Proc. of the 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE)
12-13 June 2020, Istanbul, Turkey

Available and Fault Tolerant IoT System: Applying Quality Engineering Method

Reginaldo Arakaki
Computer Engineering and Digital Systems Department
Polytechnic School – University of São Paulo
São paulo, Brazil
reginaldo.arakaki@poli.usp.br

Victor Takashi Hayashi
Computer Engineering and Digital Systems Department
Polytechnic School – University of São Paulo
São paulo, Brazil
victor.hayashi@usp.br

Wilson Vicente Ruggiero
Computer Engineering and Digital Systems Department
Polytechnic School – University of São Paulo
São paulo, Brazil
wilson@larc.usp.br

***Abstract*—IoT systems include the creation of digital platforms that connect in real-time events, many devices with different levels of robustness. That is, some components are simple, and others sophisticated, connected by a well-designed project using methods or using "try-correct" ad hoc procedures. This paper presents how faulty an Internet of Things (IoT) system could be if implemented without engineering quality and architecture fundamentals. Furthermore, describe by concepts and practical examples, how engineering design should apply architectural techniques driven by quality attributes (referenced by ISO/IEC 25010 standard). A Markov Chain Model used to improve availability and maintenance of a smart home application is presented to illustrate the importance of techniques to balance quality attributes of IoT systems.**

Keywords—IoT, Internet of Things, Education, Software Engineering, Quality Attributes, IoT Architecture, Fault-Tolerant Systems, Architectural Tactics

I. Internet of Things Evolution

One of most challenging evolution on emerging digital platforms is availability level of digital systems that brings useful functions to people, in transportation, work, food, entertainment, health, education, finance and public services.

Existing services and products accessed through computers are not sufficient. It is important that digital system can be close to customers wherever they are. To do this, some wearable devices are useful. Smartphones can be classified as examples of this kind of connected devices. They can measure and stay close to people during all the time, including sleep periods.

Smartphones is one of the most touched devices by people during the day, much more than desktop or notebook computers. People rely on smartphones for help in traffic using maps, transportation services requests, foods, entertainment, education and other conveniences. At end of the day, these devices provide actual real time events [1], fundamental aspects associated to context, which make it possible to get customer needs precisely. As a consequence, we do have much more information to recommend services and products with personalized actions, sent at the right moment and at the right place. Obviously under authorized and permitted access of data, as privacy is critical for user acceptance. Usability, accessibility, proximity and portability of smartphones devices are aspects that contribute to increasing the use for almost activities of customers. It was published in " Mobile Vs. Desktop Usage" by Broadband Search website, in 2019, 51,3% of web access were made by mobile devices [2, 3, 4].

Internet of Things (IoT) system brings huge impact on the architecture of solutions because of different interaction nature: instead of exclusively human interaction with a computer, machines and things digitally connected can interact with computers too, and these interactions could be enhanced by algorithms and machine learning methods.

An IoT system is designed and implemented with 2 types of components that are sharply in opposition concepts: at the one side, IoT sensors and actuator are weak devices, error prone; on the other hand, these weak components are connected to extremely sophisticated and powerful digital mechanisms as processor, memories, communication networks, network security control devices and cloud computing.

In this context, there are a lot of opportunities to improve digital solutions for the IoT systems. The software engineering discipline can bring some tools and concepts to deal with this kind of challenges that will be explored in this paper [4].

According to Gartner and other publications [5], in 2020 about 25 billion devices will be digitally connected in IoT systems. Some questions arise in this context: how do we get trusted system, considering strong and weak components connected? Is it possible to build fault tolerant system? Is it applicable in health areas? Or in public transportation areas? Or civil aviation system? Can we use such solutions structures (IoT) to benefit human beings in a reliable manner?

Obviously, one engineering challenge is: how do we build a fault-tolerant digital IoT system, considering strong components (like cloud computing resources) connected to weak components (like sensors and actuators)? Application for these quality systems can be viewed in a simple manner by the following classification: connected body for health and well-being supporting, connected car for digital autonomous or convenience services, connected home for digital, autonomous and convenience services and products, and connected city for digital secure public service for the people.

II. Objectives of the Paper

This paper brings architectural references to build IoT systems that present quality attributes to get resilient, flexible, trustful and fault-tolerant systems. Methods and techniques are applied to integrate IoT sensors and actuators with powerful cloud computing in a balanced way. In addition, a

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

case study of how to improve availability and maintenance aspects of a smart home application, using a Markov Chain model.

III. ISO 25010 – Quality Reference for an IoT System

The ISO 25010 standard publishing states quality attributes of a digital system [6, 7]. It classifies attributes to eight groups: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. Some of these classification groups are related to external aspects. Other with internal aspects. And, these groups contain thirty-one (31) attributes that can drive quality to a system.

It appears easy to build system controlling their quality attributes, but it isn´t. In an implementation of a IoT system, architectural decisions compound quality tradeoffs: every decision that focus on security can destroy usability characteristics; the same between volume and response time; and so on. As mentioned earlier, powerful cloud computing and weak IoT sensors are examples of architectural tradeoff that bring engineering challenges.

IV. Featuring an IoT System using ISO 25010 Standard

It is unclear how to use a quality standard to design a system if we consider a modern digital system as an IoT application. For example, a smart home where the entire house, their places, and their things are digitally connected to a robust cloud computing processing, using sensors and actuators. Some sensors would run to monitor events as gas flow, people movement, people or pet presence detection, and temperature. Some actuators would control light automation, garage gate motors, gas cutter device, air cooler equipment, voice and text automata, finance and e-commerce automata, among others.

In Fig. 1, there is a logic engineering diagram of an IoT system, with elements, as mentioned.

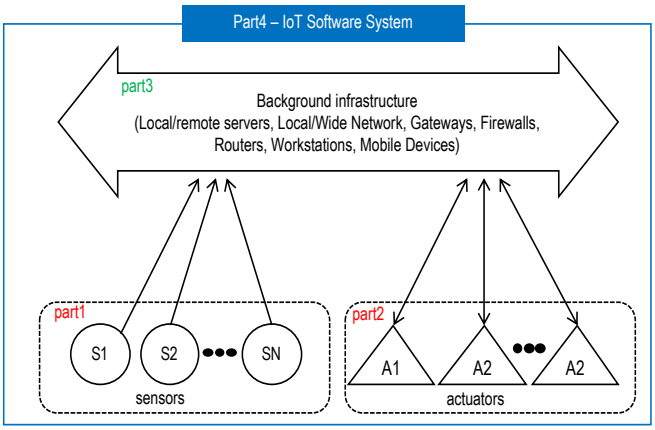


Fig. 1: Logic diagram of an IoT system.

A. IoT system features and ISO25010

According to the logic diagram represented in Fig. 1, the IoT system has four parts: sensors, actuators, background infrastructure, and the total. The architecture of the integration of that system is a composition of powerful components with weak components. How to evaluate the final quality of this system? Each part of the IoT system has a set of quality attributes, according to ISO 25010, a foundation technical reference to software products. Detailing and understanding each group of quality attributes helps us to identify gaps where engineering efforts and methods must be carefully applied to get reliable results.

The reader could ask: is it worth to know the weakness and strengths of a system? An answer could be: for each vulnerability, the engineering design of a final product has to be complete to get planned quality attributes.

B. Quality attributes of sensors

Sensors (part1), as depicted in the diagram of Fig. 1, have a vital function to collect data from the real world: temperature, gas level, blood sugar level and blood pressure of a person, acceleration and rotation rate of a movement, and GPS real-time positioning. Functional suitability is limited - that is, they usually have limitations on processing capacity and energy autonomy, related to performance efficiency.

C. Quality attributes of actuators

In Fig. 1, part2, the actuator has an essential function of touching real-world things. For example: activate a motor, accelerate, shift gears of a car; Open and close a garage gate; Automatic application of medicine or turn on/off lights; Stock control of a refrigerator and so on.

D. Quality attributes of cloud back office

Part3 of Fig. 1 compounds a digital infrastructure where communication and security components to deliver high processing power. Real-time events, intelligent algorithms, and massive data storage are available at an attractive level of digital processing.

E. Quality attributes of IoT system

Joining parts 1, 2, and 3 to compound the IoT system, there is a significant engineering challenge: how to get a good design if some parts are excellent and some parts are faulty? The partial answer is related to understanding and considering these specifications as real facts. A complete the answer is stating that a good engineering project must design, implement and sustain a system according to established quality requirements to deliver precise, useful, secure functions and services in a satisfying way to all project stakeholders.

That is, for each requirement, there is an architectural decision. For instance, a precision condition for temperature data collecting using redundancies of sensors and connectors. By using quality engineering methods one can deliver controlled results, in opposite to ad hoc procedures, remembering that a system that was designed and implemented with little or no quality care (usually with Ad Hoc procedures) is casual and amateur.

V. IoT Systems and Quality Attributes

When an IoT system reaches customers inside their cars, home, city, and body, special requirements arise from usability, including usefulness, precision, security, integrity, traceability, and reliability, considering much close interaction of a user with digital systems: all day, every moment, every place. In addition, it is essential to consider an usual profile is someone that does not know a lot about digital technology. It means that every event collected by sensors reflects real-world movement, real-world life workflows, and

the IoT system has to process and identify situations and context precisely to help its users on precise moments based on events.

TABLE 1: CRITERIA TO CLASSIFY QUALITY ATTRIBUTES.

Classification	Score/Color	Rationale
POOR	0.0 ~ 2.0	Nonexistent.
REGULAR	2.0 ~ 4.0	Vendor specific
GOOD	4.0 ~ 6.0	Configurable in the infrastructure
VERY GOOD	6.0 ~ 8.0	Configurable in functionality
EXCELLENT	8.0 ~ 10.0	Configurable in nonfunctional attributes

In Table 1, there is some criteria of scoring to develop an architectural examination, to map weak and strong points, by numbers and colors indicating four classification (level).

Poor Regular Good Very Good Excellent

	Sensors	Actuators	Cloud Infrastructure Background	IoT Software System
Functional Utility | 1. Functional completeness	2	4	0	8
Functional Utility | 2. Functional accuracy	2	4	0	8
Functional Utility | 3. Functional adequacy	2	4	0	8
Performance Efficiency | 4. Behavior in time	0	4	8	8
Performance Efficiency | 5. Resource use	0	5	9	8
Performance Efficiency | 6. Capacity (planning and management)	0	4	10	8
Compatibility | 7. Co-existence/ coexistence	1	2	9	6
Compatibility | 8. Interoperability	2	2	9	6
Usability| 9. Perception of business adequacy	0	6	2	10
Usability | 10. Ease of learning	1	6	2	8
Usability | 11. Ease of operation	1	6	4	9
Usability | 12. Protection against user error	1	6	6	8
Usability | 13. Beauty user interface	2	6	4	10
Usability | 14. Accessibility	1	6	4	10
Reliability | 15. Maturity	2	2	8	6
Reliability | 16. Availability	2	2	9	8
Reliability | 17. Fault tolerance	2	2	8	8
Reliability | 18. Recoverability	2	2	8	8
Security | 19. Confidentiality	1	2	8	9
Security | 20. Integrity	0	2	8	8
Security | 21. No repudiation	1	2	8	9
Security | 22. Authenticity	1	2	8	9
Security | 23. Traceability and responsibility	2	2	8	8
Maintainability | 24. Modularity	2	2	4	6
Maintainability | 25. Reusability	3	2	4	6
Maintainability | 26. Ease of analyze	2	2	2	7
Maintainability | 27. Ease of modifying	3	2	2	8
Maintainability | 28. Ease of testing	2	2	6	7
Portability | 29. Adaptability	2	0	4	6
Portability | 30. Ease of installation	2	0	4	6
Portability | 31. Ease of replacement	3	0	4	8

Fig. 2: Quality Attribute fot IoT systems (See column 4).

On Fig. 2, all parts (1, 2, 3 and 4) were grouped by quality attributes in line with Table 1 (values and colors) and ISO 25010 standard.

Observe that the column that refers to part 4 - system at all - each criterion indicates a level much higher than other originally identified with weak quality attribute. As mentioned, the gap has to drive engineering methods and techniques to obtain satisfactory solutions.

Reliability involves aspects such as availability, fault tolerance and maturity of components of a system. Again, sensors and actuators don´t contribute positively on final quality of system, because their POOR level of quality attributes. On the other hand, cloud infrastructure quality level is EXCELLENT. Project and design should apply special efforts to compensate unbalanced levels of aspects to get VERY GOOD level of quality (see Fig. 2).

One question arises from this context: Considering vulnerable devices, with weak reliability (columns 2, 3 and 4 – Fig. 2), how do we design and select mechanisms to be applied to get the final IoT system quality (column 5)? The answer is to use the method as proposed in [8]. A simple diagram of Fig. 3 describes an example: to control values obtained from sensors (for example, environment temperature), it´s mandatory to avoid wrong values. To do this, three tactics solve this question: one to get values; one to verify the consistency of data; and one to prepare resources to an eventual recovery from a fault execution. Observe that for each tactic, multiple solution mechanisms must be adopted. Then, with this methodological rationale, reliability is built by adding different components like redundancies of hardware, algorithms, traces, and others. In this example of Fig. 3, the Reading Integrity tactic is implemented using multiple sensors, multiple data collect, consolidation algorithms are used to get trusted measures. In addition, the Inconsistency Detection tactic indicates solution mechanisms that can improve values analysis, pattern and historic values, prediction algorithms resulting in precise execution to eliminate false positive or false negative situations. The third tactic is designed considering a compliment to two others tactics: if a wrong value causes fault function, mechanisms of Consistency Recovery tactic directs resources to return from the fault, like effective logs of events, actions alerts e other registries that help to return to an earlier consistent state of the system.

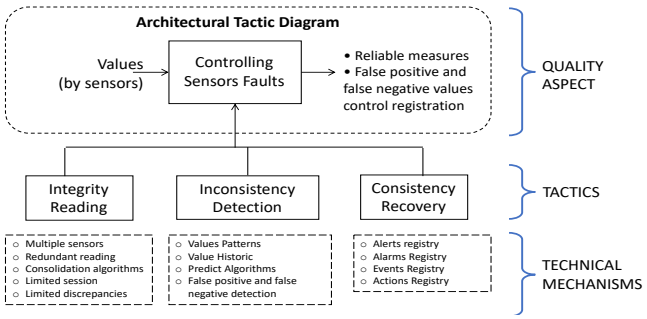


Fig. 3 – Example of architectural tactics model (Adapted from [8]).

Adopting a method like this is unusual on the Ad Hoc project. The consequence is a poor mechanism, without any quality control where sometimes one says that gap is a kind

of "technical debt" to be improved. High risks and low control - points of amateur projects.

VI. Quality Gaps in an IoT System

The Fig. 2 shows 31 quality attributes discussed in this text, considering four views corresponding to four mentioned parts: sensors, actuators, cloud, and the IoT system (ISO/IEC 25010 [9]).

It is possible to find what positive aspects are and what the gaps are. As a practical example, consider a system that needs temperature precision to control a particular room. If the temperature sensor can fail, architectural decisions should include: redundant sensors and historical annotations to avoid fails like erroneous readings, hardware faults, and maintenance actions.

As stated before, IoT systems have an additional complexity because they include in their building blocks very sophisticated components, in addition to the pure simple components. A reliable system does not show false positive or false negative detection based on data collected from sensors (see Fig. 1), or some actuators triggered on a faulty manner. Non-functional requirements must drive all implementation decisions of the system, providing controllable fault-tolerant behavior with a high level of availability.

To control the quality attributes of an IoT system, mostly depicted in this paper, by using architectural methods to get technical decisions to select components and mechanisms that support all requirements, with a focus on non-functional aspects.

VII. Smart Home Study Case

Consider a smart home as an IoT System. It is a logic design of a platform called OKIoT - Open Knowledge IoT Platform Project [10], created for sharing engineering knowledge including hardware and software accelerators. What could happen to a smart home if the internet connection fails? Or if an attacker takes down the local network?

For the smart home scenario [6, 10, 11], the functional utility is dependent on the intended use. As smart homes are shared environments, they are subject to analysis by different users, and each user could have a different goal: energy efficiency, advanced multimedia, safety, and health.

Average response time of a smart speaker could not be satisfactory for intended user journeys, thus reducing Performance Efficiency. On the other hand, compatibility could be high if the smart home system is using an open architecture and not a platform with vendor lock-in.

If the smart home was built not just for the young early adopters, but also for the majority that lives with them, usability could have a high score. System architecture decision could be the adoption of multiple interfaces: physical 3-way switch, radio-frequency controls and sensors, and smart speaker, thus giving accessibility to different users.

In terms of Portability analysis, the comparison and choice between a smart plug or a smart switch for the final IoT product. If portability is a crucial issue, a smart plug could be more easy to install than a smart switch. Maintainability could be high if the smart home is composed of modular solutions and maintenance by parts.

A fault-tolerant driven design could contribute with redundancies that deliver Reliability level improvements, by implementing a three-level operation, with communications that could be performed directly between a mobile app and the IoT module, in an integrated way via local network connectivity, or through the internet.

In order to investigate how a system designer could enhance availability through corrective and preventive maintenances, a method combining a real testbed, Markov chain and fault tree analysis is proposed.

Considering a smart home with 11 deployed IoT modules [10], actuation and sensing services could be available at module, local or internet level (i.e. when the internet is down, some services could still be available as they could be deployed at module level). One possible solution to deal with network instability and unavailability over time is the use of corrective and preventive maintenances performed at module, network or internet level. On this context, how can the designer maximize availability? Time-consuming maintenances could degrade availability if performed too much, but the system could be stuck in a loop if the maintenance is not performed at all.

A. Preventive Maintenance Periodicity Analysis on Testbed

Smart home testbed was monitored during two weeks, from June 6th 2019 to July 13th 2019. Ping response time was used as an indicator of availability (50,000 measurements, an average of 3,000 per connection type), and was performed each two seconds by a python script deployed in a local computer (Dell Inspiron P74G, Intel Pentium i5 7th gen, 4GB RAM), and all connections are 802.11.

On first week, the system was calibrated during the first two days, and then the remaining five days data formed an initial comparative basis. The results of average ping response time of each connection type are summarized on Fig. 4. Internet services depicted are: Google´s Dialogflow (natural language understanding and Google Home integration), Amazon (Relational Database Service), Blynk (IoT platform with integration with Android and iOS applications); modules are: room2, aquarium, room1, storage, kitchen, access, garage, living room, laundry and WIFI modem (responsible for local network maintenance routines); and local is the ping measured between the local computer and router. As expected, internet services had higher response time, modules close to local wifi hotspot (router itself-local, and wifi modem) had the lowest response time, and deployed IoT modules had medium response times.

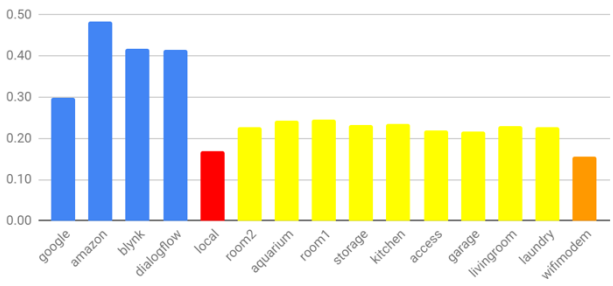


Fig. 4: Average response time (no maintenances) for internet (blue), local (orange and red), and IoT modules (yellow).

Preventive maintenances at local network level (local network restart) were performed on 3 periodicities (6 hours, 12 hours, and 24 hours). Each periodicity was deployed by

two days. As portrayed on Fig. 5, the maintenance which showed the minimum average ping response time was the preventive local network maintenance every 24 hours. Thus, preventive maintenances between shorter time intervals showed no improvement that could justify the inherent unavailability increase (if availability is understood as the probability of a system being available at the time of use, a preventive maintenance between 6 hours could increase the probability of the system being unavailable because of scheduled maintenance).

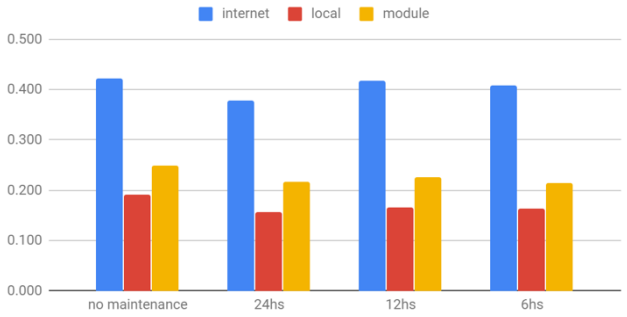


Fig. 5: Average ping response time vs periodicity of preventive maintenances.

B. *Corrective Maintenance Analysis using Markov Chain*

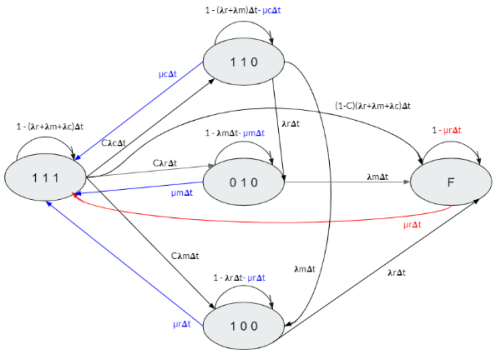
For Markov Chain availability simulations, an experimental setup using ESP8266 and Arduino Nano micro controllers was used in order to measure average time of proposed maintenances and expected failure rates. It was configured in a way that one module measures timing delays while the other is under maintenance.

The measured maintenance delays (used as MTTR – Mean Time to Recovery) were ESP reconnection and restart, Arduino Nano restart. The average was obtained based on 100 tests. In 7 hours, local ping error count (247 events) and internet (120 events) were measured (used as connection failure rates). Network maintenance (another MTTR), based on router restart, was measured using a standard external counter. The estimate for system module failure rate was based on the total time since last restart for the 11 modules present on smart home testbed. Experimental results were the following:

1. MTTR reconnection = 7.37 seconds
2. MTTR network = 80 seconds
3. Connection failure rates = 0.005 failures per second
4. Local network failure rates = 0.0098 failures per second
5. MTTR module = 7.37 seconds
6. Module failure rate = 0.0000008 per second

Simulations were performed on python script deployed in a computer (Dell Inspiron P74G, Intel Pentium i5 7th gen, 4GB RAM). With deltat = 1 second (time interval between simulation iterations), the stop criteria for simulations was of 3,000,000 iterations, which means 3,000,000 seconds = 50,000 minutes = 833 hours = 35 days.

The Markov Chain model depicted on Fig. 6 has 4 states (consider network—module—connection coding for each state).



$$A = \begin{pmatrix} 1-(\lambda r+\lambda m+\lambda c)\Delta t & \mu c\Delta t & \mu r\Delta t & \mu m\Delta t & \mu r\Delta t \\ C\lambda c\Delta t & 1-(\lambda r+\lambda m)\Delta t-\mu c\Delta t & 0 & 0 & 0 \\ C\lambda r\Delta t & \lambda r\Delta t & 1-\lambda m\Delta t-\mu m\Delta t & 0 & 0 \\ C\lambda m\Delta t & \lambda m\Delta t & 0 & 1-\lambda r\Delta t-\mu r\Delta t & 0 \\ (1-C)(\lambda r+\lambda m+\lambda c)\Delta t & 0 & \lambda m\Delta t & \lambda r\Delta t & 1-\mu r\Delta t \end{pmatrix}$$

Fig. 6: Markov model and corresponding transition matrix.

1. From the initial state 111, the connection can be lost by itself, the module can fail or the router can fail;
2. After disconnection, the module or the network can fail, or a reconnection maintenance on IoT module can be performed;
3. After router failure, the IoT module can fail or a network maintenance can be performed;
4. After module failure, it can be restarted or a local network failure might occur;
5. On Failure state, a complete system maintenance (network plus module) can be performed. As the network maintenance is the longest, its maintenance time is considered for full system maintenance.

Simulation results imply: increasing coverage (C) of fault detection has a positive impact on overall availability with corrective maintenances, as expected; given little probability of network failure and longtime requested for its maintenance, using network corrective maintenance actually decreases availability; reconnection is the most recommended corrective maintenance procedure, as module maintenance availability gain was minimal (less than 0.0003%). Markov chain simulation results are summarized on Fig. 7.

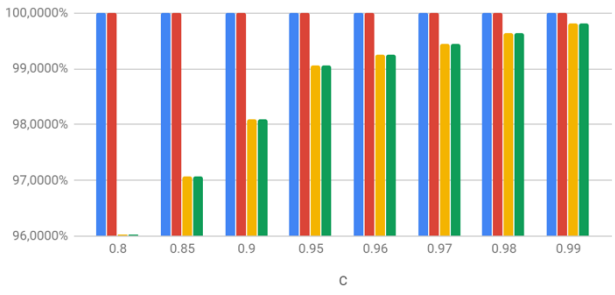


Fig. 7: Availability vs Corrective Maintenance based on Markov Chain simulation. As for the maintenances, reconnection-only is blue, reconnection plus module is red, reconnection plus network is yellow, and network, module and reconnection is green. Availability ranges from ~97% to 99.99%.

VIII. Conclusion

An IoT system must impact people. In a car in movement, in a smart home, in a hospital room, or in a smart city, a digital system will deliver convenient services, useful and helpful functions on a reliable manner. The article evidenced that adopting techniques and methods applying quality attributes could guide the development of reliable IoT systems.

Another result of this research is related to a test methodology. Non-functional testing can be specified using quality attributes as a driver about what, how, and when to test architectural scenarios. The final system can be measured and calibrated to meet the quality goals established by the problem domain of the application.

Smart home case study showed how maintenance procedures could decrease availability, if deployed on ad-hoc manner. Proposed method of assessing preventive maintenance periodicity analysis through testbed, and corrective maintenance effectiveness through Markov chain simulation showed that availability could range between 97% and 99.99% according to the combination of maintenances. The optimal set of maintenances was: preventive maintenance on network level with a 24-hour periodicity, and corrective reconnection maintenance on IoT modules.

It was not part of this paper a detailed description of how to map quality attributes to architecture decisions, but some suggestions may be useful. The technical references [4, 8, 11, 12] bring a powerful impact method to get architectural decisions based on business drivers, non-functional requirements, and tradeoffs. In particular, a way to connect concepts like quality attributes, non-functional requirements, and implementation mechanisms by using architectural tactics is presented.

References

[1] S. Saad and R. Arakaki, "An event processing architecture for operational risk management in an industrial environment", DEBS 2014 - Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, 2014.

[2] Amazon, AWS IoT, 2019.
[Online]. Available: https://aws.amazon.com/iot/. [acessed 6 April, 2020].

[3] Broadband Search, "Mobile Vs. Desktop Usage", 2019;
[Online]. Available: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics. [acessed 6 April, 2020].

[4] R. Kazman, M. Klein and P. Clements, "ATAM: Method for Architecture Evaluation", Software Engineering Institute, Technical Report CMU/SEI-2000-TR-004, 2000.

[5] Gartner, "Technologies and applications adoption and tendencies", Publication on gartner.com; 2018, 2019.
[Online]. Available: https://www.gartner.com/en/newsroom/press-release/2019-02-18-gartner-identifies-top-10-analytics-tecnology. [acessed 6 April, 2020].

[6] A. J. Ansari, A. Sathyamurthy and R. Balasubramanyam, "An Open Voice Command Interface Kit", in IEEE Transactions on Human-Machine Systems 46 (3), 2016, pp. 467-473.

[7] R. Pressman and B. Maxim, "Software Engineering: A Professional Approach", 2016, McGraw Hill.

[8] L. Bass, P. Clement and R. Kazman, "Software Architecture in Practice", SEI Series, Addison Wesley, 2013.

[9] ISO/IEC 25010, "Software Product Quality Model", ISO/IEC, 2011.

[10] V. Hayashi, R. Arakaki and R. M. Andrade, "OKIoT Open Knowledge IoT Project: Smart Home Case Studies of Short-term Course and Software Residency Capstone Project", IoT BDS 2020 – 5th International Conference on Internet of Things, Big Data and Seccurity, Czech Republic, May 2020, in press.

[11] R. Arakaki, R. Manzan and V. Hayashi, "Technical Documentation of Software Engineering for IoT System", in MBA LARC USP, Department of Computer Engineering of Polytechnic School, University of São Paulo, 2017, 2018, 2019.
[Online]. Available: https://ae4.tidia-ae.usp.br/portal. [acessed 6 April, 2020].

[12] ISO/IEC 10746, "Open Distributed Processing – Reference Model: Architecture", ISO/IEC, 2009.