Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INTERNET-OF-THINGS: A SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

MAI ALFAWAIR

AlBalqa Applied University

Mai.alfauri@bau.edu.jo

ABSTRACT

The Internet-of-Things (IoT) is a revolutionary technology that connects everyday objects to the Internet. The utilization of IoT systems created integrated systems that transformed communication, administration, and monitoring in various fields such as smart homes, healthcare monitoring, and even environmental monitoring, giving users more control over everyday objects and adding to the complexity of such systems. The successful development of IoT systems requires software engineering to extend beyond the traditional development strategies or System Development Lifecycle (SDLC), particularly owing to the heterogeneity and complexity of such systems. This paper provides an innovation by expanding the traditional System Development Lifecycle (SDLC) to accommodate the nature of IoT systems, making the traditional SDLC more convenient for use to develop IoT systems. This has been achieved by editing some of the already existing SDLC processes, as well as adding some other processes to the traditional SDLC. Such additions include aspects unique to IoT systems development, for example, things requirement, communication requirement, system component integration, and system operation, as well as other additions to the traditional SDLC. Results clearly demonstrate that IoT application development has become more systematic, efficient, and hence, requiring less time for development since the IoT SDLC has facilitated project management and improved the overall quality of the process.

Keywords: *Internet-Of-Things (IOT), System Development Lifecycle (SDLC), Network Development, Software Engineering*

1. INTRODUCTION

Readily available high-speed Internet connections and the presence of everyday objects such as mobile phones, cars and other objects that can connect to the Internet have given such objects more value in light of their ability to facilitate our everyday lives. The utilization of everyday objects connected to a network to perform their specific tasks under greater user control, thereby offering increased convenience, led to the emergence of the IoT paradigm. IoT systems provide a platform whereby multiple objects can connect and communicate over a network under the umbrella of modern wireless telecommunications. Users nowadays can use their smart phones to access and control objects through applications to manage their everyday lives.

IoT systems, like any other IT system, consist of a combination of hardware and software. Smartphones, sensors attached to objects, and other standalone devices constituting the hardware component of such systems, which can be installed and maintained by the manufacturers. The relevant software used to operate such objects and devices, including mobile applications, require appropriate design, development and maintenance in order to provide effective overall functionality of IoT systems. IT research in the fields of IoT and System Development Life Cycle (SDLC) has allowed such technology to rise and flourish, providing a favorable environment for such technology to grow rapidly and successfully.

From a software engineering point of view, traditional SDLC is created for the purpose of application development for classical computer systems, which includes designing software that performs specific tasks intended for utilization by the end-user. On the other hand IoT systems have many stakeholders: such systems involve interconnecting heterogeneous devices that are owned and managed by many independent entities, making the development process challenging. Traditional SDLC is considered basic and generic for the development of adequate software that suits the requirements of IoT systems, and hence does

not fully support the development lifecycle of IoT systems. This paper provides an innovation by proposing an SDLC adapted for IoT systems, advancing some of the already existing inadequate SDLC processes, as well as adding some other processes to the traditional SDLC. Such additions consider aspects unique to IoT systems development, including 'things' requirement, communication requirement, system component integration, and system operation. The advantage of such innovation is the provision of a more systematic and efficient SDLC for IoT systems, facilitating project management, decreasing development time, and improving the overall quality of the development process.

This matter is particularly important because IoT systems require software that permits advanced levels of productivity, provided against a background of high-quality service. Additionally, when considering SDLC, IoT systems require advanced maintenance and evolution beyond installation and implementation as new objects may periodically be added or removed, and as user requirements may change frequently. The maintenance and evolution of IoT systems should also be facilitated by tools that save time and work, since IoT requires continuous growth. In [1], the author mentions this issue.

This paper proposes an SDLC for IoT systems, as the nature of such systems requires special considerations regarding development, implementation and subsequent maintenance. Basic SDLC is considered universal and generic, yet certain challenges arise when it is implemented for IoT systems. Basic SDLC is quite insufficient when developing a complex heterogeneous system IoT, with everyday objects constituting an integral part of such systems. This article expands the basic SDLC for any system, applying special development components to preexisting stages and adding new developmental steps to the lifecycle. This paper includes firstly a literature review on IoT systems and conventional SDLC, followed by a description of the defined SDLC for IoT systems, including a discussion of its characteristics and an explanation of its processes.

2. RELATED WORK

The International Telecommunications Union defines IoT as "enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [2]. In [3], the authors state that there are four essential characteristics in IoT terms for a device to be considered a 'thing'. These are:

1. The device must be able to collect and transmit data. IoT devices should collect information and either send it to another device in their environment or directly to the Internet.
2. The device must possess action-based responses. IoT devices should be developed to respond according to particular conditions.
3. The device must be capable of receiving information. IoT devices should be developed in a way to receive information from a network.
4. The device must be capable of communicating with others. IoT devices should be developed in a way to communicate with other devices on a network. [20]

The high impact of IoT systems on various aspects of users' everyday lives is considered their main strength. They are now integrated into many areas of life, such as the home environment, business management, healthcare, transportation, logistics and industry. This involvement has become so extensive that the US National Intelligence Council (NIC) expects that "by 2025 Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more". [4]

2.1. IoT Architecture

The basic architecture of any IoT system consists of three layers: [5][6] physical, network and application. An IoT reference model was released by the IoT World Forum (IWF) to serve as a common framework, accelerating IoT deployment, and encouraging the development of replicable models [7]. This reference model consists of seven layers, shown in Figure 1:

1- **Physical devices and controllers:** consists of all the devices and controllers that control IoT devices; the "things" in IoT systems.
2- **Connectivity:** comprises of communication and processing units, such as routers and firewalls, allowing all IoT devices to communicate with each other, and with application platforms, such as computers and smartphones.
3- **Edge computing:** responsible for converting network data, which is usually high-volume, into low-volume information suitable for storage and

higher-level processing. Filtering and processing of incoming data occurs at this layer, rendering data accessible to the next layers.

4- **Data accumulation:** different types of data in different formats coming from heterogenous IoT devices are stored on this layer.
5- **Data abstraction:** responsible for the aggregation and formatting of data so that it is accessible by applications more effectively.
6- **Application:** encompasses a wide range of applications that utilize IoT input data or control IoT devices, allowing for information interpretation to occur.
7- **Collaboration and processes:** different IoT applications run by users are able to communicate and collaborate to make IoT data more useful.

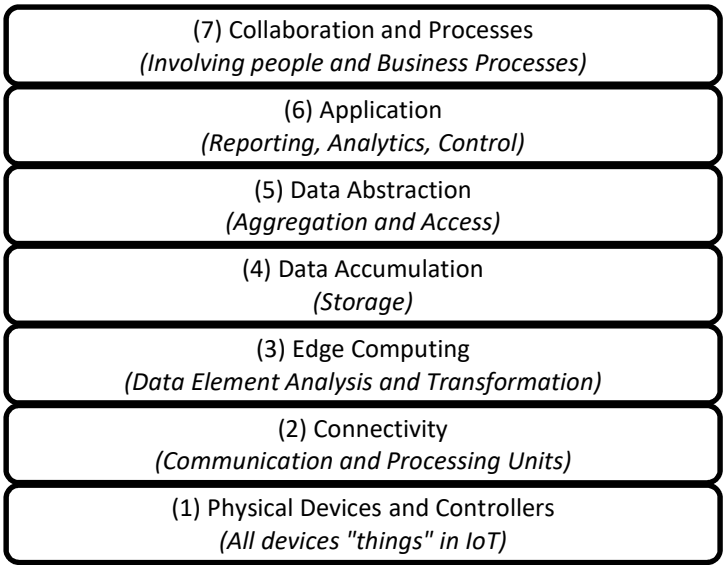

Figure 1: IoT World Forum (IWF) Reference Model [7].

2.2. IoT Challenges and Risks

Despite the several benefits IoT technology offers, and the praise it has gained, it faces many challenges and poses many risks to users [8][9][10]. These can be summarized as:

1. **Privacy:** since the IoT technology consists of large amounts of data concerning everyday objects, and potentially other private data such as users' locations, private property or even personal health records, users are concerned that the increasing amount of private data housed by the IoT may be accessed by others. Much work must be done in order to ensure these privacy concerns are met.
2. **Security:** this relates to privacy, and concerns the reliance of IoT systems on networks. Securing users' data from unauthorized access becomes a challenge. Many researchers are working to achieve high levels of security. Indeed, the NIC states that "to the extent that everyday objects become information security risks, the IoT could distribute those risks far more widely than the Internet has to date". [4][11]
3. **Compatibility:** devices from various manufacturers are connected by a network. Compatibility must thus be ensured to maintain the proper functioning of the entire system. Although this problem may be addressed if all the manufacturers agree on and implement a common standard, technical issues will nevertheless persist.
4. **Data integrity:** integrity means that the data is correct and is not modified. Data integrity is ensured by maintaining secure network connections by the use of many techniques such as end-to-end encryption. [12]
5. **Data confidentiality:** messages must be securely transmitted across the network, and must be protected from unauthorized access by the use of a variety of techniques including requiring identification in order

to permit actions to be carried out on the network.

6. **Data authentication:** this ensures that IoT data is sent from the correct sender to the specified receiver by correctly identifying every object on the network. There are many other challenges to IoT technology.

An important challenge stated in [1] is the fact that traditional software development techniques are considered generic when developing robust software suiting the requirements implied by IoT systems, which is why they do not fully support the development life cycle of IoT systems. This is particularly due to the fact that IoT systems require software that permits advanced levels of productivity, provided against a background of high-quality service. Additionally, when considering SDLC, [13] IoT systems require advanced maintenance and evolution beyond installation and implementation, as new objects may periodically be added or removed, and as user requirements may change frequently. Furthermore, the maintenance and evolution of IoT systems should be facilitated by tools that save time and effort, since IoT requires continuous growth. Yet Leau et al. offer no solution to this problem. The present paper proposes one [13].

2.3. Software Engineering Highly Decentralised Systems

Ian Sommerville, the father of software engineering, has defined it as "an engineering discipline that is concerned with all aspects of software production". [14] Software engineering is thus concerned with the organization of software development in a systematic way, which is sometimes called the software process, or the Software Development Life Cycle (SDLC). [13] It is a sequence of activities that result in the development of a software product. These activities include the following:

2.4. Software Specification

In this process, the customer and the engineer understand and define the services required from the software that is to be produced, and the constraints on its operation and development. The agreed requirements are specified in a document called the Software Requirement Specification (SRS). This process is also called software requirements, and is organized into the software requirement engineering process that comprises four stages as shown in Figure 2 [15]:

A. **Feasibility study:** an analysis to assess if the development of the software is financially effective and technically applicable. The feasibility study must be relatively cheap and quick.
B. **Requirement elicitation and analysis:** an analysis of the system's requirements in order to understand its specifications. This may involve developing system models and prototypes.
C. **Requirement specification:** involves translating the system's requirements into written documents (user and system requirements), in order to perform the next phase.
D. **Requirement validation:** involves checking system requirements for completeness and consistency. Errors in the system requirements document produced at the previous stage are inevitably discovered and modified accordingly.

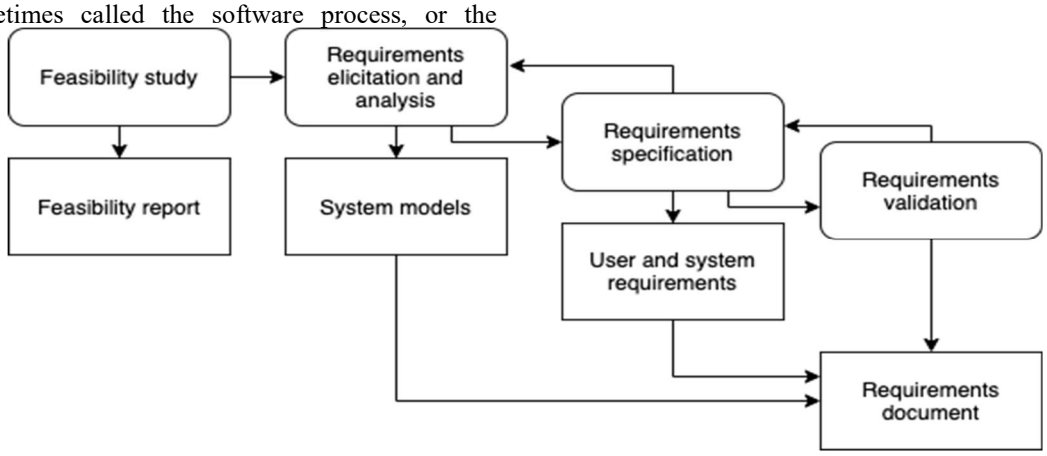

Figure 2: The Software requirement engineering process.

2.4.1. software development

This is the process of converting the specifications determined in the previous stage into an executable system. It consists of designing and programming the software. [16] In the design process, the software's structure is determined and a data model is constructed in order to produce a system architecture, database specification, and interface and component specifications. These are produced through the following activities:

A. **Architectural design:** in this phase, designers determine the overall structure of the system with its components by defining the system's modules and sub-systems with their distributions and relationships.
B. **Interface design:** this phase deals with designing interfaces for the system and its components.
C. **Component design:** in which each component of the system is designed to determine its functionality and operations, taking into account the reuse of existing components.
D. **Database design:** in this phase, the database is designed. This includes designing the structure of the system's data and its representation in the database, taking into account the reuse of existing databases.

2.4.2. software validation

This phase includes both validation and verification to confirm that the designed software meets the customer's predetermined specifications and expectations. [17] Both the process validation and the product are validated. Process validation involves checking each stage of the software development process in order to ensure that the development process is proceeding correctly and in a cost-effective manner. The main product validation technique used is program testing, in which the system is run using test data to visualize its functionality and detect defects, as well as to check it against the customer's requirements to ensure that it meets the customer's expectations. Any defect detected will undergo a debugging process whereby this defect is corrected. Repeated testing is carried out after the debugging process in order to visualize the fixed changes, and the entire system is again tested. If any further bugs are detected they are fixed, and the system is tested again. This cycle is repeated until no more bugs are found. The entire testing process consists of the following three testing levels: [18]

A. **Component testing:** every component of the system is tested separately and independently.
B. **System testing:** the components that have passed the component testing level are integrated and tested as a whole, in order to detect any defect that arises from component interaction and to find issues arising from component interfaces. The process's functional and non-functional requirements are also tested.
C. **Acceptance testing:** the system is tested using real data supplied by the customer in a process called acceptance testing to ensure that the system meets their requirements. This process ensures that the customer will accept the system for operational use.

2.4.3. software evolution

In this context, "evolution" refers to making any changes in the existing software. Modifications accommodate changing customer and market requirements. This stage is also considered as part of software maintenance. [19]

3. SDLC FOR IOT

This section will describe an SDLC that suits the technical nature and needs of the IoT technology. This SDLC is detailed in figure 3 and then explained:

Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

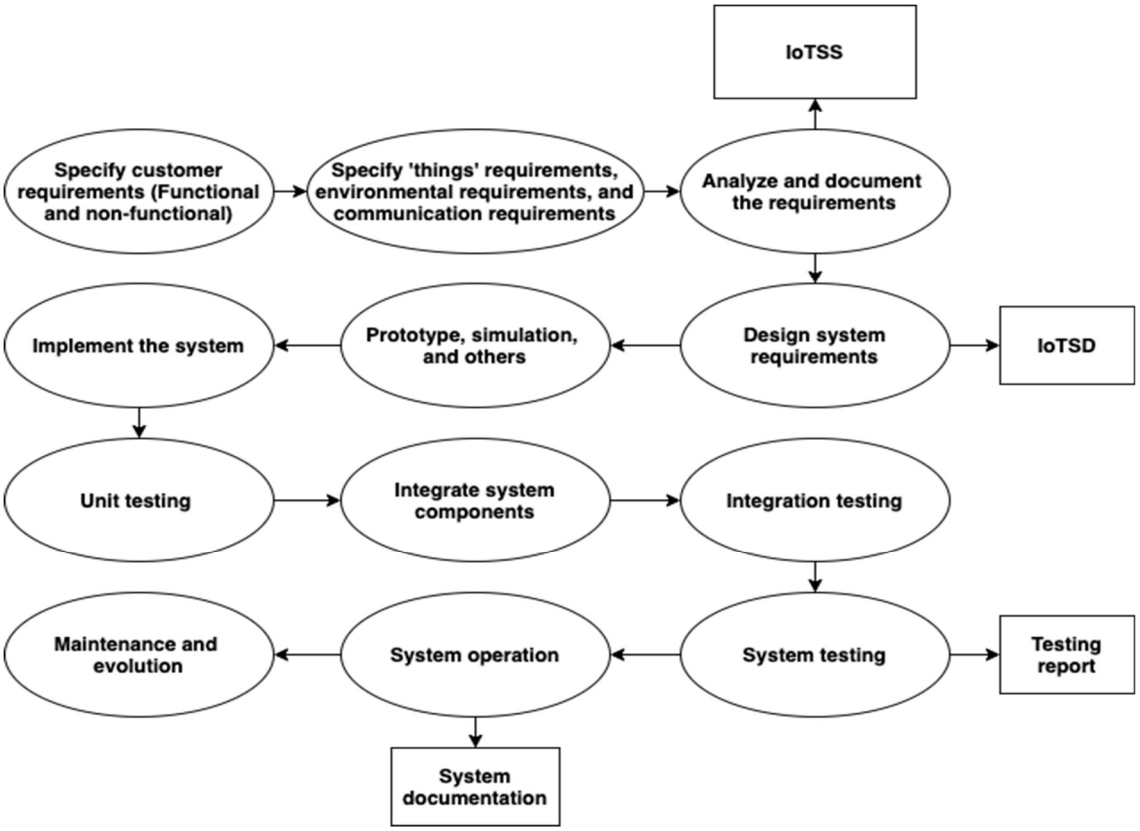

Figure 3: Detailed illustration of SDLC for IoT systems.

3.1. Specifications

The first step in the cycle is the determination of specifications. Such specifications are similar to the specifications in the conventional SDLC, yet the specifications of IoT systems are enhanced in this SDLC to suite the complex architecture and additional components of the IoT system.

System specifications are set mainly by the customer according to their requirements. Customers in this phase play an important role by describing their requirements and their expectations of the system. The specifications of any IoT system are nevertheless set not only by the customer, but are partly determined by the nature of the "things" the customer wants to handle and empower, in order to ensure compatibility between the different technical standards run on different objects. The specifications of such systems are also determined by the developers themselves, who also have their own requirements based on their customers' requirements, in order to develop the required fully-functional system. Developers therefore play some part in setting and enhancing the specifications set by the customer, since the developers integrate objects running on the various platforms that are the components of IoT systems. As developers are considered experts in developing these systems, they may further empower those systems depending on customer needs, sometimes thinking outside the box and showing their creativity in integrating real-life objects in user-controlled comprehensive systems without diverging from customers' needs and requirements. As a consequence, developers in IoT systems are actively involved in modifying and positively empowering customer specifications.

In this step, system engineers are responsible for documenting the system specifications collected from customers and system developers. The system engineer therefore studies the technical requirements of the system architecture with the components of the system (i.e., the "things") and the needs of system developers, all in relation to the specifications and needs of the customer. The system engineer then analyzes these requirements in order to clarify, organize, categorize and present them in a written document called in the present paper the Internet-of-Things System Specifications (IoTSS) document, equivalent to the Software Requirement Specification (SRS) document, in order to clarify all the requirements. This document, written by the system engineer, should

include the following categories of requirements types as defined by this paper:

1. **Functional requirements:** this type of requirement describes what the IoT system should do and the services that the developer and customer agree the system should provide.
2. **Non-functional requirements:** the system engineer in this type of requirement describes the relevant constraints on the development and operation of the IoT system in accordance with Quality-of-Service (QoS) metrics.
3. **"Things" requirements:** the requirements of each "thing" (i.e., object) in the IoT system that enable such objects to interact with other components of the system should be described in this type of requirement.
4. **Environmental requirements**: this describes the environment such as the power supply and networking requirements within which the IoT system will work.
5. **Communication requirements**: the communication that should occur between the IoT system components (the "things"), the messages, the data, its track of travel from and to the different kinds of components, where it is saved, and how it is retrieved is outlined in this type of requirements.

After these requirements are collected by the system engineer, they are drafted in an initial document that is reviewed by the developer and the customer as well as the system engineer, creating a final draft that is called the IoTSS, which serves as the contract between the client and the supplier.

3.2. Service Level Agreement

The previously described IoTSS includes a section called the Service Level Agreement (SLA). The SLA determines several aspects of the service provided and includes basic data such as the title and ID of the SLA, the type of application, and the start and end dates. Individuals and groups involved with the project and the QoS metrics used for the IoT system are also determined in the SLA. The SLA must be standardized so that it is understood by all the stakeholders involved in the development of the IoT system. The level of service a customer can expect from a provider is set also in the SLA. The developmental activities the developer should follow are also specified in the SLA. The authors in [21] propose a specialized model for SLA specification in IoT applications, such that this proposed SLA is standardized and understood by all the stakeholders.

3.3. Design

In software engineering, a pattern refers to any recurring problem with a corresponding solution specific to it. Such patterns occur on three levels: architectural and design patterns and idioms. Architectural patterns describe a system's structural organization and its division into subsystems, each with specific functions and responsibilities. Design patterns describe system objects, components and subsystems with their interactions. Idioms are the lowest-level pattern describing the way to implement certain aspects of system components through the use of programming languages and their features.

In our SDLC for IoT, when the system engineer has published the IoTSS, the designers of the IoT system translate this document into an appropriate design. The designers will firstly determine the objects to be used in the IoT system, which are regarded as the system's building blocks. The IoT system's architecture, which includes the communication paths between the different IoT system components (i.e. the communication interfaces, gateways and patterns is then designed. The different sources and destinations of data are also determined at this stage, with data integration schemes also being outlined. The exact placement of sensors on each of the IoT system components is determined in relation to data collection and communication. The designer is responsible at this stage for creating design sketches and publishing them in a design document called in the present paper the Internet of Things System Design (IoTSD), which will be used later by the developers. The designer may also create a prototype illustrating the system components and how they interact. A simulation may also be created, showing how the IoT system would operate and how the data will travel between the various components of the IoT system.

3.4. Implementation, Integration, and System Operation

The implementation process, in which developers translate the design into an executable system, directly follows the design process. In the

basic SDLC, implementation refers to programming the required software. Software is implemented either by developing and programming a new system adapted to the customer's specific demands or reusing pre-existing components to develop the required system.

In this SDLC for IoT, the essentials of software implementation are similar to those in the traditional implementation in SDLC, but with distinct differences that require discussion. The traditional implementation process in basic SDLC is concerned with software development. The task of the developer in IoT system implementation is, however, to work on real-life objects, developing software to manage and control such objects in order to organize and connect them in a network. Developers gather the required system objects, the 'things'; either such objects are suitable for use in an IoT environment, or they are modified to fit as required by adding special functionality components such as sensors to enable objects to perform a specific function. Objects can even be manufactured to perform particular functions. Software is developed according to customer requirements and the anticipated functionality of the system, all in accord with the objects' requirements. Comprehensive software components are developed for specific objects and devices in IoT systems to enable them to operate and interact with other components in the IoT environment. Software development also includes development of a Graphical User Interface (GUI) to allow user communication with the system and control on its various components. Developers at this stage test each system component in isolation to ensure that components perform their intended functions, after which they connect all system components into a network, allowing effective object-object as well as user-object communication. In this process, developers must decide on an IoT communication protocol in order to settle the connection issues between objects in the process of communication and exchanging data.

In order to integrate system components with developed software within a network, the concept of integration has also been applied to the implementation process in the SDLC for IoT systems. This is done to ensure smooth operation of the IoT system, with all its heterogeneous components. For this specific reason, an operation process has been added to implementation in the SDLC for IoT.

During this operation process the integrated system is settled in its intended location and connected to a network to initiate its operation. An initial test is performed while operating the system, before thoroughly testing the system as a whole at the next stage.

3.5. IoT System Testing and Validation

This phase is indeed conducted throughout the development process, yet it is emphasized after the system has been developed. Starting from the specification phase, testing is carried out in the context of customers and developers verifying the required specifications with those written in the IoTSS. In the design phase, the system engineer verifies that the design document (IoTSD) is well matched with the system specifications (IoTSS). After implementation has taken place, the system's operation is tested to check its functionality. The whole system is then tested against customer specifications to check that it meets customer requirements.

In basic SDLC the testing phase comprises component, system and acceptance testing. In the testing phase of SDLC for IoT, the same elements are tested, but to a higher level to accommodate the nature of IoT systems. In the component testing performed by the system developer, each system object (or 'thing') is tested individually, as are the software components, to check that they meet the intended requirements. When all system components have passed these tests, the next level of system testing is performed. Here, the integrated components of the system are tested as a whole for functionality and operation. This level of testing is thus also called operation testing, and is carried out by the testing group that includes system developers, system engineers, software engineers, computer engineers and a network engineer. This testing group tests the entire system including system objects, software and network connections. After passing this level of testing, acceptance testing must be performed. Here, the customer joins the testing group to check that the system operates according to their requirements. The system is configured and operated at the customer site, allowing the customer to test the system on-site. Since this is the third level of testing, the system has already been thoroughly tested in terms of functionality, so the purpose of this level is for the customer to 'accept' the system

– hence the name "acceptance testing". After the testing phase is complete, the testing team documents the entire procedure in the so-called testing report. This report contains all the information regarding in this phase such as the cases, results and the discovered bugs and defects revealed in the testing phase. The testing report is used to fix the bugs that have been revealed. Once the bugs have been fixed, it is used to re-test the system, acting as evidence of system testing throughout the testing phase, as well as being used for maintenance and evolution in later phases.

3.6. Maintenance and Evolution

Once the IoT system has been fully implemented, it is operated at the customer site. After operation, any change or modification to the system required by the customer calls for maintenance and/or evolution. The required modification is identified and implemented, and the system is subsequently tested. Documentation of any modification also occurs at this stage.

3.7. Documentation

All the documents created throughout the system development phases are collected and kept in an archive. These documents are retrieved when needed, such as during system maintenance and evolution.

4. EVALUATION AND RESULTS

The IOTSDLC needs to be evaluated in terms of applicability and specificity to IoT systems, as well as the efficiency of IoT system development when IOTSDLC is applied. This is essential in order to ensure the applicability and efficiency of IOTSDLC when used to develop IoT systems. To carry out the evaluation, three projects were developed using the IOTSDLC under our full supervision to test the results: A smart parking system, and two smart home systems. The developers for our three projects have previously developed smart home and parking systems with similar system requirements using the IWF reference model described previously, which is considered one of the state-of-art models for IoT system development. When previously developing these projects, direct programming and implementation were used, taking direct feedback from customers, and making changes accordingly if possible. This conventional method enforced rework, and is effort and time-consuming. When developing the three systems using the IOTSDLC, the development scheme was strictly followed and each process in development was documented. For each project, a development plan was proposed in advance for the entire project in accordance with the IOTSDLC. During development, each phase of the plan was followed accordingly, sometimes modified to adapt real circumstances. The time and the people involved with each relevant process were monitored and recorded carefully for comparison purposes.

The completion of the three projects using the IOTSDLC on time proves that the IOTSDLC is applicable for the development of IoT systems, denoting the clarity of its phases, and the simplicity it yields for the development of IoT systems. Management of the development of IoT systems became more outlined, and hence, easier to perform.

After development completion, feedback was collected from all stakeholders taking part in the three projects, and the results were analyzed. The results show that all stakeholders agree that IOTSDLC development process is fully applicable with IoT systems and exhibits the following results.

4.1. Development Time

The results were analyzed and compared with those from similar projects who have not used the IOTSDLC. The result demonstrated using the IOTSDLC is the decreased development time. When comparing the development time of similar projects, projects following the IOTSDLC took less time than the projects that have not, with the difference in time being more apparent in larger projects requiring more time, i.e. IOTSDLC shows higher time efficiency for larger projects requiring more time as shown in Figure 4.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

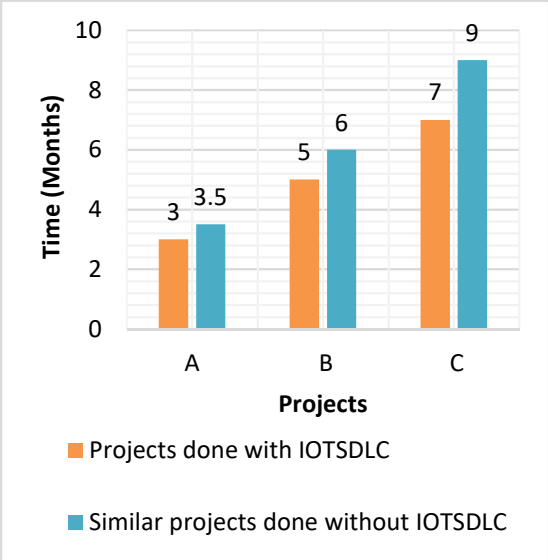

Figure 4: Comparison of the time required to complete groups of similar projects done with and without IOTSDLC.

4.2. Productivity

During development of each project, the time and the people involved with each relevant process were monitored carefully for comparison purposes. The organized structure of the IOTSDLC allows IoT system development to be smoothly completed, avoiding rework, and thus increasing productivity. As each project has been developed after the other using the IOTSDLC, it has been noticed that the productivity has been enhancing. This is partially due to natural increase in productivity over time due to repeated development by developers, and more importantly this is attributed to the organized structure of IOTSDLC, allowing system development to be smoothly completed, and avoiding rework.

4.3. Quality

The criteria of analysis of the results includes the time of development of IoT systems using IOTSDLC in comparison with conventional development processes, as well as the applicability of system development using IOTSDLC implied through the productivity of system developers. Results clearly demonstrate that IoT application development has taken less time for completion, and hence has become more systematic and efficient, since the IOTSDLC has facilitated project management and improved the overall quality of the process.

Stakeholders also confirmed that application development became more systematic than not using this approach. As a result, more projects of similar requirements could be accomplished using the same time period when implementing IOTSDLC. Hence, we can confidently state that following the IOTSDLC has made application development more systematic.

Project managers of the three projects noticed that the IOTSDLC has facilitated project development not just in terms of completion, but also in terms of management. A tailored SDLC has made the development process clear, easy to follow, and allowing project planning to be easily performed. As a result, IOTSDLC has taken into account certain development aspects not previously tackled, improving the overall quality of the development process.

Since the IOTSDLC requires full documentation for every process, whenever maintenance or evolution of the system is needed, these processes will also run smoothly and more effectively, saving effort, and requiring less investigation time. This is another circumstance where the carefully created IOTSDLC will save more effort and time, even for future development and improvement.

5. CONCLUSION

IoT system development is indeed a complex process, taking into account the development and integration of various software, hardware, and networking requirements. The overall basic scheme of system development, in fact, shares many similarities with the basic model of system development: The System Development Life Cycle (SDLC). The development process tailored for IoT systems, the IOTSDLC, has been presented in this paper, and deals with such systems in the same manner as for traditional systems, yet the nature of IoT systems requires special development considerations due to the complexity of the system and the variability of its components.

The IOTSDLC tackles the defects found in all phases of traditional SDLC when developing IoT systems. IOTSDLC further specifies system specifications, allowing system engineers to specify additional 'object' requirements determined by the specific objects that will constitute the IoT system, and how such objects will influence data communication and integration, taking into

account the complexity of the system architecture that is to be fully specified in the design phase, and including the concomitant usage of multiple communication interfaces with gateways and sensors.

The development of a system prototype also distinguishes the SDLC for IoT systems, allowing system engineers and customers to assess their expectations of the system and seeing how the system components will integrate and operate. Implementation and testing the components of the IoT SDLC are also taken to a further level by component testing, which involves testing each object on its own and then, after integration, holistically testing the entire system. The overall implementation of such development process, the IOTSDLC, has shown to facilitate overall system development, avoiding rework, increasing productivity of systems development, and outlining a clear map for specifications and integration of IoT system components, allowing facilitated project development, and enhanced project management.

REFERENCES

[1] Sarhan, Qusay Idrees. "Internet of things: a survey of challenges and issues." *International Journal of Internet of Things and Cyber-Assurance* 1.1 (2018): 40-75.

[2] Wortmann, Felix, and Kristina Flüchter. "Internet of things." *Business & Information Systems Engineering* 57.3 (2015): 221-224.

[3] Palma Genicio, Daniel, et al. "An Internet of things example: classrooms access control over near field communication." (2014).

[4] National Intelligence Council, Disruptive Civil Technologies – Six Technologies with Potential Impacts on US Interests Out to 2025, Conference Report CR 2008-07, http://www.dni.gov/ 1666nic/NIC_home.html .

[5] Madakam, Somayya, R. Ramaswamy, and Siddharth Tripathi. "Internet of Things (IoT): A literature review." *Journal of Computer and Communications* 3.05 (2015): 164.

[6] Yun, Miao, and Bu Yuxin. "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid." *2010 International Conference on Advances in Energy Engineering*. IEEE, 2010.

[7] Stalling, William. "The Internet of Things: Network and Security Architecture," Internet Protoc. J., vol. 18, no. 4, pp. 2–24, 2015.

[8] Khan, Rafiullah, et al. "Future internet: the internet of things architecture, possible applications and key challenges." *2012 10th international conference on frontiers of information technology*. IEEE, 2012.

[9] Lee, In, and Kyoochun Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." *Business Horizons* 58.4 (2015): 431-440.

[10] Hwang, Yong Ho. "Iot security & privacy: threats and challenges." *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*. ACM, 2015.

[11] Jing, Qi, et al. "Security of the Internet of Things: perspectives and challenges." *Wireless Networks* 20.8 (2014): 2481-2501.

[12] De Cremer, David, Bang Nguyen, and Lyndon Simkin. "The integrity challenge of the Internet-of-Things (IoT): on understanding its dark side." *Journal of Marketing Management* 33.1-2 (2017): 145-158.

[13] Leau, Yu Beng, et al. "Software development life cycle AGILE vs traditional approaches." *International Conference on Information and Network Technology*. Vol. 37. No. 1. 2012.

[14] Sommerville, Ian. "Software engineering 9th Edition." *ISBN-10*137035152 (2011).

[15] Kotonya, Gerald, and Ian Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[16] Schwaber, Ken, and Mike Beedle. *Agile software development with Scrum*. Vol. 1. Upper Saddle River: Prentice Hall, 2002.

[17] Rakitin, Steven R. *Software verification and validation for practitioners and managers*. Artech House, Inc., 2001.

[18] Naik, Kshirasagar, and Priyadarshi Tripathy. *Software testing and quality assurance: theory and practice*. John Wiley & Sons, 2011.

[19] Bennett, Keith H., and Václav T. Rajlich. "Software maintenance and evolution: a roadmap." *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000.

[20] Shabbir, Ghulam, et al. "Network performance enhancement of multi-sink enabled low power lossy networks in SDN based Internet of Things." *International Journal of Parallel Programming* (2018): 1-32.

[21] Alqahtani, Awatif, et al. "End-to-end service level agreement specification for iot applications." *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2018.