# Observing Interoperability of IoT Systems Through Model-Based Testing

Koray Incki(✉) and Ismail Ari

Department of Computer Science, Özyeğin University, Istanbul, Turkey
ben@korayincki.com

**Abstract.** Internet of Things (IoT) has drastically modified the industrial services provided through autonomous machine-to-machine interactions. Such systems comprise of devices manufactured by various suppliers. Verification is a challenge due to high heterogeneity of composing devices. In this paper, we present initial results of model-based interoperability testing for IoT systems to facilitate automatic test case generation. We utilize messaging model of Constrained Application Protocol so as to deduce complex relations between participating devices. We use Complex-Event Processing (CEP) techniques in order to streamline the verification process after generating proper runtime monitors from sequence diagrams. We demonstrate our solution on a fictitious healthcare system.

**Keywords:** Internet of Things · Model-based testing
Constrained-Application Protocol · Runtime verification
Complex-event processing

## 1 Introduction

IoT presents a new computing phenomenon for such devices that are smart yet resource-constrained devices. Those devices involve heterogeneous day-to-day smart objects [1], which aim to seamlessly construct new services and applications through untethered autonomous machine-to-machine (M2M) collaborations. Interoperability is a major challenge in achieving such a goal, as there might occur unprecedented interactions between those objects. It is an issue in such systems of systems that are composed of subsystems with various communication protocols, application interfaces. Such interoperability as in communication layers involves protocol specific interoperability; for example, CoAP-based devices must be interoperable with respect to the CoAP standard [4,14].

IETF task force CoRE (Constrained RESTful Environments) [3] has promoted adoption of service-oriented in IoT domain with the introduction of an application layer protocol. The Constrained-Application Protocol (CoAP) [4] presents a RESTful-like programming environment that not only helps to develop such systems, but also raises new challenges in providing for interoperability between endpoints. In this paper, we attack the interoperability problem of application layer interfaces in CoRE IoT domain.

G. Fortino et al. (Eds.): InterIoT 2017/SaSeIoT 2017, LNICST 242, pp. 60–66, 2018.
https://doi.org/10.1007/978-3-319-93797-7_8

IoT systems are intrinsically hot swap systems, such that an endpoint can be exchanged with another endpoint providing (supposedly) the same services with the same address. But, endpoint configuration can have flaws before swapping. Thus, this might cause runtime failures even though the overall system design is verified in the development phase. Thus, we believe that a runtime solution for interoperability testing is a necessity in IoT domain.

IoT systems usually consist of commercial-off-the shelf (COTS) products with nearly no knowledge of internal implementation details, we promote a black-box testing approach for providing interoperability. We propose that a model-based testing approach that leverages the RESTful-like application layer interaction model of CoAP-based IoT systems should facilitate interoperability testing efforts. Model-based testing (MBT) has been utilized in several domains [8]. We demonstrate the applicability of MBT in IoT domain through implementation of a case study with Papyrus [5] modeling tool. Our previous work on runtime verification of IoT systems [6] has demonstrated that an IoT system can be described in terms of simple events occurring in the system. Thereby, we proposed a verification approach that utilized complex-event processing (CEP) technique. In this paper, we further that research with a MBT solution that allows automatically generating test cases from sequence diagrams in a UML model.

The paper is organized such that Sect. 2 discusses recent research on interoperability testing, Sect. 3 provides our solution framework, and Sect. 4 elaborates on the implementation. In Sect. 5 we conclude with a discussion and future work.

## 2 Related Work

A black-box testing approach for assuring interoperability assumes that the individual components are thoroughly tested by the manufacturer. But, when it comes to the complexities of the integrated heterogeneous system of systems, the runtime actions of the system might be overlooked with black-box testing. In [8] Wu et al. proposes using Unified Modeling Language (UML) [7] to express the expected behavior of a component-based software. They utilize interaction diagrams to capture functionality expected from the system. They explain how UML interaction diagrams can be used to extract the context-dependence and content-dependence relations so as to use in deciding if test cases are comprehensive or not. The research doesn't provide any guidance through implementation nor the automation of a model-based testing approach.

Internet community is majorly built around web services concept, thus interoperability of those distributed and heterogeneous services is an ongoing challenge. Bertolino et al. [10] proposed an audition framework for solving this problem. They extend the UDDI registries so that the services registered to a directory is audited before it is registered. Thereby, they validate the claimed behavior of the service before such services with the same UDDI registry can collaborate with proclaimed service contracts. This is a solid contribution in service registry coordination, however it lacks to observe the runtime behavior of services.

In [13], Smythe discusses that using a the modeling approach in development of a distributed service oriented system facilitates both implementation and testing efforts. The interoperability tests are automatically generated through a series of XML Metadata Interchange (XMI) transformations over a UML model of the system. Our approach also use XMI transformations in order to facilitate runtime EPL statements for interoperability testing.

In [14] authors proposes a new solution for certification of products, which involves conformance testing of IoT devices with respect to CoAP standards [4]. In their approach, they first record the live network traffic, and save them in files for post processing. When the system test run finishes, they collect those record files and apply post mortem tests on those logs so as to find any deviation in the CoAP communication primitives from the standard specification. The test cases are prerecorded according to the standard specification. Since, they operate on recorded log files, the approach does not scale well to runtime (online) interoperability testing. Moreover, they focus solely on protocol implementation interoperability, so the solution does not scale well for application specific interaction models.

## 3 Model-Based Testing for Interoperability

Software intensive systems has increasingly being developed with component-based architectures [9]. IoT systems are no exception to that adoption in the industry. Particularly, application layer protocols such as CoAP have made it possible to treat such systems purely as service-oriented systems. Monitoring of services in SOA systems has been valuable for post mortem analysis [12]. Thus, we will define how model-based testing approach can be used for describing the service interactions, and consequently facilitate interoperability testing at runtime.

Interoperability issue might be present at various levels of communication layers. Application level interoperability can be defined to occur between service calls, such that endpoint-A calls a service that exists in endpoint-B with the correct signature and parameters, and also data interoperability. In this research we address solely service call interoperability.

Figure 1 summarizes our solution framework. A system integrator first (Step-1) needs to model the interoperability scenarios in sequence diagrams. Each diagram captures expected behavior of an individual service of a particular endpoint in terms of CoAP interactions with other endpoints; thus, there must be as much sequence diagrams as the number of services provided by an endpoint in an IoT system, in order for fully covering all behaviors. The interactions are represented as asynchronous message exchanges in the sequence diagrams. In second step, a model-to-text transformation algorithm is exerted on each sequence diagram to transform event relations in it into EPL statements. EPL statements act as runtime monitors. EPL statement is an executable special purpose instruction written in Event Processing Language (EPL) of Esper CEP engine [15]. EPL statements are implemented as (see Sect. 4) Java classes that

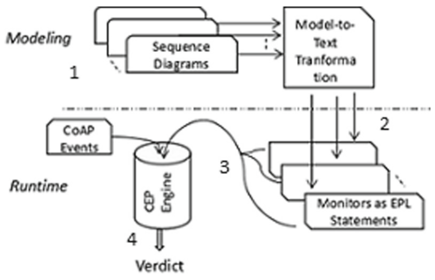


**Fig. 1.** MBT process for interoperability

can be run on any Java compatible platform. Those are registered (in Step-3) with an Esper engine running either on a stand-alone endpoint acting as an edge computing solution for interoperability testing, or it can be provided as a service over a cloud implementation. In step 4, the CoAP events that are captured from the running network by means of sniffing it passively are injected into the Esper engine for monitoring through complex-event processing. The *Verdict* can either be *SUCCESS* or *FAIL* depending on the result.

## 4 Implementation

The example implementation assumes a healthcare system based on research in [16]. They present a case study on interoperability testing for HL7 systems with a sample hardware reference implementation. The communication model is based on CoAP (Fig. 2).

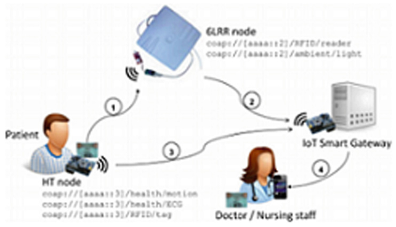

GLRR node
coap://[aaaa::2]/RFID/reader
coap://[aaaa::2]/ambient/light
1
2
3
4
Patient
IoT Smart Gateway
HT node
coap://[aaaa::3]/health/motion
coap://[aaaa::3]/health/ECG
coap://[aaaa::3]/RFID/tag
Doctor / Nursing staff

**Fig. 2.** Healthcare interoperability [16]

In [6] we showed that a CoAPsystem can be expressed in terms of *Send Events* in the system. Thus, we can represent a patient consent scenario with a sequence diagram as in Fig. 3. For ensuring privacy, a doctor must first request patient's consent for observing health data (e.g., ECG) ($m_1$). After the patient grants the consent ($m_2$), the doctor can ask to observe the patient data ($m_3$).

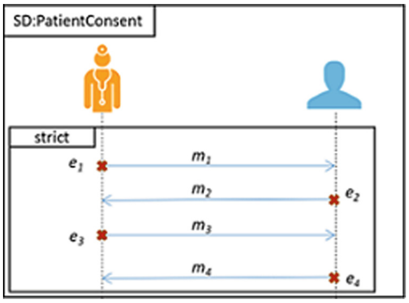


**Fig. 3.** Patient consent sequence diagram

After that, the sensor on patient can periodically sends the measured data ($m_4$). Each $e_i$ represents a *send event* for corresponding message $m_i$.

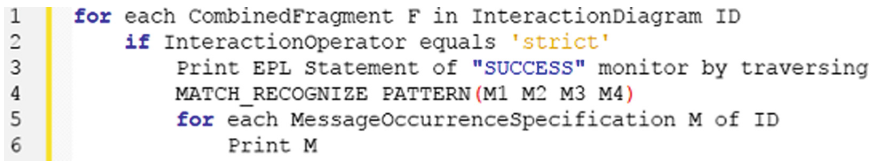
$$Follows(e_i, t_i, e_j, t_j) \\ \equiv \exists e_i, t_i, e_j, t_j Happens(e_i, t_i) \wedge Happens(e_j, t_j) \wedge (t_i < t_j) \tag{1}$$

The system in Fig. 3 can be represented in terms of events (Eq. 1) by using *Follows* relations as described in [6]. Equation 1 states that $e_i$ must be followed by $e_j$ if they appear sequentially on the sequence diagram (e.g., $e_2$ follows $e_1$). Thus, by observing if each sequential pair of ($e_i$, $e_j$) at runtime satisfies *Follows* relation we can conclude that interoperability patient consent requirement. In order to conclude with a *SUCCESS* verdict (Eq. 2), we must have $e_1 \prec e_2 \prec e_3 \prec e_4$, where $\prec$ denotes the precedence relation. For a *FAIL* result (Eq. 3), a $e_i \nprec e_j$ must hold for $(i, j) \in \{(1, 2), (2, 3), (3, 4)\}$, where $\nprec$ represents doesn't precede relation. Equation 3 states that the CEP engine must select all the complex events that occur as a result of $m_1$ is followed by either $m_3$ or $m_4$ message before an $m_2$ event occurs in order to indicate a failure case. The same rule can be extended for other messages as well.

$$select\ 'SUCCESS' from\ HealthEvent\ match\_recognize( \\ measures\ A.mId\ as\ a\_id, B.mId\ as\ b\_id, C.mId\ as\ c\_id, D.mId\ as\ d\_id \\ pattern\ (A\ B\ C\ D)\ define\ A\ as\ A.mId = m_1, B\ as\ B.mId = m_2, \\ C\ as\ C.mId = m_3, D' as D.mId = m_4); \tag{2}$$

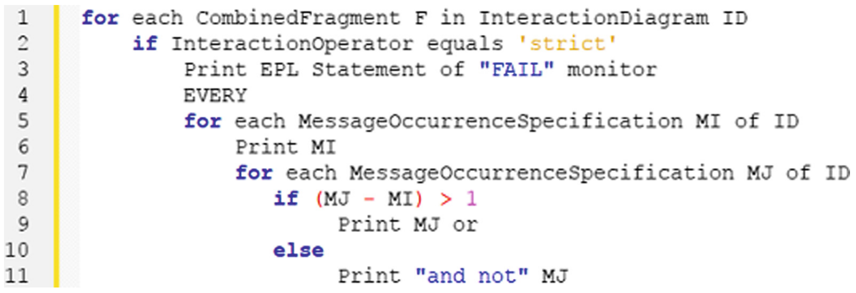
$$select\ 'FAIL', m1\ from\ pattern\ [every m1 = HealthEvent(mId = m1) -> \\ ((m3 = HealthEvent(mId = m3)\ or\ m4 = HealthEvent(mId = m4))\ and \\ not\ m2 = HealthEvent(mId = m2))]\,; \tag{3}$$

Figures 4 and 5 list algorithms for generating EPL statements for *Success* and *Fail* cases of interoperability testing in Acceleo. Acceleo runs over XMI definitions of sequence diagrams, and generates Java classes containing corresponding EPL statements (Eqs. 2 and 3). After executing M2T code in Papyrus [5], a Java class



```
for each CombinedFragment F in InteractionDiagram ID
    if InteractionOperator equals 'strict'
        Print EPL Statement of "SUCCESS" monitor by traversing
        MATCH_RECOGNIZE PATTERN(M1 M2 M3 M4)
        for each MessageOccurrenceSpecification M of ID
            Print M
```

**Fig. 4.** Algorithm for generating epl statement of success verdict



```
for each CombinedFragment F in InteractionDiagram ID
    if InteractionOperator equals 'strict'
        Print EPL Statement of "FAIL" monitor
        EVERY
        for each MessageOccurrenceSpecification MI of ID
            Print MI
            for each MessageOccurrenceSpecification MJ of ID
               if (MJ - MI) > 1
                    Print MJ or
               else
                    Print "and not" MJ
```

**Fig. 5.** Algorithm for generating EPL statement of fail verdict

containing a similar EPL statement as in Eq. 2 is generated. This EPL statement selects all the matching sequences of messages as described in Fig. 3.

The solution framework can be extended to other scenarios by following the procedure and implementation details described in Sects. 3 and 4. The event relations logic, how to sniff a network for CoAP packets through a CoAP sniffer, and how to run runtime monitors as EPL statements are explained in [6]. Note that, the solution framework would be applicable to both online and offline testing provided that raw CoAP packets are injected into the CoAP Sniffer. This solution enables for observing interoperability of IoT systems at runtime.

## 5 Conclusion

The paper presented a framework for facilitating interoperability testing of IoT systems. The framework promotes interoperability through model-based testing techniques. We utilized sequence diagrams in order to describe expected interactions between endpoints. Then, those are extracted from the diagram so as to compose a set of runtime monitors in terms of CEP EPL statements. We demonstrated the applicability of this approach with a case study on a healthcare system. Our future work will address a more comprehensive interoperability approach by involving structural and semantics testing; which will present a domain-specific metamodel for CoAP-based IoT systems, and the framework will be incorporated in a cloud service such that the solution can be used as a service. We believe that we can model the interactions between IoT systems with thorough event relations, which elaborates on the application layer protocol behavior.

## References

1. Fortino, G., Trunfio, P.: Internet of Things Based on Smart Objects, Technology, Middleware and Applications. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-00491-4
2. Spichkova, M., Schmidt, H., Peake, I.: From abstract modelling to remote cyber-physical integration/interoperability testing. CoRR Journal, abs/1403.1005 (2014)
3. IETF Constrained RESTful Environments (core) Working Group. https://datatracker.ietf.org/wg/core/about/
4. Shelby, Z., Hartke, K., Bormann, C.: The constrained application protocol (CoAP). IETF RFC-7252 (2014)
5. Papyrus Modeling Environment. https://eclipse.org/papyrus/
6. İnçki, K., Arı, İ., Sözer, H.: Runtime verification of IoT system using complex event processing. In: Proceedings of 14th IEEE International Conference on Networking, Sensing and Control, Italy. IEEE Press (2017)
7. Unified Modeling Language (UML) Version 2.5. http://www.omg.org/spec/UML/2.5/
8. Wu, Y., Chen, M.-H., Offutt, J.: UML-based integration testing for component-based software. In: Erdogmus, H., Weng, T. (eds.) ICCBSS 2003. LNCS, vol. 2580, pp. 251–260. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36465-X_24
9. Allen, P.: Component-based Development for Enterprise Systems: Applying the SELECT Perspective. Cambridge University Press, Cambridge, UK, New York (1998)
10. Bertolino, A., Polini, C.: The audition framework for testing web services interoperability. In: 31st EUROMICRO Conference on Software Engineering and Advanced Applications (2005)
11. Vega, D.E., Schieferdecker, I., Din, G.: Design of a test framework for automated interoperability testing of healthcare information systems. In: 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine (2010)
12. Canfora, G., Di Penta, M.: Testing services and service-centric systems: challenges and opportunities. IEEE IT Prof. **8**(2), 10–17 (2005)
13. Smythe, C.: Initial investigations into interoperability testing of web services from their specification using the unified modelling language. In: Proceedings of International Workshop on Web Services Modeling and Testing (WS-MaTe 2006) (2006)
14. Chen, N., Viho, C., Baire, A., Huang, X., Zha, J.: Ensuring interoperability for the Internet of Things: experience with CoAP protocol testing. Automatika **54**(4) (2013)
15. EsperTech Complex-Event Processing Tool. http://espertech.com/
16. Gebase, L., Snelick, R., Skall, M.: Conformance testing and interoperability: a case study in healthcare data exchange. In: Proceedings of the 2008 International Conference on Software Engineering Research and Practice, SERP 2008, Las Vegas (2008)
17. Acceleo Model to Text Language. https://www.eclipse.org/acceleo/