that lead to a combinatorial explosion in the number of test cases.

Regarding the construction of testbeds, a significant number of prototypes have been reported. The prototypes mainly depended on the IoT domain and the purpose of testing. They vary from general testbeds [8], [10] to specific testing strategies such as protocol and scalability testing [11] or system security testing [12].

Test automation frameworks and testbeds, specifically focusing on interoperability and integration testing, are discussed in several studies. Wu *et al.* summarized currently available approaches to test Wireless Sensor Networks (WSNs) and include integration testing in their discussion [13]. However, the study is not exhaustive, as other possible approaches can be identified and employed.

Pontes *et al.* have recently presented a pattern-based test automation framework for integration testing of IoT systems. In their proposal, interoperability issues such as various communication protocols are tackled [3], but overall, the concept focuses rather on physical-devices-based testbed rather than on a mixture of physical devices and their simulations.

Another study by Rosenkranz *et al.* discusses issues arising from the heterogeneity of IoT systems and their interoperability testing and proposes a high-level concept of interoperability testing framework [14]. The concept focuses on the integration of standalone IoT systems and principally, simulation of individual components is not emphasized.

Considering the analyzed studies, we can conclude that the general support for IoT interoperability and integration testing that allows flexible scaling from simulations to physical testbed configuration is not discussed sufficiently in the literature.

## III. PatrIoT Framework

In our explanation of the PatrIoT framework, we first summarize its design principles. We follow with a description of its structure, enabling efficient and scalable automated interoperability and integration testing for IoT solutions.

### A. Design Principles

During the development of the PatrIoT framework, we considered the following design and conceptual principles to increase the applicability and flexibility of the framework.

1) The framework is developed as a flexible and open solution under a nonrestrictive open-source license Apache 2.0.
2) The framework employs current established open-source resources as much as possible. The goal of using these defacto standards is to ensure a better learning curve, better applicability, and better integration potential with other frameworks or parts of the testing infrastructure.
3) The framework is developed to be scalable to support a wide range of interoperability and integration testing, such as application programming interface (API) testing and complex end-to-end (E2E) integration testing.
4) The framework allows a flexible composition of physical and simulated items to deal with and substitute for the nonexistence of some IoT items that are not ready yet due to hardware cost, for example.
5) Simulation of the devices is solved as a modular system in which a particular device is used in the testing framework as a configurable module. Predefined sample devices are provided to the framework users to be used as templates.
6) The framework allows the connection of data inputs from Model Based Testing (MBT) tools via open interfaces. Namely, these interfaces are provided for combinations of test data inputs [15] and path-based test cases [16].
7) The framework supports the creation of well-structured and organized automated interoperability and integration tests, which minimize the maintenance overhead that may arise by keeping the test cases up-to-date when a System Under Test (SUT) is being changed. Similarly as in web applications [17] for instance, the IoT would is not an exception in this point.
8) The framework is constructed as a set of cooperating reusable modules that are connected via open interfaces. This gives the individual parts of the framework the possibility to be integrated with the existing parts of its users' testware.

### B. The Main Components of the PatrIoT Framework

The framework comprises several principal components. The responsibilities of these components are as follows:

- The **test runner** represents the main user interface so that the users may control the testing process in the framework. This component executes the defined automated test cases in a configured testbed and ensures the running of all auxiliary activities related to this execution. The test runner also unifies access to different parts of the testbed (e.g., network simulation manager or device emulator) into one scripting environment. The test runner is composed of two main subcomponents:
  - The **integration test runner** is responsible for the execution of the defined automated integration test suites.
  - The **performance test runner** is responsible for conducting a performance testing of the SUT. This process also involves metrics collection to support the evaluation of performance testing.

- The **testbed hub** is responsible for the orchestration of the physical and simulated devices' behavior with the infrastructural parts of the testbed. The hub transmits events and actions to the individual devices and elements of the testbed environment. For example, the change in the network topology, the evolution of the data generated by a simulated component, or the disconnection of the component from the network. The hub also ensures the monitoring of the test environment.
- The **network simulation manager** is responsible for the emulation of the network connections between the components. The component provides interfaces that allow the definition and deployment of selected network topologies and their management.
- The **device emulator** provides the capability for emulating various types of physical IoT devices as well as the generation of different kinds of data by probabilistic algorithms. This allows the emulation of the realistic behavior of such devices. We present more details for the simulated supported devices in Section III-D.
- The **collector** is responsible for data collection from the testbed environment (namely, network, device emulators and other components). The collector records details about these data to allow the rerunning of the test cases in the same conditions and scenarios to reproduce the detected defects. This data recording process is essential because the simulated IoT devices may produce a random torrent of data.
- The **provisioner** is responsible for defining, deploying, setting up, and initializing the testbed. The provisioner allows storing multiple configurations of the testbed and initializing selected configurations.
- The **reporter** is responsible for processing and presenting the test results and the events that occurred within the testbed during the testing process. This component aims to provide a unified reporting interface, which accepts the test results and events and states from the testbed to maintain the context information needed for effective defect analysis.

### C. The Network Infrastructure Simulation of the SUT

As mentioned previously, the PartIoT framework testbed can be physical parts of the network infrastructure and/or combined with a simulated network infrastructure. The components' configurations of this simulation are performed in the Docker environment. Here, the configuration of the physical IoT devices in the testbed is connected to the physical part of the network infrastructure. The virtual parts of the testbed (i.e., the simulated IoT devices, the simulated active elements of the network as gateways, and other infrastructural elements) are deployed as individual Docker containers. The individual links of the simulated network are configured as virtual links between the Docker containers or the Docker container and the interface to the physical part of the network. Within this system, a flexible scaling of the testbed configuration is achieved, which leads to an easy deployment mechanism by the Docker infrastructure.

### D. The Simulation of IoT Devices

The framework provides a set of reusable building blocks from which a particular IoT device simulation can be assembled. In its current version, the framework provides two principal types of simulation, the data provider and the actuator.

The **data provider** can be assembled from a set of predefined building blocks in three levels: data generator, data transformer, and connector. The *data generator* produces a stream of data, which simulates the data produced by a physical data provider (e.g., a sensor connected to the network). A particular flow of data is driven by a configuration based on a defined probabilistic function, for example, the data for the temperature in time or the GPS trajectory in time. Subsequently, the *data transformer* converts the stream of data produced by the data generator to a format in which the device provides the data to the IoT system (e.g., JSON format). Finally, the *connector* connects the simulated device to the rest of the IoT system and ensures the transmission of the data.

The **actuator** simulates a more complex IoT device, which reacts to the API calls from the rest of the system and returns the result or the feedback data after performing a particular action triggered by the API call. This simulated device can be assembled from a set of predefined building blocks in three levels: interface, internal state machine, and connector. The *interface* defines the API of the simulated device in a format in which the device communicates with the rest of the IoT system (e.g., JSON or SOAP). The *internal state machine* defines a simplified behavior of the simulated component, which reacts to the device API calls and responds by a particular data that is provided as the output of the particular API. The *connector* connects the simulated device to the rest of the IoT system and ensures the ability to connect to the API of the simulated component.

In addition to the building blocks in the above categories, the framework contains a set of sample predefined simulated devices. These samples include several simulated data providers as motion detection sensors, light level sensors, CO and $CO_2$ levels sensors, temperature sensors, and fire outbreak detectors. Additionally, several examples of simulated actuators for smart-home appliances are available.

### E. Implementation Details

In the implementation of the framework, we followed the rule to reuse and utilize the already established open-source components. We used a modified version of the JUnit 5[2] framework. Since the current version of the JUnit framework is insufficient for our requirements, we added several extra features to the framework to specifically support automated interoperability and integration testing. These features involve additional support for the synchronization and orchestration,

[2]https implementation integration test runner://junit.org/junit5/

such as the implementation of warning state in the automated test, interruption possibility in a situation where the test flow is broken, and more flexible orchestration possibilities of the test step flow.

We implement the performance test runner using the already established open-source community project Perfcake[3] to integrate and adapt with the PatrIoT framework for IoT testing specifics. The network simulation manager is implemented as a component to control the Docker[4] container deployment process and set up the respective network interfaces for the deployed containers. The containers are controlled by a rich API written in Java that allows the user to specify in the code how the software should be built and deployed. It also provides an API for the network topology definition, which spawns several virtual networks interconnected by software routers and connects the SUT to appropriate places. Device emulators and all other components of the framework are implemented as open-source Java 8 classes in a modular structure that allows future flexible extensions of the framework and its integration with third-party testing tools and environments. The collector component provides an open interface to Elasticsearch[5] solution to allow further processing, analysis, and visualization of the test results and related context information from the testbed. The reporter component is created by an adaptation of the current Maven Surefire plugin[6], which is currently used for test reporting in the JUnit platform.

## IV. Industrial application

The PatrIoT framework is currently being employed in several IoT projects, which confirm the applicability of the proposed concept and technical solution. In this section, we summarize (1) the general feedback from these projects, (2) results from the application of the PatrIoT in the IoT middleware communication infrastructure Active Messaging Queue (AMQ Online), a product of Red Hat, which is employed in a number of current solutions, for instance by Bosch, Swiss Federal Railways (SBB), British Petrol or Intermountain Healthcare, and, (3) we describe a demo project of a smart street system, which documents capabilities of the framework and serves its users to learn the framework and apply it.

### A. General Feedback

The general feedback from the pilot applications confirms the very good flexibility of the framework in the composition of a mixed testbed, which is composed of physical and simulated IoT devices, as well as very good scalability between these two options. The fact that the framework is built using JUnit as its basic test runner component also creates several benefits, as the pilot projects show. As JUnit is known as the defacto standard for unit testing and close-API integration tests, the framework is applicable to a wide spectrum of automated tests, including pure unit tests, various types of integration tests, and user-interface-based automated tests using the JUnit framework as its runner.

As examined during the projects, the heterogeneity of the programming languages (Java as the core framework language) versus other languages, e.g., Python, in the case of rescue mission planning and management systems, limits the application of the PatrIoT for the classic unit tests of the SUT; however, for integration testing and user-interface-based automated tests, the capabilities are not limited. In such cases, individual SUT devices and modules act as black boxes, exposing APIs that are exercised by the automated tests run by the PatrIoT framework.

The necessity to understand the tested API and connect to a tested device or module by an available interface represents, based on the experience from the case studies, a crucial key point, allowing effective testability of the individual system parts. However, this is a general issue in IoT test automation that cannot be influenced by the test automation style or concept.

Limits of the framework might be observed in the case of event synchronization of the physical and simulated devices. The framework provides options for synchronization of events in the IoT system during the test via test runner and testbed hub components; however, for the more complex and precise synchronization of events, the developer of the automated tests needs to handle the behavior explicitly in the automated test code. More extensive support for the complex synchronization cases is included in the development roadmap of the framework.

### B. IoT Middleware Communication Infrastructure

AMQ Online is Red Hat product for managed, self-service messaging on Kubernetes and OpenShift. AMQ Online can run on user-provided infrastructure or in the third party provided cloud solutions and simplifies running a messaging infrastructure for IoT solutions. AMQ Online can be used for many purposes, such as moving an IoT messaging infrastructure to the cloud without depending on a specific cloud provider or building a scalable messaging backbone for an IoT system. The solution is employed by several end users in their solutions, for instance, Bosch, Swiss Federal Railways (SBB), British Petrol and Intermountain Healthcare.

During the fourteen months of the pilot project of PatrIoT application in the AMQ Online system, the framework was adopted by AMQ development and testing team and included in its test automation infrastructure, serving as a complement to established unit tests. Automated integration tests for AMQ Online use cases were prepared, including close-API integration tests as well as complex integration scenarios.

The initial effort of AMQ development and testing team needed to get familiar with the framework was 6 Man-days (MDs), including preparation of the test automation plan for new integration tests.

We evaluated two latest releases of the AMQ Online product: release 1.4.0 (denoted as REL1 further) and release 1.5.0 (denoted as REL2 further).

[3]http://perfcake.org/
[4]https://github.com/docker
[5]https://www.elastic.co/products/elasticsearch
[6]https://maven.apache.org/surefire/maven-surefire-plugin/

Over the pilot project run, compared to previous manual execution of integration tests, requirements coverage increased by 35% for REL1 and further increased by 24% for REL2. Increase in line and branch coverage of the code was slightly increased; however, due to the previous high level of code line coverage, which was 92%, additional execution of some code lines by new integration tests was not relevant to evaluate.

The time needed to prepare PatrIoT-based automated integration tests was 14 MDs in REL1 and 3 MDs in REL2. In REL 1, 25 complex integration test scenarios were created, consisting of 173 test steps. For REL 2, additional seven complex integration test scenarios were created, consisting of 56 test steps.

In both of individual releases, three builds with automated regression tests were conducted. The effort needed for one round of automated tests using the PatrIoT framework was 2 MDs. Most of this effort spent on failure analysis and defect reporting. In contrast, the effort needed to one round of regression tests conducted by the previous manual way of integration testing was 8 MDs.

The collected data verify the economic viability of test automation using the PatrIoT framework - Test automation in one release gained 36 MDs savings, which overweight the total 17 MDs needed for the preparation of the tests.

Regarding the effectiveness of the PatrIoT-based automated tests, compared to the 18 total regression defects detected by unit tests and manual integration testing, eight new defects were detected by PatrIoT in REL1 and 13 in REL2. By the severity of these defects for both releases, they included two blockers, eleven critical defects, one major defects, and six minor defects.

Considering the actual state of AMQ Online, which is well covered by unit tests and regular regression tests discover tens of defects maximally, the application of PatrIoT automated integration tests was effective in finding new defects present in the system.

### C. Demo Smart Street Testbed

Discussed features of the framework are also documented in another project, the demo smart street testbed, resulting in a set of open examples that are available for the framework's users. The examples further demonstrate the capability of the framework to flexibly scale from a purely physical testbed to a completely simulated IoT infrastructure. The most important case is when the testbed is mixed from physical and simulated devices.

The smart street model is composed of a set of smart houses (each represented by a set of sensors and actuators), gateways of the smart houses, and a controlling system for the smart street. They are implemented as a server service. The physical device of the smart house model contains actuators for central heating, air conditioning, lights, home sound system, opening the garage doors, and automated opening of windows. Moreover, it contains sensors for room temperature, air humidity, and an RFID tracking system. The simulated device of the smart house, in addition to these devices, comprises more actuators, such as automatic curtains on the windows, opening the house gate, and home appliances (e.g., a coffee machine) connected to the network. Additionally, the set of sensors is enriched by other devices for motion detection, light level, CO and $CO_2$ levels, outside temperature, and fire outbreak detectors. To make the testbed example available in the framework and independent of a real smart home deployment, a physical model of the smart house is available (its prototype is presented in Figure 1).

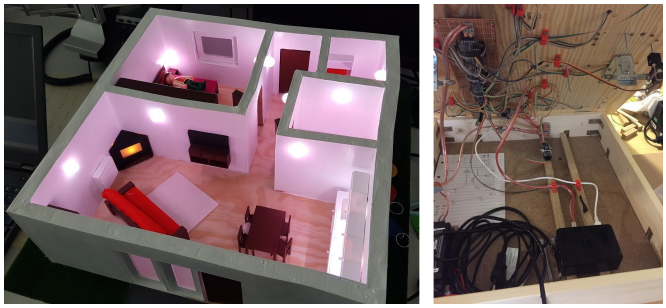

Fig. 1. A prototype of the physical smart house model

Table I presents a set of available sample testbed configurations within the framework. We presented the possible scalability of the framework to support testbeds from a purely physical configuration to purely simulated testing environments.

TABLE I
SAMPLE TESTBED CONFIGURATIONS AVAILABLE IN THE PATRIOT FRAMEWORK

| ID | Configuration | Testbed Type |
|---|---|---|
| #1 | One smart house, all devices physical, physical gateway | Purely physical testbed |
| #2 | One smart house, all devices physical, simulated gateway | Mixed testbed |
| #3 | Two smart houses and smart street server. House A: all devices physical, physical gateway. House B: all devices simulated, simulated gateway, smart street system simulated | Mixed testbed |
| #4 | Two smart houses and smart street server, all devices and gateways of House A and House B simulated, smart street system simulated | Purely simulated testbed |

The testbeds are accompanied by a set of automated sample test cases that evaluate a set of defined use cases of the smart street. The templates also demonstrate the recommended reference architecture for creation of automated tests.

## V. DISCUSSION

The concept of universal and flexible test automation and testbed construction framework is promising. There are two issues to be discussed in this context. Conceptually, a question can be raised if considerable flexibility in mixing the physical devices testbed with the simulated devices is needed, and if the problem can be solved by a simple strict application of the V-model. In other words, the individual parts can be tested at the unit level, integration interfaces at the unit integration level, and then the units are combined, and the final configuration

can be tested end-to-end using a lighter testbed configuration. This approach sounds like a logical option, and several studies reported this approach (e.g., [1], [6], [18]). Our industry project observations from interviews with 15+ IoT solution providers suggest that such an approach can reach its limits. The premise "if units work and individual interfaces also work, then the E2E process shall also work" is not simply valid, as many practitioners have documented numerous examples of actual tricky interoperability and integration defects.

Another concern may be raised regarding the construction of a universal test automation framework for IoT systems. Considering the specifications of various systems, various protocols are currently employed, and the nature of the individual systems and the variety of user devices may make this goal seem too ambitious. However, this concern can be minimized by the open architecture of the testing framework to allow the connection of various devices and infrastructural parts by adding proper communication adapters between the test cases and the devices. Additionally, adding new types of simulated devices, which are configurable from predefined building blocks, will minimize that concern. Such flexibility is the goal of the PatrIoT framework; however, to be objective, it may reach its limits for some IoT systems or their specific parts.

## VI. Conclusion

The increasing demand for effective interoperability and integration testing of IoT solutions and the issues related to the testbed design and construction also implies a demand for open and flexible test automation frameworks to support this process. The construction of a universal test automation framework is a significant industrial challenge. The related problems start with the composition of the testbed. As it might be unrealistic to create a testbed that is composed of solely physical devices, the simulation of the devices arises as a prospective option. The construction of these frameworks and simulators can naturally tend to a proprietary solution with low reusability of the framework. Another possible issue is low-level flexibility to enable the combination of physical and simulated devices. All these issues are addressed by the PatrIoT framework, which we presented in this paper. The framework allows the creation of a flexible environment to construct a testbed dynamically from a mixture of physical devices and infrastructure parts and particular device simulation. The simulators can be configured using predefined modules of the framework. In this process, a set of predefined examples can be used.

The pilot application of the framework showed the economic effectiveness of the created automated tests in terms of detected defects as well as the economic viability of the test automation process. As part of our future work, we are extending the framework to support even more complex cases of synchronization of events during the tests. We are also adding more modules that allow the creation of simulators of various IoT devices. The next prospective direction is adding more capability of the framework for agile testing means, as we successfully tried for web-based systems [19].

## References


[1] E. J. Marinissen, Y. Zorian, M. Konijnenburg, C.-T. Huang, P.-H. Hsieh, P. Cockburn, J. Delvaux, V. Rožić, B. Yang, D. Singelée, I. Verbauwhede, C. Mayor, R. van Rijsinge, and C. Reyes, "Iot: Source of test challenges," in *2016 21th IEEE European Test Symposium (ETS)*, May 2016, pp. 1–10.

[2] S. K. Datta, C. Bonnet, H. Baqa, M. Zhao, and F. Le-Gall, "Developing and integrating a semantic interoperability testing tool in f-interop platform," in *2018 IEEE Region Ten Symposium (Tensymp)*. IEEE, jul 2018. [Online]. Available: https://doi.org/10.1109%2Ftenconspring.2018.8691994

[3] P. M. Pontes, B. Lima, and J. a. P. Faria, "Izinto: A pattern-based iot testing framework," in *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops*, ser. ISSTA '18. New York, NY, USA: ACM, 2018, pp. 125–131. [Online]. Available: http://doi.acm.org/10.1145/3236454.3236511

[4] S. Sicari, A. Rizzardi, L. Grieco, G. Piro, and A. Coen-Porisini, "A policy enforcement framework for internet of things applications in the smart health," *Smart Health*, vol. 3-4, pp. 39–74, sep 2017. [Online]. Available: https://doi.org/10.1016/j.smhl.2017.06.001

[5] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for IoT smart city experiments," in *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE, sep 2016. [Online]. Available: https://doi.org/10.1109/isc2.2016.7580875

[6] J. Kiruthika and S. Khaddaj, "Software quality issues and challenges of internet of things," in *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Aug 2015, pp. 176–179.

[7] S. Lee, S. Lee, H. Yoo, S. Kwon, and T. Shon, "Design and implementation of cybersecurity testbed for industrial IoT systems," *The Journal of Supercomputing*, dec 2017. [Online]. Available: https://doi.org/10.1007/s11227-017-2219-z

[8] J. Lanza, L. Sánchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, and V. Gutiérrez, "Large-scale mobile sensing enabled internet-of-things testbed for smart city services," *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, p. 785061, 2015. [Online]. Available: https://doi.org/10.1155/2015/785061

[9] E. E. Kim and S. Ziegler, "Towards an open framework of online interoperability and performance tests for the internet of things," in *Global Internet of Things Summit (GIoTS), 2017*. IEEE, 2017, pp. 1–6.

[10] J. H. Huh, D. H. Kim, and J.-D. Kim, "Newsbed: The internet of things testbed platform," in *2017 International Conference on Information Networking (ICOIN)*, Jan 2017, pp. 492–494.

[11] V. F. Rodrigues, E. Correa, C. A. da Costa, and R. da Rosa Righi, "On exploring proactive cloud elasticity for internet of things demands," in *2017 XLIII Latin American Computer Conference (CLEI)*. IEEE, sep 2017. [Online]. Available: https://doi.org/10.1109/clei.2017.8226417

[12] J. Choi, Y. In, C. Park, S. Seok, H. Seo, and H. Kim, "Secure IoT framework and 2d architecture for end-to-end security," *The Journal of Supercomputing*, mar 2016. [Online]. Available: https://doi.org/10.1007/s11227-016-1684-0

[13] J. Wu, W. Jiang, Y. Mei, Y. Zhou, and T. Wang, "A survey on the progress of testing techniques and methods for wireless sensor networks," *IEEE Access*, vol. 7, pp. 4302–4316, 2019.

[14] P. Rosenkranz, M. Wahlisch, E. Baccelli, and L. Ortmann, "A distributed test system architecture for open-source iot software," in *Proceedings of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems*, ser. IoT-Sys '15. New York, NY, USA: ACM, 2015, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/2753476.2753481

[15] B. S. Ahmed, K. Z. Zamli, W. Afzal, and M. Bures, "Constrained interaction testing: A systematic literature study," *IEEE Access*, vol. 5, pp. 25 706–25 730, 2017.

[16] M. Bures and B. S. Ahmed, "Employment of multiple algorithms for optimal path-based test selection strategy," *Information and Software Technology*, vol. 114, pp. 21–36, 2019.

[17] M. Bures, "Metrics for automated testability of web applications," in *Proceedings of the 16th International Conference on Computer Systems and Technologies*, 2015, pp. 83–89.

[18] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2014, pp. 417–423.

[19] M. Bures, K. Frajtak, and B. S. Ahmed, "Tapir: Automation support of exploratory testing using model reconstruction of the system under test," *IEEE Transactions on Reliability*, vol. 67, no. 2, pp. 557–580, 2018.