Puello, P., Leal, H., & Cabarcas, A. (2022). Propuesta de plan de pruebas orientado a proyectos de Internet de las Cosas. *Teknos Revista Científica,* 22 (2), 59–73.

Propuesta de plan de pruebas orientado a proyectos de Internet de las Cosas

Proposal for a test plan oriented to Internet of Things projects

Plinio Puello[1], Humberto Leal[2], Amaury Cabarcas[3]

[1] *Sytems Engineering Department, University of Cartagena, Cartagena de indias-Bolívar, Colombia.*
ppuellom@unicartagena.edu.co
[2] *Sytems Engineering Department, University of Cartagena, Cartagena de indias-Bolívar, Colombia.*
hlealb@unicartagena.edu.co
[3] *Sytems Engineering Department, University of Cartagena, Cartagena de indias-Bolívar, Colombia.*
acabarcasa@unicartagena.edu.co

Recibido: 15/ago/2022 Revisado: 30/sep/2022
Aceptado: 30/oct/2022 Publicado: 30/dic/2022

Resumen: Este estudio introduce un plan de pruebas orientado a proyectos basados en el Internet de las Cosas (IoT) apoyando la implementación de pruebas de calidad. Lo anterior está motivado para seguir verificando y validando soluciones tecnológicas en este ámbito. Este método propuesto apunta a apoyar la implementación de procesos rigurosos de aseguramiento de la calidad mediante el desarrollo de un plan de pruebas específico para proyectos de IoT. Se seleccionó una metodología mixta basada en el modelo de cascada y las actividades de Scrum para tratar entre los requisitos rigurosos, los procesos de aseguramiento de la calidad, y la agilidad en el desarrollo de nuevos entregables a través de iteraciones. Además, se identificaron las actividades asociadas a las capas de percepción, red y aplicación de la arquitectura utilizada para los proyectos de IoT. Se utilizó la técnica de juicio de expertos para validar la pertinencia de las pruebas propuestas de acuerdo con la arquitectura de tres capas establecida para los proyectos basados en el Internet de las Cosas. Como principal contribución, este trabajo presentó un plan de pruebas basado en las necesidades de las capas arquitectónicas y la metodología mixta seleccionada para el desarrollo de proyectos basados en IoT. Se identificaron dos criterios que influyen en la toma de decisiones sobre las pruebas recomendadas, considerando las capas de arquitectura de los proyectos basados en el marco del Internet de las Cosas.

Palabras clave Plan de prueba, internet de las cosas, metodología de desarrollo de software, arquitectura.

Abstract: This study introduces a test plan oriented to projects based on the Internet of Things (IoT) to support the implementation of quality tests. The above is motivated to verify further and validate technological solutions in this area. This proposed method aims to support the implementation of rigorous quality assurance processes by developing a specific test plan for IoT projects. A mixed methodology based on the waterfall model and Scrum activities was selected to deal between rigorous requirements, quality assurance processes, and agility in new deliverables development through iterations. Besides, activities associated with the perception, network, and application layers of the used architecture for IoT projects were identified. The expert judgment technique was used to validate the relevance of the proposed tests according to the three-layer architecture established for projects based on the Internet of Things. As the main contribution, this work presented a test plan based on the architectural layers needs and the mixed methodology selected for developing projects based on IoT. Two criteria were identified that influence decision-making about recommended tests, considering architecture layers of projects based on the Internet of Things framework.

Keywords Test plan, internet of things, software development methodology, architecture.

1 Introduction

Quality assurance is crucial in developing software that meets users' functional and non-functional requirements (Leach, 2016). However, this aspect has not received much attention in the Internet of Things (IoT) area, so some unsolved issues might require consideration (Bures et al., 2019). Regarding IoT, the quality assurance process has generated challenges not previously addressed due to the heterogeneity of used devices, the need for cloud services, available communication protocols, and privacy/security mechanisms (Bures et al., 2019)(Muccini & Moghaddam, 2018).

For compliance with the quality assurance policy in software development, many companies and authors have suggested testing quality measurements in technological solutions based on IoT. The authors could identify Model-Based Testing, Model Checking, IoT reliability models, protocol testing, IoT usability, and IoT performance(Bures et al., 2019).

Several IoT-based project architectures have used layered methods (Muccini & Moghaddam, 2018)(Al Asif et al., 2019)(Zheng et al., 2016), starting with three known approaches: (i) perception, (ii) processing-storage, and (iii) application. These have added increasingly specialized functions to include up to six (6) layers that vary between business, network, and adaptation. However, different proposals were found when reviewing the most used architectures in IoT projects. However, other proposals were found when examining the most used architectures in IoT projects.

Other architectural proposals are generally comprised of three layers (Yaqoob et al., 2017): (i) perception, (ii) transport, and (iii) application. Subsequently, more layers are included with functionalities oriented to security, specific provider cloud services, and network scalability.

Studies with more specific application approaches have addressed their architectures based on the requirements they must meet. An example is an architecture that supports accurate agriculture applications based on the lambda architecture approach. This tool would also be viable for big data applications, with data processing and visualization needs through specialized tools (Quiroga Montoya et al., 2017). Similarly, other authors have proposed an architecture for IoT and Big Data solutions, with a similar approach to the previously described system. In this case, the layers implicated IoT and network devices, IoT gateway, cloud storage, Big Data analytics, and API management with the dashboard as upper layers, as shown in Fig. 1(Marjani et al., 2017).

Some architectures consider three main components in their design: hardware, communication, and webserver. The above implicates supporting accurate agriculture IoT projects to automate crop irrigation based on different environmental parameters. Specifically, the hardware component was composed of sensors and actuators. The above was followed by the communication network component for data transport using the Message Queue Telemetry Transport MQTT protocol. Finally, the server component deals with data storage and processing, along with the presentation of the generated information to the end-user (Cabarcas et al., 2019).

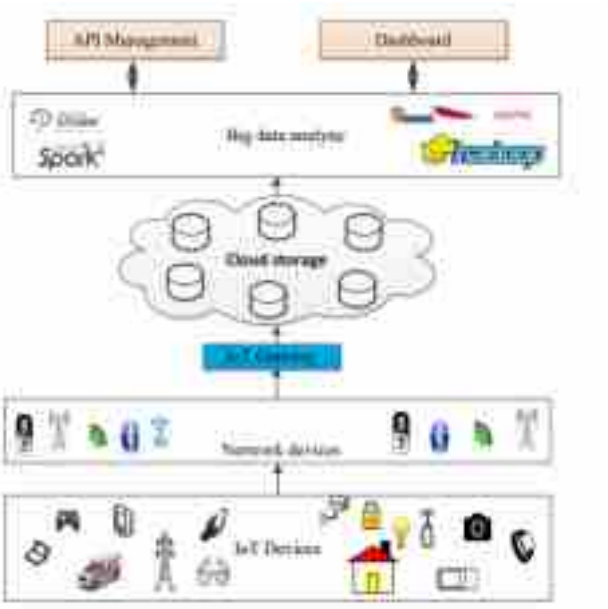

Fig. 1: IoT architecture and big data analytics (Marjani et al., 2017)

According to the studies described above, it is evident that there is no available in the literature an overall architecture for this type of project, which implicates an opportunity to generate new straightforward methods. Nevertheless, the three (3) layer architecture was identified as a modern and adaptable solution that plays a role as a starting point, considering the perception, network, and application

layers. Next, more specialized layers can be added depending on each IoT project (see Fig. 2).

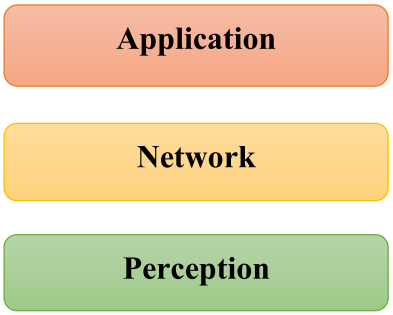

Fig. 2: Three-layer IoT reference architecture

Therefore, this study's reference architecture, which guides the testing activities, comprises the perception, network, and application layers. The first layer considers sensing devices, actuators, and microcontrollers. The network layer contains those devices and hardware responsible for sending data from the ground, such as antennas, gateways, and intermediate services. It should be highlighted the benefits of the employed communication protocols. The application layer focuses on services, storage, processing, and presentation to the end user.

The ecosystem of tools and methodologies for testing IoT systems has been sufficiently addressed based on the reported literature (Bures et al., 2019), (Muccini & Moghaddam, 2018)(Dias et al., 2018). Still, some approaches support the tests mentioned above (Dias et al., 2018)(Kim et al., 2018). Testbeds or test environments, in conjunction with emulators and simulators, are presented as essential alternatives to achieve quality assurance in different components of an IoT solution (Dias et al., 2018).

This article proposes a test plan oriented to IoT projects to support quality verification and validation steps, complying with the assurance policies established in the process models implemented in recognized companies of IoT solutions development. Test strategy, resources, evaluation, and documentation are considered for the test plan formulation. These are articulated with activities developed during the construction of IoT solutions, for which a mixed methodology is proposed combining the classic waterfall model (Ruparelia, 2010) and Agile Scrum (Schwaber, 1997).

The proposed test plan's consistency is not limited to the mixed methodology. The method began from a three-layer reference architecture to identify essential components regarding an IoT solution: (i) perception, (ii) network, and (iii) application, according to Fig. 2. In this way, this work suggests the types of appropriate tests to be performed for each evaluated component. The above is stated considering a narrow dependence between the phases of the proposed mixed methodology, which influences the type of tests that will finally be selected given the stage in which the project progresses.

In addition to the above, the suggested reference architecture (Fig. 2) and the tests to be carried out at each stage must be adapted according to the specific requirements, quality attributes to be met, and the scope and complexity of the project. Therefore, specialized layers for data processing and analysis will be relevant for projects that require it [8], as well as business logic, adaptation, or data management are some of the specialized layers that can be used (Muccini & Moghaddam, 2018), (Al Asif et al., 2019). Depending on the focus on interoperability, security, and privacy, which are characteristic challenges of IoT projects, they indicate the need to integrate tests that evaluate these aspects in the test plan (Bures et al., 2019), identified in section 3.

In summary, this work is structured as follows: first describes the methodology that shows the proposed test plan proposal. Follows depicts the results based on the proposed mixed methodology description, test plan overview, and the articulation for this proposal implementation considering defined layers of a generalized architecture based on IoT. Finally, the obtained results and contributions are further discussed, conclusions and directions for future works are given.

2 Methodology

(a) Phase 1 – Methodologies and common activities selection

Two software development methodologies were selected: the waterfall (Bures et al., 2019) and scrum models (Schwaber, 1997). The use of a mixed approach has been previously analyzed, and advantages have been recognized when combining these methodologies (Dody-Lesmana et al., 2016)(Ching & Mutuc, 2018)(Kuhrmann et al., 2017). Some IoT projects have recognized the specific advantages of the waterfall model and Scrum,

respectively (Wardana et al., 2019)(Vogel et al., 2019)(Sachdeva & Chung, 2017). Regarding the IoT framework, it has been identified that there are several alternatives for project development, and previous investigations have confirmed the gained leverage of using mixed approaches (Jacobson et al., 2017).

Since the waterfall model uses rigorous documentation and planning processes, together with the phases that allow quality assurance through verification and validation (Ruparelia, 2010), it was pertinent to select this model to support the methodology with which the management plan was proposed as evidence of the present study.

As a complement to the above, the changing requirements of IoT projects were considered. This is due to the heterogeneity of devices, coming from different manufacturers, variety of communication protocols and standards (Bures et al., 2019), (Muccini & Moghaddam, 2018). In summary, it allows the Scrum methodology to be identified as supporting the adaptations required in IoT projects (Schwaber, 1997).

The above allows for justifying the waterfall and scrum model selection as baseline methodologies in this work.

Then, the activities carried out in each of its phases were detailed. The common activities in both methodologies were identified, considering that a hardware-software prototype or embedded system based on IoT is being developed. Thus, a mixed method for developing IoT solutions is selected.

(b) Phase 2 - Construction of a test plan overview

In this phase, a general view of the test plan was prepared, including (i) Test strategy, (ii) test resources, (iii) evaluation of tests executed, and (iv) documentation. These present various aspects of the IoT project-oriented test plan based on the mixed methodology described above.

(c) Phase 3 - Test plan activity cycle

The principal needed activities to ensure IoT-solutions quality was established from the software cycle. Besides, the design artifacts generated by each activity are identified.

(d) Phase 4 – Test plan proposal

The types of evidence that can be considered in an IoT project based on the previously documented three-layer architecture (perception, network, and application) are acknowledged. Finally, the method defines a group of tests reported in the literature and tools to be executed in each layer.

The expert judgment technique was also used to validate the relevance of the proposed tests according to the three-layer architecture established for projects based on the Internet of Things (Monroy et al., 2017)(Cabero-Almenara & Llorente Cejudo, 2013). The population of experts who participated comprised five professionals with experience in software engineering, the internet of things, internet service engineering, and business architecture. The instrument used allowed evaluation of the relevance of the tests proposed for each layer of the architecture, using an assessment scale distributed by the following response options: high level of relevance (HL), medium level of relevance (ML), and low level of relevance. relevance (LL).

3 Results

3.1 Methodology and common activities selection

The waterfall model is used as a basis for other software development methodologies. This framework is characterized by several aspects, including evaluation, requirements definition, analysis, design, expansion, and implementation stages (Ruparelia, 2010). This approach is recognized as the most advanced and straightforward among available methodologies (Huo et al., 2004). Therefore, this method is used in the proposed test plan.

The Scrum methodology addresses its development process through a series of short iterations, called sprints. The above is under the principle that the development process is unpredictable and complicated. This approach is characterized by its flexibility and affinity to satisfy initial requirements and adapt to new ones that arise during the process (Schwaber, 1997). Also, it is

exceedingly feasible for small development projects (Ruparelia, 2010).

The Scrum development methodology has been previously used in IoT projects (Esteves Maria et al., 2015). Other IoT projects have been developed using the waterfall model (Afif et al., 2020)(Shah et al., 2018). In both methodologies, functional systems were built, merging hardware and software aspects required for an IoT solution.

Agile methodologies are appropriate for small projects with changing requirements, strict time and budget limits, and developers' small pool. Factors like interoperability, scalability, and security are essential for IoT projects, focused on meeting functional requirements and end-user satisfaction. According to the above, it is pertinent to establish good scenarios for adopting each of these methodologies.

An agile approach is appropriate in the lower-level implementation, development, and design. The rigorous documentation and planning that characterize the waterfall model make it suitable for projects where quality control is a primary concern. It is also ideal when the requirements are clearly defined (Singhto & Denwattana, 2016). Otherwise, the Waterfall tends to be slower than other agile methodologies, including the Scrum model (Singhto & Denwattana, 2016). In contrast, adopting the Waterfall model is more advantageous in high-level planning and design phases. Finally, it is suggested to use the waterfall model in the system and integration testing phase (Hayata & Han, 2011).

Although the Waterfall and Scrum methodologies are opposites, they present common characteristics and settings (Huo et al., 2004). In practice, these methods tend to be hybrid due to experience and pragmatism (Kuhrmann et al., 2017)(Singhto, 2016).

Consequently, a mixed methodology is selected to achieve the benefits of using the Waterfall and Scrum models f-3or IoT system construction [30].

Table I shows the phases that comprise the Waterfall and Scrum methodologies. The used stage was selected, which points to a pragmatic development process that leads to quality results in IoT projects.

The selected mixed methodology obeys Waterfall and Scrum combination in a V model represented by Waterfall-up-front, agile-methods, and waterfall-At-End (Hayata & Han, 2011), shown in Fig. 3, considering the above. The preceding seeks to use phases and practices of the waterfall and scrum model with more significant benefits.

Initially, the waterfall model is used for requirements analysis and initial or high-level design, where the architecture is defined. Subsequently, the Scrum methodology is used in the low-level design, implementation, and unit tests. In projects where quality assurance is essential, it is addressed through a Waterfall methodology via integration and system tests(Hayata & Han, 2011).

The above allows the waterfall model, in the beginning, reducing delivery times. Using Scrum in development, designers can increase software development speed, avoiding delays, and recalculating delivery dates often observed in traditional methodologies. Otherwise, high-level tests and acceptance criteria are developed using the waterfall model to evaluate the resulting product quality (Singhto & Denwattana, 2016).

Table 1. Methodologies: Waterfall, Scrum, and mixed.

Waterfall	Scrum	Mixed
Requirements	Planning	Requirements
Design	High-level architecture	High-level design
Implementation	Sprint – Development, Design, Review.	Sprint - Development, low-level design, unit testing, customer review.
Test	Sprint – Test	Testing – Integration, and system.
Deployment	Closing	System deployment or upgrade

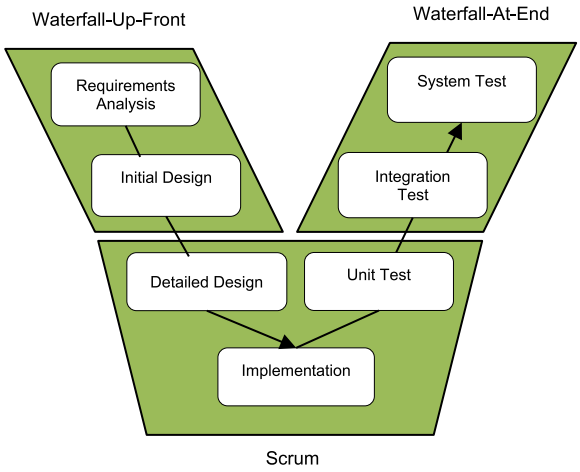

Fig. 3**:** Mixed methodology selected from the waterfall and scrum models (Hayata & Han, 2011).

3.2 Construction of a test plan overview.

Fig. 4 shows the assembled sections representing how the test plan components are composed, considering strategies and resources for the tests, evaluating their development, and documentation or artifacts resulting from the previous steps.

The testing strategy focuses on project test analysis, design, implementation, and execution. These act as the basis or starting point for the other test sections. Mainly, it focuses on the required techniques, types of tests, tools, and expected deliverables. The test resources allow identifying those factors of human and non-human nature, such as hardware, software, support tools, and test configuration, among others. It means those involved resources that are needed and intervene in the testing process.

The test evaluation allows for establishing an acceptance degree based on the methods defined for this purpose: results assessment, test document description, and test inputs-output analysis. The previous components and needed types of tests are part of the overall test plan based on the results obtained in this work, organized according to the IoT system's layer. Finally, the artifacts or parts of information that will allow the specification and test execution documentation are part of the documentation component. These included release notes, test sets, checklists, test reports, and test codes.

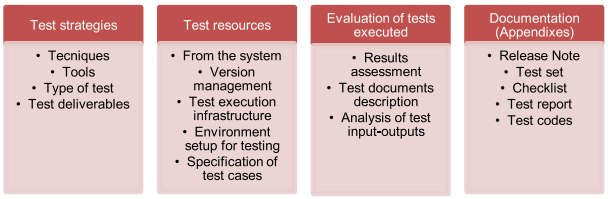

Fig. 4: Test plan overview

3.3 Test plan activity cycle

The system on which tests are performed must be structured, going through different activities to improve the system's quality (Hooda & Singh Chhillar, 2015). It is composed of test planning, design, configuration, execution, and evaluation. Within these activities, artifacts are generated that allow supporting their performance. Fig. 5 displays the cycle of activities of the plan and the artifacts involved in each activity, which generally occurs when this phase is approached.

Fig. 4: Testing plan lifecycle

Above, some artifacts may result from a specific stage of the process, such as test case requirements (Garousi, Felderer, Karapıçak, et al., 2018). Others like test case input, expected outputs, test code, test documentation (Garousi, Felderer, Karapiçak, et al., 2018), test scripts, set-up, clear-up procedures, files,

databases, environment, software, among other tools used in the testing process (Leotta et al., 2018).

3.4 Test plan proposal

The test plan is structured based on the previously presented three-layer architecture for IoT projects. Considering the layers of perception, network, and application, for each one, the types of tests are suggested, considering the IoT project characteristic.

3.4.1 Perception layer test

This layer includes sensors, actuators, microcontrollers, and low-level software-operated hardware(Dias et al., 2018)(Jacob & Mani, 2018) (Table II). In the domain of the perception layer, it refers to embedded systems. These are usually validated and verified following a V model (Chandra-Mani & Kumar-Agarwal, 2019).

Table 2. Tests recommended based on architecture layer

Layered architecture and system.	Recommended Tests
Application layer	• Unit testing applied to software modules. • Gray box testing. • GUI testing.
Network layer	• Unit testing applied to software modules. • Interoperability testing. • Semantic testing. • Conformance testing. • Network impact testing.
Perception layer	• Unit testing applied to software modules. • Embedded Integration testing. • Model-based testing.
System	• End-to-end testing. • Elasticity and scalability testing. • Security testing. • Stress testing. • Performance testing. • Acceptance testing.

The V-model (Fig. 6) sequentially follows different testing levels from the unit to the system. Thus, there are unitary (Dias et al., 2018), integration (Garousi, Felderer, Karapıçak, et al., 2018), system (Garousi, Felderer, Karapıçak, et al., 2018), and approval tests (Qian & Zheng, 2009).

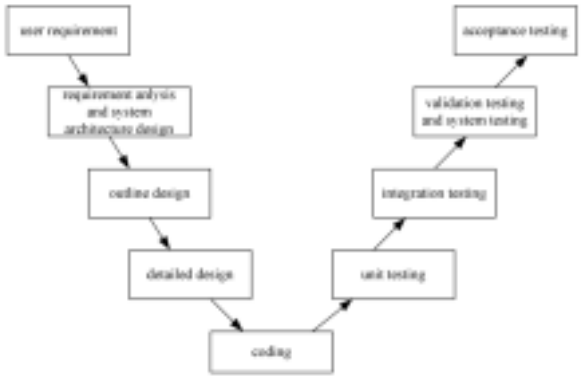

Fig. 6: Software testing V model [37]

Unit testing evaluates the smallest unit of software design and its modules to find defects. Integration tests occur after unit tests; these seek to assess probable defects between data and messages transmitted between software modules. Both types of tests are intended to verify the program design (Qian & Zheng, 2009).

System tests are used to verify system design and quality compliance, operation, and performance characteristics. It is usually performed in a real environment, where, after specifying test cases, they cover all the functionality and performance that the system must meet (Qian & Zheng, 2009).

Acceptance tests seek to validate whether the software satisfies user requirements (Qian & Zheng, 2009). This type of black-box testing seeks to effectively evaluate product quality, usually testing the system as a whole (Leotta et al., 2018). Other types of tests relevant to IoT projects can be model-based tests and reliability tests.

Model-based tests are widely used techniques in embedded systems. It has been a great point of attention from research and industrial environments (Garousi, Felderer, Karapıçak, et al., 2018). These consist of test case generation, starting from the requirements, using a data model, and avoiding testers defining the test types to be supported manually. The

advantages achieved are reflected by reaching test coverage, with a small number of tests (Blackburn et al., 2006).

Reliability tests assess the condition that the system performs expected functionality under specific settings over a period. A hardware-level approach evaluates the element's reliability periodically or at random intervals, identifying failure rates(Jacob & Mani, 2018).

3.4.2 Network layer tests

This layer gathers those hardware devices, antennas, and gateways that allow data transmission from sensor and actuator devices to the application services (Jacob & Mani, 2018)(Table II). It also considers the technical aspects and characteristics of the used communication protocols. A particular approach is required to ensure quality attributes concerning interoperability (Kim et al., 2018)(Jacob & Mani, 2018).

One level of communication protocol is used for interoperability and conformance testing. Interoperability tests seek to ensure whether a product that implements a standard is interoperable with other products executing the same specification, usually performed by different manufacturers. Conversely, conformance tests determine how the implementation of a particular standard is under the same standard (Kim et al., 2018).

Network impact tests can be used to analyze the quantitative and qualitative performance of an IoT solution. Under different restrictions applied to the network, the aim is to test the network's performance under different topology parameters, network area and size, and node proximity (Jacob & Mani, 2018).

Semantic tests pursue to validate the accuracy of the ontology used in IoT application data. Specifically, it seeks to confirm that meaning can be extracted from the data collected based on an ontology used. Besides, it searches for interoperability mechanisms between different standards, based on their ontology, to allow the storage and extraction of meanings from data with other structures, ensuring that the stored data is correct.

It is necessary to use tools in this area that validates the messages received from IoT devices, confirming message structure to guarantee its validity according to the used ontology. Once the data is verified, it is entered into the database (Kim et al., 2018).

There is a lack of tools and approaches for security testing in IoT systems (Dias et al., 2018). Especially on authentication, authorization, and secure data transmission (Zhang et al., 2014). Consequently, it is vital to test how secure the network is when adding nodes, modifying permissions, and encrypting data transmitted from nodes to remote destinations.

3.4.3 Application layer tests

In this layer, elements that are part of cloud services, databases, web solutions, and mobile devices are commonly contemplated (Jacob & Mani, 2018) in which the end-user usually interacts (Table II). One way to perform mixed tests is through gray-box tests. They contain the most relevant black- and white-box test aspects; these include the characteristic system behavior and its functionality. The above ensures that the product is built with the appropriate functionality and follows defined guidelines and conventions. Regression tests can also be performed at this stage in conjunction with integration tests contemplated (Jacob & Mani, 2018).

Testing graphical user interfaces (Graphical User Interface Test - GUI Test) guarantees the user's interface is friendly and free of ambiguity. The application consistently uses colors, icons, and other graphical interface designs contemplated (Jacob & Mani, 2018).

End-user application testing occurs when the application final version is already available and integrated with the IoT system. This feature can be considered as a system test, which interacts with all layers of the IoT system. Performed by the target user (stakeholders), it tests the system's functionalities and determines if the IoT system is relevant contemplated (Jacob & Mani, 2018).

It is needed to identify a testing approach when using cloud services to ensure compliance with elasticity, scalability, and security. According to the demand behavior, elasticity and scalability tests evaluate cloud provider characteristics when resource provisioning is satisfied elastically. Vertical and horizontal scalability must be put to the test. The first step evaluates whether, with more powerful resources, the growing demand is satisfied. The second one assesses if adding more resources, the performance improves. A load balancer should be tested to show if it behaves appropriately according to demand (Nachiyappan & Justus, 2015).

Different scenarios are required for the elasticity aspect according to the system uses, which evaluate how the system's elasticity changes and is coupled to demand (Nachiyappan & Justus, 2015)(Iyer et al., 2013). Cloud services must evaluate resource security from a cloud provider, considering the vulnerabilities they might be exposed to. However, such vulnerabilities are not uniquely associated with cloud providers and web applications. If the system employs role-based access, this should also be tested (Nachiyappan & Justus, 2015).

Using device behavior monitoring, network statistics, and similar metrics is a useful tool for detecting anomalies related to network security (Kanstrén et al., 2018). This instrument becomes essential, considering the attacks on devices and networks, usually reflected in their manipulation to send corrupt data and potentially affect stored information(Desnitsky & Kotenko, 2016).

Stress tests are useful tools to ensure that the platform is reliable and performs effectively, even when exceeding normal conditions. It is usually performed by generating high loads on the system using simulators. The aim is to maintain the system's stability in the face of these abnormal conditions of excessive stress (Nachiyappan & Justus, 2015).

3.4.4 Experts judgment evaluation

The expert judgment technique was used to validate the tests proposed in the layers of the three-tier architecture (Cabero-Almenara & Llorente Cejudo, 2013).

Through the validation of expert judgment, it was possible to show in table IV that the proposal of the tests established for the three-tier architecture has a high relevance (HL).

Table 3. Tests recommended based on methodology and architecture layers.

Layered architecture and system.	Recommended Tests	E1	E2	E3	E4	E5
Application layer	Unit testing applied to software modules.	HL	HL	HL	HL	HL
	Gray box testing.	HL	HL	HL	ML	HL
	GUI testing.	HL	HL	HL	HL	HL
Network layer	Unit testing applied to software modules.	HL	HL	HL	HL	HL
	Interoperability testing.	HL	HL	HL	HL	HL
	Semantic testing.	HL	HL	HL	HL	HL
	Conformance testing.	HL	HL	HL	HL	HL
	Network impact testing.	HL	HL	HL	HL	HL
Perception layer	Unit testing applied to software modules.	HL	HL	HL	HL	HL
	Embedded Integration testing.	HL	HL	HL	HL	HL
	Model-based testing.	HL	HL	HL	HL	HL
System	End-to-end testing.	HL	HL	HL	HL	HL
	Elasticity and scalability testing.	HL	HL	HL	HL	HL
	Security testing.	HL	HL	HL	HL	HL
	Stress testing.	HL	HL	HL	HL	HL
	Performance testing.	HL	HL	HL	HL	HL
	Acceptance testing.	HL	HL	HL	HL	HL

4 Discussion

According to the results obtained in phase 1, adopting a mixed methodology that includes Waterfall and Scrum models is suggested. Phase 2 presents the test plan's general structure and its constituent components, taking into account strategies and resources for the test plan, evaluating developed tests, and artifacts documentation resulting from the previous steps.

Phase 3 presents the activities to be carried out in a test plan from the software life cycle. For each activity, the artifacts involved are identified.

As the main contribution in phase 4, the types of tests to be carried out are identified considering the layers of the IoT project architecture, according to the approach addressed in the mixed methodology (Waterfalls-Scrum) and the selected three-layer architecture.

The above allows, depending on the phase and the product components, to select of the types of plans tests suggested (Table IV). This proposal was positively validated by the experts who participated in the assessment of this proposal.

4.1 Sprint - Development, unit tests

The tests follow the guidelines of the agile Scrum methodology. Usually, during the sprint execution, the method proceeded with the development and definition of unit tests covering the system's new versions (Table IV). These ensure that each new component is tested according to its expected performance.

The above involves grouping functionality to operate a sensor or actuator or carry out a specific microcontroller process. Therefore, each new software module must be accompanied by its respective unit tests, guaranteeing that the implemented behavior or functionality produces the appropriate outputs considering different possible inputs. It would be presented at the perception level through modifications to the firmware that runs on an embedded processor. However, the ease of implementing unit tests relies on embedded code with specific hardware. Therefore, the code intended to operate specific hardware must be tested with the physical component running.

Unit tests must be reflected by new firmware modifications in components such as gateways and improvements to intermediary services or middleware at the network layer. Similarly, it is necessary to implement unit tests for each new software module developed at the application layer level. These new modules reflect functionality addition or the refinement of an existing one.

Table 4. Tests recommended based on methodology and architecture layers.

Mixed methodology (Waterfall and Scrum)	Layered architecture	Recommended tests
Sprint – Development	Application layer	• Unit testing applied to software modules
	Network layer	
	Perception layer	
Integration testing	Application layer	• Gray-box testing • GUI testing
	Network layer	• Interoperability testing • Semantic testing • Conformance testing • Network impact testing
	Perception layer	• Embedded Integration testing • Model-based testing
Acceptance and system testing	Application layer	• End-to-end testing • Elasticity and scalability testing • Security testing • Stress testing • Performance testing • Acceptance testing
	Network layer	
	Perception layer	

4.2 Integration test

During the integration testing stage, the proposed test plan allows further evaluation of various aspects of the layers and interactions between internal components (Table IV). In the perception layer, integration tests can be performed with the embedded software modules, ideally by individually testing the previously evaluated modules. Otherwise, model-based tests help derive test cases (test case derivation), identify those to be performed on the system, and achieve test coverage (test coverage).

At the network layer level, several tests are identified that can be used. The relevance of the test and selection will depend on the communication protocol or protocols and the standards selection. Therefore, in environments where standards are used, W3C (World Wide Web Consortium) - SSN (Semantic Sensor Networks) W3C-SSN, Smart Applications Reference (SAREF), Standards for M2M and the Internet of Things like oneM2M (Kim et al., 2018). Regarding IoT devices, compliance testing shows a vital resource to ensure compliance with the standard used specifications. Interoperability tests are required if the system is used or planned to interact with devices from different manufacturers.

Validation mechanisms must be used when specific standards are used for messages structure received from devices. Starting from a base ontology defined by this same standard, the received message semantics and syntax are subjected to validation, and if everything is correct, the information is stored.

Network impact tests present an essential resource to evaluate the network protocol's performance and characteristics. The specific hardware for its transmission might comprise communication modules, antennas, and gateways, among others. Thus, this type of test aims to evaluate communication performance regarding network size, network area or extension, information reception, network topology, among others.

At the application layer level, gray-box tests, usually following integration practice, facilitate the integration change process in conjunction with graphical user interface (GUI) (Jacob & Mani, 2018). The above step ensures that the usability condition for the end-user is not affected.

4.3 System testing and acceptance

The system and acceptance tests are carried out to guarantee that each IoT system component works appropriately. Both software products and systems (hardware and software) are what the customer initially expected. Given the above, different tests can be used in the base architecture layers (Table IV).

End-to-end testing is a helpful tool to verify that the set of interconnected components or subsystems works correctly (Tsai et al., 2001). The test takes place from the end-user perspective; thus, it evaluates available functionalities that can be effectively executed, by the end-users, through all required components (Jacob & Mani, 2018). In this case, the assembled components' functioning is carried out according to the implemented IoT architecture (perception, network, and application).

Scalability tests ensure the system can work correctly according to the implemented vertical or horizontal scalability mechanisms. The objective is to satisfy the IoT product demand needs of end-users or managed devices. Elasticity is a distinctive aspect when using cloud technologies. If this technology supports the IoT solution, it must evaluate the provider resource behavior according to the demand. Similar to scalability tests, it guarantees that the elasticity mechanisms satisfy the current need by requiring or freeing up operating resources.

Stress tests are essential to perform on each layer of the system. These tests help to detect faults in each component. At the application level, they are used to knowing the system's behavior in the face of many requests from the end-user. They validate network stability and intermediate services at the communication level, especially when facing many messages sent by devices. Finally, at the perception layer level, it is relevant to analyze the system's behavior considering the continuous use of sensors,

actuators, and hardware resources in IoT devices for a long time.

Security testing is essential in IoT systems. The application layer can be addressed using conventional strategies, reviewing programming practices, vulnerabilities to SQL injection, or Cross-site scripting (XSS). At the network layer, it is relevant to establish and verify data encryption mechanisms for data transmission. It also includes the authentication verification and authorization characteristics of nodes belonging to the network, whether provided by the protocol or middleware services.

Complementary to the above, the security monitoring implemented at the application level, which evaluates the devices (perception layer) and the network (network layer), provides a tool to ensure its integrity and data. Considering various attacks that may occur, anomaly detection techniques and network statistics can be implemented on monitored data to identify the network's potential malicious usage (Kanstrén et al., 2018). The previous requires a prior definition of rules that allow identifying this behavior; therefore, it is crucial to consider aspects such as sensor readings history, system business rules, and physical positioning of devices, among others (Desnitsky & Kotenko, 2016).

Test cases should consider performance tests. The objective is to evaluate aspects such as latency, bandwidth, packet loss, and user concurrency, among other factors that impact the application's performance. This procedure can go side by side with stress tests (Jacob & Mani, 2018).

Similar to end-to-end testing, acceptance testing is an essential tool for validating the proposed solution in IoT projects. Besides, it allows testing product or services functionalities, causing interaction between all system components. It may also be convenient to complement acceptance tests with a real scenario deployment. They provide a final environment to validate the proposed solution, given a real scenario, communication, hardware, network, and other aspects of the product (Leotta et al., 2018).

5 Conclusion

A study was carried out regarding suitable development methodologies for IoT projects and test plants that can be employed according to components usually found in this type of system. This study initially recommends the Waterfall and Scrum models for developing IoT projects due to their versatility to support rigorous requirements analysis and quality assurance processes. These characteristics are complemented by developing a new version of products in short iterations. The above is the main reason why this methodology is recommended for this type of project. Additionally, a reference architecture used in IoT projects was identified. The architecture comprises three layers: (i) perception, (ii) network, and (iii) application. Besides, this study recommends the appropriate types of tests to evaluate each component mentioned above, given their nature.

Consequently, two factors were identified to influence test plan selection, considering IoT projects' architecture layers. The first is the methodological phase in which the project is located, and the second involves evaluated components.

Expert judgment was a technique that allowed verifying whether the proposed test plan complied with a set of criteria, standards, and requirements related to practice tests for IoT projects supported by their experience.

Therefore, this work's main contribution is the test plan proposal, given the two previous criteria. Finally, the results obtained can support implementing rigorous quality assurance processes by developing a specific test plan for IoT projects. It is recommended for future works to identify possible criteria to consider from a project management perspective. In this case, variables such as infrastructure costs and tool acquisition for test execution can be further considered. These can include team familiarity with implementing testing techniques, planning (over time), analyzing, and executing selected tests.

References

Afif, A. N., Noviyanto, F., Sunardi, Akbar, S. A., & Aribowo, E. (2020). Integrated application for automatic schedule-based distribution and monitoring of irrigation by applying the waterfall model process. *Bulletin of Electrical Engineering and Informatics*, *9*(1), 420–426. https://doi.org/10.11591/eei.v9i1.1368

Al Asif, M. R., Momin, M. A., Hossain, M. A., Roy, S., &

Mahfuz, N. (2019). Prototype Implementation of Edge Encryption in IoT Architecture. *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*, 1–7. https://doi.org/10.1109/ICCCNT45670.2019.8944810

Blackburn, M. R., Busser, R. D., Nauman, A. M., & Morgan, T. R. (2006). Model-based testing in practice. *INFORMATIK 2006 - Informatik Fur Menschen, Beitrage Der 36. Jahrestagung Der Gesellschaft Fur Informatik e.V. (GI)*, *2*, 197–204. https://doi.org/10.1145/302405.302640

Bures, M., Cerny, T., & Ahmed, B. S. (2019). Internet of things: Current challenges in the quality assurance and testing methods. *Lecture Notes in Electrical Engineering*, *514*, 625–634. https://doi.org/10.1007/978-981-13-1056-0_61

Cabarcas, A., Arrieta, C., Cermeno, D., Leal, H., Mendoza, R., & Rosales, C. (2019). Irrigation system for precision agriculture supported in the measurement of environmental variables. *Proceedings - 2019 7th International Engineering, Sciences and Technology Conference, IESTEC 2019*, *March 2020*, 671–676. https://doi.org/10.1109/IESTEC46403.2019.00125

Cabero-Almenara, J., & Llorente Cejudo, M. (2013). La aplicación del juicio de experto como técnica de evaluación de las tecnologías de la información y comunicación (TIC). *Eduweb*, *7*(2), 11–22.

Chandra- Mani, A., & Kumar-Agarwal, A. (2019). Improvement in Validation Process of Embedded Software using Regression Testing. *2nd INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND SOFTWARE ENGINEERING (ICACSE-2019)*, 644–649. https://doi.org/10.2139/ssrn.3351062

Ching, P. M., & Mutuc, J. E. (2018). Modeling the Dynamics of an Agile Scrum Team in the Development of a Single Software Project. *IEEE International Conference on Industrial Engineering and Engineering Management*, 386–390. https://doi.org/10.1109/IEEM.2018.8607430

Desnitsky, V., & Kotenko, I. (2016). Automated design, verification and testing of secure systems with embedded devices based on elicitation of expert knowledge. *Journal of Ambient Intelligence and Humanized Computing*, *7*(5), 705–719. https://doi.org/10.1007/s12652-016-0371-6

Dias, J. P., Couto, F., Paiva, A. C. R., & Ferreira, H. S. (2018). A brief overview of existing tools for testing the internet-of-things. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, 104–109. https://doi.org/10.1109/ICSTW.2018.00035

Dody-Lesmana, I.-P., Nurul-Karimah, R., & Widiawan, B. (2016). Agile-Waterfall Hybrid for Prevention Information System of Dengue Viral Infections: A Case Study in Health Department of Jember, East Java, Indonesia. *2016 Fourteenth International Conference on ICT and Knowledge Engineering*, 1–6.

Esteves Maria, R., Rodrigues Junior, L. A., Guarino De Vasconcelos, L. E., Mancilha Pinto, A. F., Takachi Tsoucamoto, P., Angelim Silva, H. N., Lastori, A., Marques Da Cunha, A., & Vieira Dias, L. A. (2015). Applying Scrum in an Interdisciplinary Project Using Big Data, Internet of Things, and Credit Cards. *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*, 67–72. https://doi.org/10.1109/ITNG.2015.17

Garousi, V., Felderer, M., Karapiçak, Ç. M., & Yilmaz, U. (2018). What We Know about Testing Embedded Software. *IEEE Software*, *35*(4), 62–69. https://doi.org/10.1109/MS.2018.2801541

Garousi, V., Felderer, M., Karapıçak, Ç. M., & Yılmaz, U. (2018). Testing embedded software: A survey of the literature. *Information and Software Technology*, *104*, 14–45. https://doi.org/10.1016/j.infsof.2018.06.016

Hayata, T., & Han, J. (2011). A hybrid model for IT project with Scrum. *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*, 285–290. https://doi.org/10.1109/SOLI.2011.5986572

Hooda, I., & Singh Chhillar, R. (2015). Software Test Process, Testing Types and Techniques. *International Journal of Computer Applications*, *111*(13), 10–14. https://doi.org/10.5120/19597-1433

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and agile methods. *Proceedings - International Computer Software and Applications Conference*, *1*, 520–525. https://doi.org/10.1109/cmpsac.2004.1342889

Iyer, G. N., Pasimuthu, J., & Loganathan, R. (2013). PCTF: An integrated, extensible cloud test framework for testing cloud platforms and applications. *Proceedings of the International Symposium on the Physical and Failure Analysis of Integrated Circuits, IPFA*, 135–138. https://doi.org/10.1109/QSIC.2013.65

Jacob, P. M., & Mani, P. (2018). A reference model for testing internet of things based applications. *Journal of Engineering Science and Technology*, *13*(8), 2504–2519.

Jacobson, I., Spence, I., & Ng, P.-W. (2017). Is there a single

method for the internet of things? *Communications of the ACM*, *60*(11), 46–53. https://doi.org/10.1145/3106637

Kanstrén, T., Mäkelä, J., & Karhula, P. (2018). Architectures and experiences in testing IoT communications. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, 98–103. https://doi.org/10.1109/ICSTW.2018.00034

Kim, H., Ahmad, A., Hwang, J., Baqa, H., Le Gall, F., Reina Ortega, M. A., & Song, J. (2018). IoT-TaaS: Towards a Prospective IoT Testing Framework. *IEEE Access*, *6*(c), 15480–15493. https://doi.org/10.1109/ACCESS.2018.2802489

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E., & Prause, C. R. (2017). Hybrid software and system development in practice: Waterfall, scrum, and beyond. *ACM International Conference Proceeding Series*, 30–39. https://doi.org/10.1145/3084100.3084104

Leach, R. (2016). *Introduction to software engineering* (R. LeBlanc (ed.); Second Edi). CRC Press Taylor & Francis Group. https://content.taylorfrancis.com/books/download?dac=C2014-0-34027-5&isbn=9781315371665&format=googlePreviewPdf

Leotta, M., Ricca, F., Clerissi, D., Ancona, D., Delzanno, G., Ribaudo, M., & Franceschini, L. (2018). Towards an Acceptance Testing Approach for Internet of Things Systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10544 LNCS*, 125–138. https://doi.org/10.1007/978-3-319-74433-9_11

Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I. A. T., Siddiqa, A., & Yaqoob, I. (2017). Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. *IEEE Access*, *5*, 5247–5261. https://doi.org/10.1109/ACCESS.2017.2689040

Monroy, M. E., Arciniegas, J. L., & Rodríguez, J. C. (2017). Caracterización de los Contextos de Uso de la Ingeniería Inversa. *Informacion Tecnologica*, *28*(4), 75–84. https://doi.org/10.4067/S0718-07642017000400010

Muccini, H., & Moghaddam, M. T. (2018). IoT architectural styles: A systematic mapping study. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11048 LNCS*, 68–85. https://doi.org/10.1007/978-3-030-00761-4_5

Nachiyappan, S., & Justus, S. (2015). Cloud testing tools and its challenges: A comparative study. *Procedia Computer Science*, *50*, 482–489. https://doi.org/10.1016/j.procs.2015.04.018

Qian, H. M., & Zheng, C. (2009). A embedded software testing process model. *Proceedings - 2009 International Conference on Computational Intelligence and Software Engineering, CiSE 2009*. https://doi.org/10.1109/CISE.2009.5366362

Quiroga Montoya, E. A., Jaramillo Colorado, S. F., Campo Muñoz, W. Y., & Chanchí Golondrino, G. E. (2017). Propuesta de una Arquitectura para Agricultura de Precisión Soportada en IoT. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *24*, 39–56. https://doi.org/10.17013/risti.24.39-56

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13. https://doi.org/10.1145/1764810.1764814

Sachdeva, V., & Chung, L. (2017). Handling non-functional requirements for big data and IOT projects in Scrum. *Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering*, 216–221. https://doi.org/10.1109/CONFLUENCE.2017.7943152

Schwaber, K. (1997). SCRUM Development Process. In *Business Object Design and Implementation* (Issue February 1986, pp. 117–134). https://doi.org/10.1007/978-1-4471-0947-1_11

Shah, W. M., Arif, F., Shahrin, A. A., & Hassan, A. (2018). The implementation of an IoT-based Flood Alert System. *International Journal of Advanced Computer Science and Applications*, *9*(11), 620–623. https://doi.org/10.14569/ijacsa.2018.091187

Singhto, W. (2016). Adopting a Combination of Scrum and Waterfall Methodologies in Developing Tailor-made SaaS Products for Thai Service and Manufacturing SMEs. *2016 International Computer Science and Engineering Conference (ICSEC)*, 1–6.

Singhto, W., & Denwattana, N. (2016). An experience in blending the traditional and Agile methodologies to assist in a small software development project. *2016 13th International Joint Conference on Computer Science and Software Engineering, JCSSE 2016*, 9–13. https://doi.org/10.1109/JCSSE.2016.7748914

Tsai, W. T., Bai, X., Paul, R., Shao, W., & Agarwal, V. (2001). End-to-end integration testing design. *Proceedings - IEEE Computer Society's International Computer Software and Applications Conference*, 166–171. https://doi.org/10.1109/CMPSAC.2001.960613

Vogel, B., Peterson, B., & Emruli, B. (2019). Prototyping for internet of things with web technologies: A Case on Project-Based Learning using Scrum. *2019 IEEE 43rd Annual Computer Software and Applications Conference*, 300–305. https://doi.org/10.1109/COMPSAC.2019.10223

Wardana, A. A., Rakhmatsyah, A., Minarno, A. E., & Anbiya, D. R. (2019). Internet of Things Platform for Manage Multiple Message Queuing Telemetry Transport Broker Server. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, *4*(3), 197–206. https://doi.org/10.22219/kinetik.v4i3.841

Yaqoob, I., Ahmed, E., Hashem, I. A. T., Ahmed, A. I. A., Gani, A., Imran, M., & Guizani, M. (2017). Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. *IEEE Wireless Communications*, *24*(3), 10–16. https://doi.org/10.1109/MWC.2017.1600421

Zhang, Z. K., Cho, M. C. Y., Wang, C. W., Hsu, C. W., Chen, C. K., & Shieh, S. (2014). IoT security: Ongoing challenges and research opportunities. *Proceedings - IEEE 7th International Conference on Service-Oriented Computing and Applications, SOCA 2014*, 230–234. https://doi.org/10.1109/SOCA.2014.58

Zheng, R., Zhang, T., Liu, Z., & Wang, H. (2016). An EIoT system designed for ecological and environmental management of the Xianghe Segment of China's Grand Canal. *International Journal of Sustainable Development and World Ecology*, *23*(4), 372–380. https://doi.org/10.1080/13504509.2015.1124470