Service-Based Testing of IoT Devices

INTRODUCTION

The Internet of Things has tremendous promise, some of which has already been realized and valued by consumers everyday. But unfortunately, the reality is that many IoT applications and devices are not ready for 'prime time.'

A recent study found that 80 percent of IoT apps are not being tested for security flaws. In another study, the Barr Group found that 56% of embedded device developers don't review source code for security vulnerabilities and 37% don't have a written coding standard. These are not encouraging statistics, and it's clear that IoT device manufacturers need to take quality, safety, and security more seriously. Test automation is one important step in order to make sure testing is being done more rigorously, consistently, and thoroughly. Testing, especially for security vulnerabilities, is often seen as too costly and complex, and is therefore rushed or overlooked altogether.

But it's an expensive mistake to let your customers (and attackers) test your IoT device security for you.

Although "things" is the term used in the Internet of Things, the reality is that consumers and enterprises aren't interested in just things or devices. The real promise of IoT is in the data these devices produce and the actions they take. Even the smallest sensor is providing a useful service, and although an embedded device, IoT, or mobile app developer might not tend not to think in terms of services, it's important to change that mindset in order to meet the functional, quality, performance, and security requirements demanded in the fully-connected world of IoT. To truly advance the quality of IoT software development, a services perspective and orientation is needed.

IOT IS MADE UP OF SERVICES

Despite the name that has stuck in IoT, the "things" themselves are not the star of the show. More key to IoT is the information gathering, control of key infrastructure, and the sensing of the real world that these devices provide.

Consumers aren't interested in just the temperature of one room in their house or the video feed from a single camera. They are interested in the next level -- making sure their security system detects movement all around the house or that their air conditioning is maintaining a comfortable temperature. Enterprises aren't interested in the output of a single logic controller in a factory but rather production throughput of an assembly line. **This is an important change in perspective because it forces device developers to better appreciate the context of their product and its use cases.**

Believe it of not, your device or application is probably part of a broader service. Individual embedded devices may not traditionally be considered part of a service; however, connectivity into larger systems means they should be. For example, in an automobile, the role of the engine control unit (ECU) alone is to ensure proper combustion and emissions in the engine, but the car is tracking fuel economy, using the ECU, and reporting it to a central server over a wireless connection. This mileage data is then being used to plan routes and estimate operating costs. All of a sudden, the ECU is a critical leaf node in a business decision making process.

Adopting this perspective widens the context of an individual device and its scope of operation, affecting the approach of overall system design, as we move from device-centric thinking to service-centric:

- **Conglomeration:** The Internet of Things consists of too many "things" for each to be valuable on their own. Devices need to be organized together in order to provide useful information at a higher level. For example, an HVAC system doesn't need to report the temperature in every room. Individual sensors report to a supervisory control system (like SCADA systems in industrial control) which makes local decisions, which in turn are reported to higher level systems that may be offsite.
- **Self-monitoring:** Higher-level business decision-making processes would be overwhelmed in a sea of data if every individual sensor reported everything, all the time. In our HVAC example, a localized supervisory control system can maintain building temperature based on an amount set by a centralized process (for example, based on weather and electricity rates). The enterprise-level systems would therefore rely on a service provided by the HVAC system, on a building-by-building basis, that reports critical information such as energy usage.
- **Interchangeability:** Over time, the services provided by this conglomeration of devices becomes more valuable than the devices themselves. The individual sensors and controllers can be swapped out wholesale with another product if the overall business goals are still met. If the quality of service remains the same, or better, the hardware is interchangeable. On the surface, this might seem like a bad thing for device manufactures, and for some it certainly is. But smart companies that understand the importance of services and compete on the quality of those services become the market leaders.

WHY SERVICE-BASED TESTING IS CRITICAL FOR IOT SUCCESS

Once adopting a service-centric approach, it makes sense that design, implementation, and testing follow suit. Realizing that the service provides the business value, it becomes critical to ensure that devices are meeting requirements in this respect. Obviously, testing functional operation at unit, subsystem, and system level are still important, but broadening the scope of testing provides immediate benefits.

Instead of viewing system quality in terms of meeting individual device requirements, the scope is broadened to consider the **quality of the services** provided. In the HVAC example, a new temperature sensor might be lighter, lower cost, with long battery life, and have excellent wireless range. But how well it works with the building-wide control system is just as important as all of the new features.

Testing at the **service level** ensures **non-functional requirements** are met. For example, performance and reliability is difficult to assess at the device level or during software unit testing. Service-based testing can simulate the operational environment of a device to provide realistic loads. In the HVAC example, the new temperature sensor can be tested with varying request rates to see if it meets performance requirements.

Cyber attacks against IoT systems will originate from the network itself, by attacking the exposed APIs. Service-based testing can create simulated environments for robust **security** testing, either through fuzzing (random and erroneous data inputs) or denial-of-service attacks. A new temperature sensor in the HVAC example might operate correctly with expected requests but crash when overloaded. An attacker might be able to exploit this to overload the system and cause an outage.

TEST AUTOMATION TO THE RESCUE

Automation in the software development process becomes critical as the scale of the IoT implementation rises. Security and non-functional requirements become more important as connectivity and scale increases. Service-based automated testing becomes critical during integration and system testing phases, while also enabling testing for security, stability, and performance.

Let's take a look at Parasoft's depth and breadth in test automation, as shown in Figure 1, as it applies to each phase of the SDLC. The key takeaway is how each solution complements the other and scales as the product grows. Unit testing is complemented by static and run-time analysis. During integration, unit testing progresses to API and service testing tools, which then progress to service virtualization.

© 2017 Parasoft Corporation

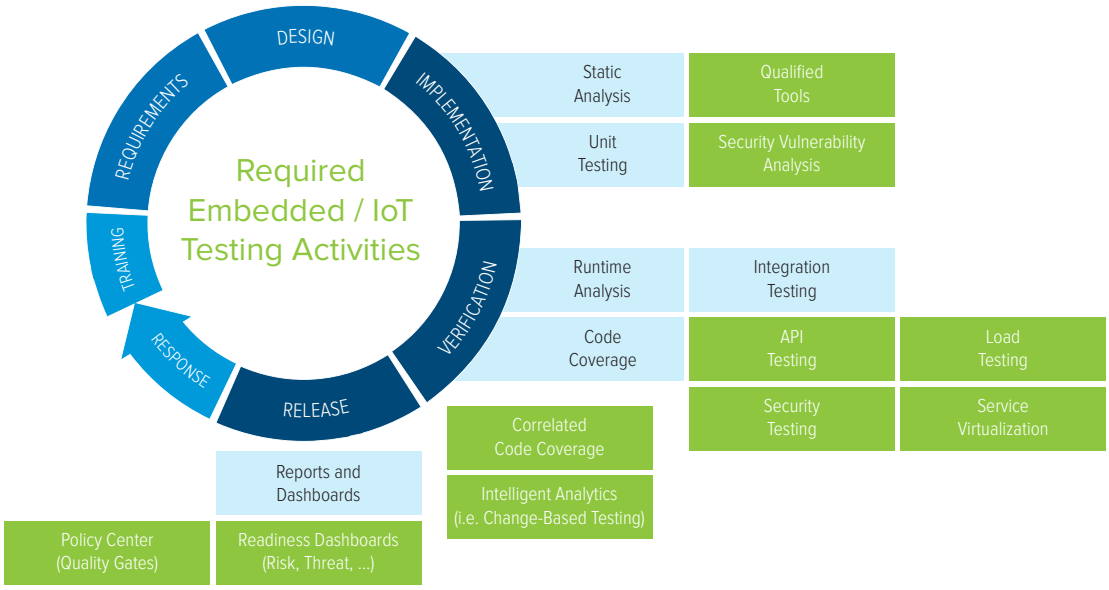

Figure 1: *Test automation solutions and where they are introduced into the software development lifecycle.*

Most developers are familiar with some aspects of test automation; however, many are not thinking in terms of services or services-based testing and virtualization. As the paradigm shifts from "things" to "services," the benefit of service-based testing becomes more apparent.

RISK AND COST REDUCTION WITH AUTOMATION

As a product progresses through the development process into integration, testing becomes more complicated and expensive. This is also the stage that many critical bugs are found, including security and performance issues not detectable at the unit level. A big risk is the availability of testing staff, labs, and hardware resources.

The solution here is automation. Automation provides a way to increase testing productivity, repeatability, and scope of system testing. An automated, services-based approach to testing is critical for the success of a newly-developed IoT device.

Benefits include the following:

- **Increased Testing Productivity:** Automation can assist in the generation of service-based tests, and testing loads that are repeatable and extensible. Runtime analysis tools are run alongside live tests to detect and trace errors. Defects are more easily caught and fixed than with manual testing. Once a fix is made, automation provides easy regression tests. Repeating tests as part of a continuous testing, agile, or iterative development process is supported.
- **Removing hardware and lab resource bottlenecks:** One of the biggest issues in test scheduling is the availability of dependent components. These may be other devices on the network, or higher level functions in the IoT network. Simulating these dependencies removes testing bottleneck while making an entire virtual lab available on every tester and developer's desktop.
- **Scaling to current and future needs:** As system integration progresses, the scope of testing increases to include more and more dependent components. Scalability is key in IoT systems, since the device under test is expected to perform in a highly-complex environment. During the device's lifecycle environment, complexity will increase, and automated testing must scale with the product.

© 2017 Parasoft Corporation

- **Practical and realistic performance and security testing:** Development teams do what they can to test performance and security during integration, but without automation, it's often time-consuming and expensive to set up realistic scenarios in the lab. A services-oriented approach provides a framework for specifying and verifying performance and testing security.
- **Real-world test environments:** Developers need to quickly simulate lots of environments that enable them to test both expected and unexpected functional scenarios. Service-based testing enables teams to simulate services that have malfunctioned, ensuring their software performs correctly.

BEYOND FUNCTIONAL TESTING: SECURITY AND PERFORMANCE

A pure functional test isn't enough to bring a device to market. Non-functional requirements such as security and performance are critical, but are some of the hardest characteristics to test for. A device that has poor performance or poor security simply isn't competitive, but meeting time-to-market constraints and properly testing performance and security is a serious challenge for IoT devices. Service orientation provides a common way to specify requirements (e.g. performance on a per-service bases) and for testing security (e.g. attacks exploiting exposed services and APIs).

Performance and Load Testing

Performance and security requirements are likely to be expressed in terms of a quality-of-service statement. For example, an HVAC system might be required to maintain building temperature to 75 degrees within two hours based on a 5-degree difference in outside temperature, reporting current temperature every 10 seconds. If designing a thermostat for this product, you can test the functionality at the individual level and perhaps the performance too. However, if a deployed system consists of hundreds of thermostats, performance of the device is just a small part of a complex network of other devices.

Security Testing

Security requirements are often system-level and vague. In the HVAC system example, a thermostat might be required not to fail under heavy network loads. A denial-of-service attack relies on flooding the target with heavy traffic, sometimes with malformed packets. Penetration and fuzz testing is used in these cases to make sure devices can withstand a hostile network environment. Scalability available in a virtual environment exceeds what's possible with real hardware.

SERVICE VIRTUALIZATION AND VIRTUAL LABS

The next logical step for automated service-based testing is virtualization. A complete virtual environment is possible by simulating all dependent components plus a full suite of test stimuli. The benefit of service virtualization is not just realistic and repeatable test environments but the ability to duplicate and deploy at will thus creating a "virtual lab" — a lab that's just as effective as the real thing but at a fraction of the cost.

A real test lab requires the closest physical manifestation of the environment an IoT device is planned to work in, but even in the most sophisticated lab, it's difficult to scale to a realistic environment. A virtual lab fixes this problem by evolving past the need for hard-to-find (or perhaps non-existent) hardware dependencies. Using sophisticated test automation tools is key to leveraging virtual test labs. These tools include:

Service Virtualization

Simulate all of the dependencies needed by the device under test, in order to perform full system testing. This includes all connections and protocols used by the device with realistic responses to communication. For example, service virtualization can simulate an enterprise server back-end that an IoT device communicates with to provide periodic sensor readings.

Service and API Testing

© 2017 Parasoft Corporation

API testing provides a way to drive a device under test in a manner that ensures the services it provides (and APIs provided) are performing flawlessly. These tests can be manipulated via the test automation platform to perform performance and security tests as needed.

Runtime Monitoring

Runtime monitoring detects errors in realtime on devices under test, capturing important trace information. Memory leaks, for example, which can remain undetected in a finished product, can be caught and resolved early and cheaply with runtime monitoring.

Test Lab Management & Analytics

Control the virtual lab(s) with sophisticated test lab management and analytics. Once virtualized, the entire lab setup can be replicated as needed, and test runs can be automated and repeated. Analytics and executive dashboards provide necessary summaries of activities and outcomes, and high-level information to take to stakeholders.

The edge computing IoT ecosystem is shown below in Figure 2, depicting a typical environment in which embedded IoT devices are deployed. Sensors and control devices communicate information to the Edge, which is a series of appliances or applications that can receive information and use logic to communicate back to a device or up to the cloud. The cloud then has higher-level logic that allows it to act upon that information. The cloud is a set of services — microservices, connections to databases, additional logic, or third-party services — a complex web of functional building blocks, shown below to the right.

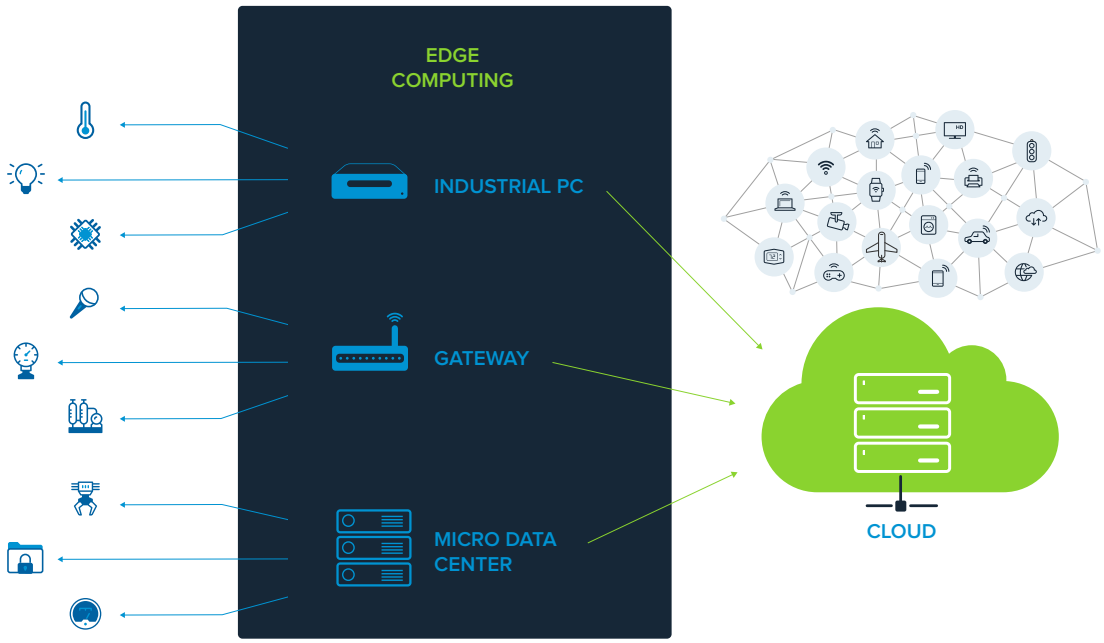

Figure 2: *A typical IoT ecosystem in which embedded devices would be deployed*

When it's time to test in the IoT ecosystem, testing is required at many layers. To test new functionality introduced in the gateway, for example, validation that the gateway can receive information from sensors, and can communicate that in the way you've built the business logic.

In order to validate all of this complexity, Parasoft Virtualize (which simulates required dependencies) and Parasoft SOAtest (which drives tests) are used to simulate those inputs. These tools provide simulations of realistic calls from the devices over the network (whether they are protocols like REST/HTTP, or IoT popular protocols like CoAP, XMPP, or MQTT), and test that the device under test (the gateway in this example) is communicating with the cloud services appropriately, by validating the responses that come back from SOAtest. Figure 3 below shows an example of how a virtual lab environment can be created for edge devices under test.

© 2017 Parasoft Corporation

If there are external ways of communicating information into that gateway, those calls can be simulated as well. Parasoft Virtualize is designed to stabilize the testing environment, to create predictable responses to requests that leverage test data from SOAtest, fully testing the gateway and services.

Finally, the top-level services might be communicating back to the edge, and back to other sensors and external actors, and it might be important to know that the flow from your inputs are making their way through the environment back to the back-end systems. Parasoft Virtualize is used to simulate the receiving of those calls down to the edge (down to the IoT devices) and then relay that information back to SOAtest to confirm that the call made the round trip and behaved the way it was expected inside the IoT ecosystem. The combination of Parasoft Virtualize and SOAtest provides full control to test the whole environment, even within the complexities of an IoT ecosystem.

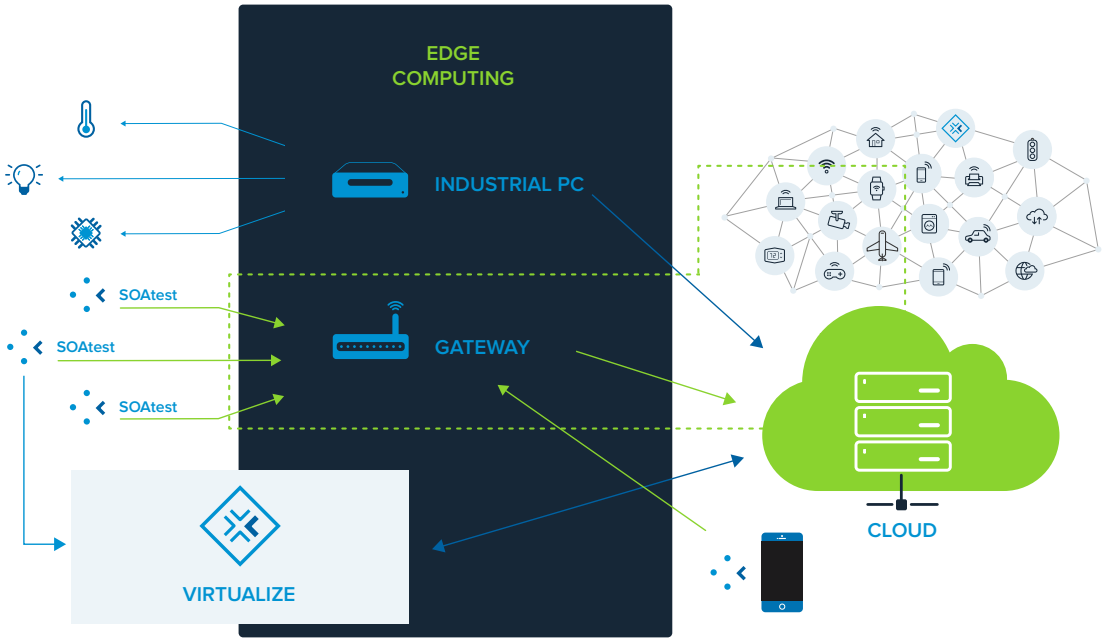

Figure 3: *The role of Parasoft's Virtualize and SOAtest tools create a virtual lab environment for an edge device under test*

VIRTUAL LABS REDUCE TIME, COSTS AND RISK

Normal test environments are expensive, probably more so than most development managers forecast for. A study by voke Research found that the average investment in a pre-production lab was $12 million. In terms of time, the average time to provision the lab was 18 days and a further 12-14 days were spent on configuration. These labs take lots of time and money to set up, and even after that, they act as the bottleneck for testing due to limited access. Further, day-to-day operational costs of physical labs are significant. In most cases, duplicating a physical lab to increase test throughput is cost prohibitive.

The benefits of service virtualization include improving access and availability of testing devices, with better control of behavior of virtualized dependencies, reducing costs and increasing test speed.

The benefits of the virtual IoT test lab include:

- **Improved quality through better, more complete testing:** Service-based testing ensures that key use cases are exercised and perfected. Automated performance tests ensure stability and reliability under heavy load. In addition, runtime monitoring ensures that hard-to-find bugs are detected and traced.
- **Improved security with automated penetration tests that simulate malformed data:** Load testing can simulate denial of service attacks and runtime monitoring can detect security vulnerabilities. Test repeatability ensures

© 2017 Parasoft Corporation

that each iteration, patch, or release is tested in the exact same manner. In addition, test development and manipulation (i.e. improving and creating new tests) is simplified.

- **Reducing testing time, risk, and cost by removing the need for expensive dependencies required for complete systems testing:** Automation provides repeatability and consistency that manual testing cannot, while providing better and more complete testing. Virtual labs greatly reduce the provision time needed for physical lab setups, impacting total test time.

CONCLUSION

Realizing that IoT is really about the services results in better, differentiated embedded devices in the new connected world they operate in. Manufacturers that focus on the services are less likely to be interchanged with equivalent hardware. In order to achieve the required performance, quality of service, and security that IoT systems require, service-based testing is essential.

Test automation is a proven approach to reduce costs and risk. The next big step to improve quality and security for IoT devices is to use virtual labs that combine service virtualization, service-based testing, virtual lab management, and runtime monitoring. This greatly reduces the provisioning and configuration costs while great increasing the quality of the testing being performed.

REFERENCES

"80 percent of IoT apps not tested for security flaws, study", SC Magazine, January 20, 2017, https://www.scmagazine.com/iot-app-remain-untested-and-lack-of-urgency-to-fix-problem/article/632714/

"Barr Group's 2017 Embedded Systems Safety & Security Survey Reveals Vulnerabilities in the IoT", Barr Group, February 24, 2017. https://barrgroup.com/content/barr-groups-2017-embedded-systems-safety-security-survey-reveals-vulnerabilities-iot

"voke Market Snapshot Virtual and Cloud-based Labs", voke Research, https://www.vokeinc.com/average-virtual-lab-management-roi.html

ABOUT PARASOFT

Parasoft helps organizations perfect today's highly-connected applications by automating time-consuming testing tasks and providing management with intelligent analytics necessary to focus on what matters. Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software, by integrating static and runtime analysis; unit, functional, and API testing; and service virtualization. With developer testing tools, manager reporting/analytics, and executive dashboarding, Parasoft supports software organizations with the innovative tools they need to successfully develop and deploy applications in the embedded, enterprise, and IoT markets, all while enabling today's most strategic development initiatives — agile, continuous testing, DevOps, and security.

www.parasoft.com

Parasoft Headquarters:
+1-626-256-3680

Parasoft EMEA:
+31-70-3922000

Parasoft APAC:
+65-6338-3628

PARASOFT®
Automated Software Testing

Copyright 2017. All rights reserved. Parasoft and all Parasoft products and services listed within are trademarks or registered trademarks of Parasoft Corporation. All other products, services, and companies are trademarks, registered trademarks, or servicemarks of their respective holders in the US and/or other countries.