Towards Self-Healing in the Internet of Things by Log Analytics and Process Mining

Prasannjeet Singh

Department of Computer Science, Linnaeus University, Sweden. E-mail: ps222vt@student.lnu.se

Francesco Flammini

Department of Computer Science, Linnaeus University, and Mälardalen university, Sweden.

E-mail: francesco.flammini@lnu.se

Mauro Caporuscio

Department of Computer Science, Linnaeus University, Sweden. E-mail: mauro.caporuscio@lnu.se

Mehdi Saman Azari

Department of Computer Science, Linnaeus University, Sweden. E-mail: mehdi.samanazari@lnu.se

Johan Thornadtsson

Sigma Technology Group, Sweden. E-mail: johan.thornadtsson@sigmatechnology.se

The Internet of Things (IoT) will be used in increasingly complex and critical applications where heterogeneous devices will work together in connected systems. In this paper we address methods for log-analytics and process mining in order to support automatic problem detection and diagnosis in IoT. We introduce the idea of generating consistent event logs over various IoT devices in a particular format, and later a roadmap for it to be used in process mining. The paper also provides information about various statistics on process mining and its future prospects. Those methods are essential to provide a foundation for the future generation IoT systems that will be capable of self-healing.

Keywords: self-healing, self-repair, diagnostics, prognostics, event correlation, threat detection, Internet of Things.

1. Introduction

The Internet of Things (IoT) is not a new concept; however, it is used more than ever nowadays even on a personal basis. There are hardly any personal electronic devices which are used independently without being connected to a system or the Internet. This phenomenon is not just limited to personal systems and is also prevalent in other industries like heavy machinery, automobiles, etc., where every individual system is connected to one or more systems, which monitors or controls them according to the requirement. The IoT can be described as a system (of systems) that includes the collection of tangible components that are interconnected, either wireless or wired, and the whole system is connected to Internet. The components may include a range of sensors which can use various types of local area connections and wide area connectivity, and can record human or other activities which can then be relayed to another component, such as the processing system. Such information can be used to operate the system effectively, to gather data, to troubleshoot the system, or to perform any other relevant tasks

(20). The deployment diagram of a typical IoT system is shown in Figure 1. Multiple connectivity between entities can be observed. Some devices are connected wireless whereas some have a wired connections. Moreover, the sub-systems belong to different manufacturers. However, all of them work together to form an orchestration and perform functions of different criticality.

Similar to the Figure 1, most of the components in an IoT system are manufactured by different vendors. Although many failures *within* an individual component may be taken care of by their respective manufacturers, it is likely that any other failure *between* different components in an IoT system, such as a simple connectivity problem, will not be supported by the manufacturers, and it is generally not covered under any warranty.

A popular report from Forrester (17) predicts that there will be a huge growth in the IoT industry in the next years where enterprises will increase their efforts to introduce voice-based services to consumers and new European guidelines will allow commercializing the IoT data. In 2010, the number of objects connected to the Internet surpassed the earth's human population (1). Ad-

ditionally, Symantec^a predicts that there will be up to 21 billion connected devices by 2020, and not just the consumers, but cities and companies will also start adopting *smart* technologies in their operations. A report from Gartner^b says that the worldwide spending in IoT Security was 1.1 Billion USD and it is projected to skyrocket at more than 3.1 Billion USD in 2021, which is a direct result of the increase in vulnerabilities in the systems. The above discussion leads us to believe that diagnosing and troubleshooting failures in an IoT system will be a huge problem in the near future which may not be covered by any manufacturer, unless all the entities within the system belong to the same company.

Self-healing refers to the automatic recovery process by detection and diagnosis of faults and their subsequent correction in a temporary or a permanent manner. Self-healing systems are of particular interest as because self-healing directly impact improvements in dependability. Self-adaptive systems are ones that monitor their execution environment and react to changes by modifying their behavior in order to maintain an appropriate quality of service. Therefore, there is a substantial intersection between self-healing and self-adaptiveness. self-healing systems can be considered as a specific kind of self-adaptive systems^c.

There are many ways to detect if an error has occurred in a system (23). One of them is System-Level Monitoring that is used by several commercial enterprise applications. Another notable method is to detect errors/failures at the application layer. However, one of the most common way to examine a system for detecting any error, fault or failure is by analyzing the event logs generated by the system. Process mining is also discussed that generates its models via event log analysis. Therefore, in this paper, we focus on the idea of analyzing event logs for Self-Healing of IoT by automatically diagnosing and troubleshooting threats (faults, error, failures) in IoT devices. The choice of the approaches addressed in this paper is supported by a recent survey of the state-of-the-art addressing open issues, challenges and opportunities in this research field (4).

The organization of this paper is as follows: IoT error prevention, detection and diagnosis will be briefly surveyed in Section II. Specific meth-

ods for log analysis and process mining will be addressed in Section III. Finally, Section IV will draw conclusions and hints about future developments.

Fig. 1. A Typical Smart Home Deployment Diagram.

2. Prevention, Detection and Diagnosis in IoT

For the sake of consistency in this paper, we will follow the taxonomy for dependable and secure computing as proposed by Avizienis et al.(2). Since the expected number of bugs in a system is proportional to its lines of code (14), it can be safely assumed that it is impossible to exhaustively list down all possible faults in a system. Therefore, introduction of faults in a system is inevitable, and more emphasis is given in developing a *fault-tolerant* system, rather than building a *fault-less* system. A system could be termed as *fault-tolerant*, if it is able to prevent the *fault* from turning into a *failure*. (24) proposes a fault tolerant mechanism which is implemented on a WuKong^d middleware. (9) proposes a fault-tolerant architecture for healthcare IoT systems consisting Wireless Sensor Networks (WSN).

Mishra et al. in their research(16) propose a method for fault-tolerant routing in IoT. A fault-tolerant routing protocol is suggested for IoT systems that is based on mixed cross-layer and learning automata (LA). Kouwe, in their thesis(31) propose *fault-injection*, where artificial faults are injected into a system in order to learn the behavior of a system with activated faults, and eventually prevent system failure.

Cyber Physical Systems such as SCADA (Supervisory Control And Data Acquisition) is an

^aSymantec, 5 predictions on the future of the internet of things. <https://nr.tn/2O9V1xE>

^bW. Bamiduro, R. van der Meulen, Worldwide iot security spending will reach \$1.5 billion in 2018, Gartner (2018). <https://gtr.it/2Kd0hg1>

^cArtur Andrzejak, Kurt Geihs, Onn Shehory, John Wilkes - Self-

Healing and Self-Adaptive Systems Dagstuhl Seminar 09201., <https://www.dagstuhl.de/en/program/calendar/semhp/?seminr=1351> [2016]. <https://news1abntu.github.io/wukong4iox/>

^dK.-J. Lin, D. Shih, Y.-C. Wang, J. Hsu, N. Reijers, G. Chou, B. Tsai, Wukong - intelligent middleware for internet-of-things

appropriate example for the current state-of-the-art in fault tolerant Internet of Things system(22). These systems are competent in offering flexibility, stability and fault tolerance. These systems exploiting cloud computing services, integrated with Internet of Things can be judged as Smart Industrial Systems which are predominantly employed in smart grids, smart transportation, eHealthcare and smart medical systems.

Intentional attacks are common threats in IoT. Thus, it is important to employ an IoT intrusion prevention mechanism in to prevent those threats. Various methods of intrusion detection are discussed and a taxonomy for the same by Zarpelão et al.(34). Bertino and Islam(3) propose various guidelines that can prevent an IoT system from being compromised. Kasinathan et al.(12) propose a DoS Detection Architecture for 6LoWPAN IoT systems using *Suricata*, an open-source intrusion detection system that detects and eliminates the attack using appropriate countermeasures before the network operations are disrupted. Another efficient way of intrusion prevent is the installation of Honeypot in a system. Likewise, Honeynets are an aggregation of Honeypots that are intended to imitate usual servers and network services(19). In general, honeypots are essentially a technique of deception, where the defender purposely hoodwinks the attacker into acting in one's favor(6). While Yu et al.(33) rejects the idea of Honeypots in an IoT setup due to non-scalability and dependency issues, La et al.(13) considers the possibility and uses game theory to analyze the situation where both attacker and defender try to deceive each other. While virtual patching is a type of firewall often mentioned as a Web Application Firewall (WAF), an IoT system also needs a full-fledged firewall as most of the embedded systems that are part of the IoT system have little to no security. A recent improvement on the traditional firewalls for IoT is *Smart Firewall*, which, unlike traditional software-based firewall, is a hardware-based device^e. Gupta et al.(10) have proposed an implementation of firewall for Internet of Things using Raspberry Pi as a gateway.

3. Log Analysis and Process Mining

Traditional logging techniques in major applications are not structured and many of the errors logged by them don't lead to faults, while these errors might be useful in some cases, most of the times they only decrease the readability of error logs by humans. Furthermore, there are some instances where errors, which were directly

responsible for the failure of a system, were failed to be captured by these logging techniques. Considering that we are only focused on errors that lead to a failure, these techniques lead to a lot of false positives and false negatives. Here we focus only on the effective errors, or the errors which leads to the activation of faults, which in turn leads to a failure.

Most of the logging patterns try to detect errors and log it by placing a line of code at the end of a block of instructions, these techniques may not be able to detect errors such as infinite loops, etc. Therefore, instead of working on the resultant error logs, Cinque et. al. (5) aims to develop a new logging mechanism which takes care of these issues, and others such as false positives and false negatives by monitoring changes in the control flow of the program, this is done by placing the logging instructions strategically in the code. Moreover, it aims to detect only those errors which cause failure.

3.1. The Rule-Based Logging-Mechanism

Further, a set of error-modes are established, based on the widely accepted taxonomy in the dependability area(2). Such as Service Error (SER: Prevents an invoked service from reaching the exit point.), etc. Different types of code injections, called login rules (LR) in the form of events are then done in the source code of system which helps in the logging, as an example, LR-1 (Service Start), LR-2 (Service End), HTB (Heartbeat), etc.

In Rule-Based logging, these Login Rules don't write the log-file directly but are processed dynamically by a framework called as LogBus. The above logging mechanism can be extended to work within an IoT system by introducing a new entity within the system which can be in the form of a framework. We name it as the *LogMonitor* (Figure 2). Contrary to traditional logging mechanisms where codes written within the system update the error logs, the LogMonitor will monitor all the events spread over various IoT systems in an environment logged by the aforementioned logging rules, and eventually update the error log. This way a consistent error log can be generated for the entire system. However, this mechanism assumes that we have access to modify the internal codes of each entity within the IoT system. Other tedious methods, such as parsing of event logs(11; 7) from each entity may have to be done, in case the internal access is unavailable.

3.2. Defining Event Logs

In the previous section, an approach was discussed to make an event log consistent in an IoT system. However, there still is a need to define the event logs and categorize them in such a way that appropriate tools can be developed that works

^eK. Colburn, Smart firewalls protect the internet of things — which are the best? — wtop (2017). <https://wtop.com/tech/2017/10/smart-firewalls-protect-internet-things-best/>

Fig. 2. Conceptual structure of a Log-Monitor.

Table 1. A Sample Event Log

Case ID	Event ID	Time	Activity
1	85477	15-01-2019:14:05	connection request received
1	85478	15-01-2019:14:06	SYN
1	85479	15-01-2019:14:06	SYN-ACK
1	85480	15-01-2019:14:08	pin verification
1	85481	15-01-2019:14:08	verification successful
1	85482	15-01-2019:14:08	ACK
1	85483	15-01-2019:14:09	connection successful
2	92199	11-02-2019:09:11	connection request received
2	92200	11-02-2019:09:12	SYN
2	92200	11-02-2019:09:12	SYN-ACK
2	92201	11-02-2019:09:13	pin verification
2	92201	11-02-2019:09:15	verification failed
2	92202	11-02-2019:09:16	pin verification
2	92202	11-02-2019:09:17	verification failed
2	92200	11-02-2019:09:18	pin verification
2	92201	11-02-2019:09:18	verification successful
2	92201	11-02-2019:09:20	ACK
2	92202	11-02-2019:09:21	connection successful

on each category efficiently. Some event logs contain time-stamps; some contain user-readable texts while others do not. Some contain just a few sets of information in the events, while some may contain hundreds of information in a single line of event log. Although the format of the event logs depend on the manufacturer, in most of the standardized cases, an event log has an *Event ID*, *Time Stamp*, one or more primary attributes such as *Activity*, and other secondary attributes such as *Source*, *Server Name*, etc.^f Thus, for a broad understanding and intuitiveness, let us assume a general event log sample, that is readily comprehensible, as shown in Table.1. The given event log can represent most types of event log generated by a system, as it includes some identification (*Case ID* and *Event ID*), a *Time Stamp* and in this case, one primary attribute (*Activity*) and zero secondary attribute. The number of attributes for an event log has no restriction, however, these event logs are bound by some underlying assumptions(28). One important assumption among the aforesaid is that each event refers to some activity. For example, in Table.1 , every event consists of an activity, such as pin verification, connection successful, etc. A system such as the *LogMonitor*, as discussed in the Section 3.1 above, could be programmed in such a way that it generates the event logs in the above format.

As can be observed, the event logs in Table.1 are initially divided into Cases. Each case is further divided into Event IDs. It is worth noting that events listed under a case should only relate to precisely one case. Moreover, all the events in

a case should be chronologically ordered, and at least one attribute is mandatory for an event log to exist.

3.3. Process Mining

An IoT system can be expected to generate anywhere from a thousand to hundreds of thousands of events logs instances in an hour, and as a result it is imperative to extract meaningful information from them that can help us troubleshoot any errors. One of the ways through which this can be achieved is Process Mining. It is an excellent tool to capture meaningful information from event logs. It is mainly used to offer new medium to discover, monitor and improve processes in a quality of application domains. However, it can also be used to analyze and rectify errors in a system. It is a comparatively new research discipline that tries to extract knowledge from event logs of real processes (i.e. not assumed processes) which are readily available in today's information systems. A report from Gartner estimated the market size of Process Mining to approach \$160 Million in 2018 (8). This is expected to grow to \$1,421.7 million by 2023 with 50.3% as the Compound Annual Growth Rate^g. The same report also predicts the growth of Process Mining usage in the IT industry (Figure 3).

A process model generated by process mining using the event logs can be very effective in handling large amounts of event logs. Furthermore,

^fIBM: Event Log Attributes. <https://ibm.co/2yGbDE2>

^gPress-Release, Global process analytics market, 2023 by process mining type, deployment type, organization size, application, Tech. rep., MarketWatch (2018). <https://on.mktw.net/2YafVIE>

Fig. 3. Projected Process Mining Use Case in 2020

many commercial and academic systems have implemented process mining algorithms. Wil van der Aalst(32) claims that an active group of researchers are working on process mining and it has become one of the “hot topics” in Business Process Management (BPM) research. Additionally, process mining functionality is added by more and more software vendors to their tools. Notable examples are: Discovery Analyst (Stereologic), ARIS Process Performance Manager (Software AG), and Comprehend (Open Connect). Aforementioned reasons led us to believe that Process Mining can be an effective tool for solving the problem of Self-Healing in IoT systems. Besides this, as of July, 2015, there were 18 PhD students being funded by Philips who were working in analyzing the Philips X-Ray machines and electronic shaving devices among other items to see how these machines are used in the field, when do they fail and why do they fail, etc.^h. Moreover, most of the X-Ray machines manufactured by Philips use Process Mining for fault diagnosis (25; 32).

Since process mining primarily deals with event logs, it is assumed that the event logs used are in accordance with the guidelines mentioned in Section 3.2. The event-log is the starting point and can be used to conduct three types of process mining. We will briefly describe these three parts below (Figure 4).

- **Process Discovery:** The most important part in Process Mining, it creates a model (called as Process Model) using event logs of real processes, but not using any theoretical or deduced information.
- **Conformance Checking:** Here, a process model which was created through Process Discovery is compared with an unseen event log of the same process. This is done to check if in reality, an event log conforms to the model and vice-

versa.

- **Enhancement:** As the word says, here the objective is to improve the existing process model using additional information recorded in the event log.

While Process Discovery still remains the most popular type of process mining, it's usage is constantly declining, where as the usage of the other two, i.e. Conformance Checking and Enhancement is increasing and is expected to increase in the future as well(8) (Figure 5).

Fig. 4. Different types of Process Mining in terms of Input and Output.

For creating a process model in Process Mining, we start from a bunch of behavior (event logs, in our case) and automatically construct the model based on the logs. This is because commonly, there are no appropriate existing models, or the models are flawed or incomplete. Reference(21) describes process model which is *roughly an anticipation of what the process will look like*.

Fig. 5. Projected adoption of basic process mining types.

^hW.

van der Aalst, Process mining data science in action (2015). <https://www.youtube.com/watch?v=kLeLaNzw9hI&t=704s>

A process mining model can be represented using various notations, and the fact that there

are many notations available demonstrates the significance of process modeling. A simple event log as shown in the Table.1 may be represented using an intuitive model. However, in case of an IoT system, or any electronic system whatsoever, the event logs are far more complex than the one illustrated. As a result, a legitimate, well-defined notation should be used to represent a process model. Moreover, process models for complicated systems are prone to many errors(30), some of which are outlined underneath.

- *Oversimplification of the model:* Most models have a propensity to focus only on the desirable behavior, and in a way overlook the events that are less likely to happen. There are chances that these “overlooked” events cause most of the errors and thus, the model may not be helpful.
- *Incompetence in satisfactorily capturing human behavior:* Simple and mundane process involving human engagement are prone to modifications, as human behavior is unpredictable. These changes, although minor, should nevertheless be incorporated in a model, as these nonconformities by and large result in an eventual fault.
- *Restricted or redundant abstraction level:* It is important that an appropriate abstraction level is chosen based on the objective and the input data. There is a plausibility that the model is too abstract to answer a detailed question, or conversely, a model is too detailed, and thus redundant, to answer a simple question.

A manually composed process model are susceptible to these (and similar) errors. Moreover, a poorly designed model may engender wrong conclusions. To eradicate these problems, process mining takes the help of event logs to create a process model. An event log notifies the exact steps that the system underwent to complete any task. Additionally, using a surfeit of event logs for modelling will result in the inclusion of all the events that might be less likely to happen, as discussed above. Moreover, a process model is capable of providing different view in different abstraction levels for the same system.

Algorithms such as the α -algorithm (29) can automatically generate process models based on event logs, as will be discussed in future sections. Notations such as EPC’s, BPMN, UML activity diagrams, Transition System, Workflow Nets, YAWL etc. can be used to describe process model. However, Petri nets are capable of describing concurrent processes without much effort like the transition systems. This is one of the oldest process modeling language. Due to its broad usage, it has also been extensively investigated; and as a result, there exists many tools to analyze them^h (25; 27; 32). A Petri net, as suitably described by authors in (15) is “a directed bipartite graph, in

Fig. 6. A Petri-net process model generated using α -algorithm for the sample event log displayed in Table.1

which the nodes represent transitions (i.e. events that may occur, signified by bars) and places (i.e. conditions, signified by circles).” Petri nets have a static network structure. *Tokens* flow through the network governed by certain firing rules. The distribution of tokens over places (also referred as *marking*) determines the state of the Petri net. (18) provides a broad understanding of Petri nets. Furthermore, a *labeled Petri net* is an extension of the basic Petri nets, including a set of activity labels. Figure 6 is a Petri-net model for the sample event log shown in Table 1.

There are various existing algorithms that convert event logs to a process model as above. One of the most prominent and rather naïve algorithm is the α -algorithm (29). Additionally, Figure 7ⁱ is an example of a seemingly more complicated process model. It was created using ProM Tools. The inset in Figure 7 shows a zoomed-in version of the shaded region of the process model in Figure 7. Similarly, process models also contain detailed information, but relevant information may be available only in a particular section.

4. Conclusion

As we see the importance and growing presence of IoT and Connected Cyber-Physical systems around the world, we stressed the fact that there is need for methodologies, tools, technologies, procedures and frameworks supporting automatic error detection and – possibly – repair. In this paper we have investigated the usage of process mining based on log file analysis as a useful tool to

ⁱThis sample process model was created using the Event-Logs of BPI Challenge 2018 (<https://data.4tu.nl/repository/uuid:3301445f-95e8-4ff0-98a4-901f1f204972>). The log-file contains 2,514,266 instances of events that resulted in the process model as shown in the figure.

Fig. 7. A complex process-model using Petri-nets, and a zoomed-in version of the shaded region shown in the inset.

support self-healing. We have shown the potential of those techniques and also highlighted some open issues to be addressed in future research. For instance, bisecting process models requires various protocols to be followed (26), which are not covered in this paper. However, once a robust process model is built, the following situations can be detected and managed through model analysis:

- (1) Something did not happen that should have.
- (2) Something happened that should not happen.
- (3) If the IoT system is being used as it was intended to be.
- (4) Which is the most frequent path followed by the user, etc.

There are various tools available in the market to create a process model, e.g. ProM, Disco, etc. However, in order to create a process model for a new system, a plugin may be needed so that the event logs can easily be comprehended by the tool. Note that apart from self-healing, process mining may also be capable of solving the problem of error predictability, where an end-user may be warned beforehand in case external events and/or user actions have a high probability of causing an error. Such analysis belongs to the categories known as prognostics, early warning and situation assessment. In conclusion, it can be said that generating consistent event logs throughout all the IoT devices in one environment by using a system such as the *LogMonitor* can be very effective for analyzing inter-device faults. Furthermore, generating process models using process mining while adhering to the *Event Log Rules* can be a great step forward in the field of self-healing in IoT systems.

References

1. A. Al-fuqaha, S. Member, M. Guizani, M. Mohammadi, and S. Member. Internet of things : A survey on enabling. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.
2. A. Avižienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.
3. E. Bertino and N. Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, Feb. 2017.
4. M. Caporuscio, F. Flammini, N. Khakpour, and P. Singh. Smart-Troubleshooting Connected Devices: Concept, Challenges and Opportunities, *Future Generation Computer Systems*, 2019.
5. M. Cinque, D. Cotroneo, and A. Pecchia. Event logs for the analysis of software failures: A rule-based approach. *IEEE Transactions on Software Engineering*, 39(6):806–821, 2013.
6. D. C. D. C. Daniel and K. L. K. L. Herbig. Strategic military deception: Pergamon policy studies on security affairs. Pergamon Press, 1982.
7. M. Du and F. Li. Spell: Streaming parsing of system event logs. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 859–864, 2017.
8. Gartner. Market guide for process mining. Technical report, Gartner, 2019.
9. T. N. Gia, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fault tolerant and scalable iot-based architecture for health monitoring. In *2015 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, Apr. 2015.
10. N. Gupta, V. Naik, and S. Sengupta. A firewall for internet of things. In *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, pages 411–412. IEEE, Jan. 2017.
11. P. He, J. Zhu, S. He, J. Li, and M. R. Lyu. Towards automated log parsing for large-scale log data analysis. *IEEE Transactions on Dependable and Secure Computing*, 15(6):931–944, 2017.
12. P. Kasinathan, C. Pastrone, M. A. Spirito,

- and M. Vinkovits. Denial-of-service detection in 6lowpan based internet of things. In 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 600–607. IEEE, Oct. 2013.
13. Q. D. La, T. Q. S. Quek, J. Lee, S. Jin, and H. Zhu. Deceptive attack and defense game in honeypot-enabled networks for the internet of things. *IEEE Internet of Things Journal*, 3(6):1025–1035, Dec. 2016.
 14. M. Lipow. Number of faults per line of code. *IEEE Transactions on Software Engineering*, SE-8(4):437–439, 1982.
 15. G. Manoj, J. Samson Immanuel, P. S. Divya, and A. P. Haran. Modelling of system configuration and reconfiguration for iots. In T.-h. Kim, D.-s. Ko, T. Vasilakos, A. Stoica, and J. Abawajy, editors, *Computer Applications for Communication, Networking, and Digital Contents*, pages 285–292, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
 16. S. Misra, A. Gupta, P. V. Krishna, H. Agarwal, and M. S. Obaidat. An adaptive learning approach for fault-tolerant routing in internet of things. In 2012 IEEE Wireless Communications and Networking Conference (WCNC), pages 815–819. IEEE, Apr. 2012.
 17. M. Pelino, J. S. Hammond, C. Dai, P. Miller, J. Belissent, J. A. Ask, N. Fenwick, F. E. Gillett, T. Husson, M. Maxim, C. Voce, C. Garberg, and D. Lynch. Predictions 2018: Iot moves from experimentation to business scale. Technical report, Forrester, 2017.
 18. C. Petri and W. Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008.
 19. N. Provos and T. Holz. Virtual honeypots : from botnet tracking to intrusion detection. Addison-Wesley, 2008.
 20. L. Research. An introduction to the internet of things (iot). Technical report, Lopez Research, 2013.
 21. C. Rolland. A comprehensive view of process engineering. *Lecture Notes in Computer Science*, 1413(c):1–24, 1998.
 22. A. Sajid, H. Abbas, and K. Saleem. Cloud-assisted iot-based scada systems security: A review of the state of the art and future challenges. *IEEE Access*, 4:1375–1384, 2016.
 23. L. M. Silva. Comparing error detection techniques for web applications: An experimental study. 2008 Seventh IEEE International Symposium on Network Computing and Applications, pages 144–151, 2008.
 24. P. H. Su, C. S. Shih, J. Y. J. Hsu, K. J. Lin, and Y. C. Wang. Decentralized fault tolerance mechanism for intelligent iot/m2m middleware. 2014 IEEE World Forum on Internet of Things, WF-IoT 2014, pages 45–50, 2014.
 25. W. Van Der Aalst. Process mining. *Communications of the ACM*, 55(8):76, 2012.
 26. W. van der Aalst. Process mining data science in action, 2015.
 27. W. van der Aalst and C. Stahl. Modeling business processes: a petri net-oriented approach. The MIT Press, 2011.
 28. W. M. P. van der Aalst. Getting the Data, chapter 4, pages 95–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
 29. W. M. P. van der Aalst. Process Discovery: An Introduction, chapter 5, pages 125–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
 30. W. M. P. van der Aalst. Process Modeling and Analysis, chapter 2, pages 29–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
 31. E. van der Kouwe. Improving software fault injection. PhD thesis, Vrije Universiteit Amsterdam, 2016.
 32. G. Vossen. The process mining manifesto - an interview with wil van der aalst. *Information Systems*, 37(3):288–290, 2012.
 33. T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks - HotNets-XIV, pages 1–7, New York, New York, USA, 2015. ACM Press.
 34. B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37, Apr. 2017.