Requirements for an IoT middleware with Speech recognition interfaces

Johana A. Manrique[1], Jesús M. T. Portocarrero[2]

[1] Universidad Autónoma de Bucaramanga, Centro de Excelencia y Apropiación en Internet de las Cosa – CEA IoT,
Bucaramanga, Colombia

[2] Nuance Communications Inc
Rio de Janeiro, Brasil

jmanrique4@unab.edu.co, jesus.talavera@nuance.com

Abstract. Internet develops as a new paradigm known as Internet of things where people and daily things are connecting to the Internet. In the virtual world, things need digital interfaces to facilitate communication between human-machine. However, since IoT is a complex paradigm, the development of these applications becomes a challenging task.
IoT is influencing how we live, but the interaction in a natural way between human-machine is still far from being non-intrusive. To achieve this concern, it is necessary to make use of basic human capabilities such as voice, which occurs naturally but in the IoT, do not still widely used it.
Hence, this paper proposes the basic functional and non-functional requirements for SWITCH a middleware with research potential for hiding the complexity in the development of IoT applications, in terms of (i) IoT reference architectures, (ii) middleware for IoT, and (iii) speech recognition systems for IoT.

Keywords. Internet of Things, Middleware, Speech Recognition.

1 Introduction

IoT focuses on connecting everyday things, providing to people digital interfaces for making easy human-machine interaction [1], [2]. According to Borgia [3], IoT involves communication with anything (between devices, applications, people), at any time and place, through any network.

Currently, the proliferation of devices connected to the Internet is bigger in relation to the number of people in the world. CISCO estimated by 2030, more than 500 billion devices would be connected to the Internet, an average of 8 devices per person [4].

With IoT, daily things can be equipped with devices such as sensors, allowing them to generate and exchange data with minimal human intervention, transforming many aspects of how we live: personal, social, health, logistics, industrial, and others, knowing as applications domains [5]. Each domain has specific requirements as required by the application. The development of an IoT application faces a varied list of difficulties, as known: (i) There is no standard configuration to develop IoT applications (horizontal and vertical approach) [6], [7], (ii) The IoT ecosystem is composed of several heterogeneous technologies (hardware and software) [2]. Some of these technologies have decades of existence. Other new techs have been created under the requirements that IoT attends [8], and (iii) According to Patel and Cassou [9] the development of an IoT application involves several stakeholders work, who guarantee a more structured and better defined process in accordance with all the layers of an IoT reference architecture.

To avoid overloading costs in terms of financial, personnel and development time resources, it is important the implementation of platforms that facilitate the development of IoT applications and the administration of the generic requirements for the different domains [3], [10], [11]. An IoT platform defines as the middleware and infrastructure that allows end users to interact with smart things [2].

The development of this type of platform will be one of the fastest growing tech market segments in the coming years [12].

As the number of devices to connect things to the Internet increases, the number of applications available to handle them increases as well. Daily things do not have interfaces for interacting with humans in a non-intrusive way. Some applications use digital prints for identify attributes in the net, but a better method for natural interaction can be the human capabilities such as voice or movements and that is why the research of this area continues growing up [18], [19].

Considering the relevance of IoT in several domains, this paper proposes the basic functional and non-functional requirements for a Smart middleWare for Iot with speeCH recognition – SWITCH, a middleware with research potential for hiding the complexity in the development of IoT applications with voice-based interfaces. The paper as organized as follows: (i) generic functional and non-functional requirements for IoT middleware (Section II), (ii) Specific requirements for SWITCH (Section III), and (iii) finally, recommending future research directions.

2 Requirements analysis for a generic IoT middleware

The middleware offers common services for the agile development of applications that support the heterogeneity between the communication and computing devices, and the interoperability within the diverse applications and services that are executed in those devices [20]–[22]. For understanding the requirements that this type of IoT platform must address, we conducted a literature review in three items described below.

2.1 IoT reference architectures

At the architectural level, a middleware must provide an Application Programming Interface - API to support the developers, programming abstraction, interoperability, context-aware, adaptability, self-governed, distributed services.

Fig 1 presents the categories on the IoT reference architectures cited in the literature review. The IoT-A project is an architecture with more open access documentation for developers. In addition, it is one of the most cited in the literature. The proposal has a layered architectural design approach, as follows. Application, virtual entity, IoT service and device derived from the main abstractions identified in the domain model. The layer communication supports the large number of devices connection. The requirements expressed by the stakeholders regarding the possibility of creating services and applications in the IoT application layer are covered by IoT process management and service organization. To address the expressed concern about trust, security and privacy in the IoT that supports for this layer. Finally, the management layer is required to manage the interaction between the functional groups of this architecture.

In general, a RA establish a common understanding of the IoT paradigm with a set of components that identify its main concepts, relationships and limitations. Also, facilitates the development of systems for IoT. The generic structure of the architectures is comprised in layers/levels that start from the device (hardware) to the application (software). Generally, they have

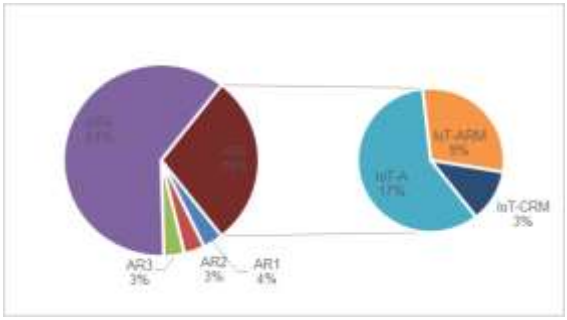

Fig 1. IoT Reference Architectures RA more cited in the literature

- RA1: ITU-T reference model [13].
- RA2: Industrial Internet Reference Architecture (IIRA) [14], [15].
- RA3: Reference Architecture Model Industry (RAMI 4.0) [16].
- RA4: Others IoT reference architecture proposed [10].
- RA5: IoT-A Project [17] divided in IoT-A: Internet of Things Architecture, IoT-ARM: Architecture Reference Model and IoT-CRM: Communication Reference Model

two cross layers: security and data management, network or devices. These can be used in the other components of the horizontal structure of the RA according to the specific needs of each one. They share relevant non-functional requirements for IoT as horizontality (support for different application domains), heterogeneity, scalability, connectivity, identity management, device management, communication management and security.

2.2 Middleware for IoT

Even though the IoT concept is relatively new, the idea of monitoring and controlling devices through computers and networks has existed for decades, and that is why IoT is often confused with other technologies. Some middleware in the literature described as IoT middleware, they were also described as WSN middleware, a few years ago [23].

Based on the literature review, there are different types of middleware platform developed for IoT categorized according to their design approach [2], [22], [24], [25]. We describe below the most approaches cited in the literature.

Event-based middleware is viable when an application has mobility and common failures. Events run from the components of the sending application (producers) to the components of the receiving application (consumers). An event is a significant change of state [26], [27]. This approach has advantages such as the use of patterns for publication and subscription (asynchronous), scalability, real-time processing with minimum delay. However, some of them do not provide interoperability, adaptability and context awareness.

Service-oriented middleware builds software in a service form. It is based on Service Oriented Architecture (SOA), which provides abstractions for the underlying hardware through a set of services that applications need. The services can be designed, implemented and integrated into a framework taking into account the incorporation of a service provider (hosting services), a consumer of services (represents any application) and a record of services (actions) to offer an environment flexible and easy for the development of applications [28], [29]. This approach has advantages such as the discovery of services, composition of services and reuse of services. However, code management is not easy to develop.

Agent-based middleware, where applications are divided into modular programs to facilitate the injection and distribution of the network, using mobile agents. When migrating from one node to another, the agents maintain their execution status. This facilitates the design of decentralized systems capable of tolerating partial failures [30]. This approach has advantages are resource management (network load reduction and network latency reduction), code management (asynchronous and autonomous execution and protocol encapsulation), availability, reliability (robustness and fault tolerance) and adaptability. However, its greatest disadvantage is the low interoperability among resources.

Cloud-based middleware focuses on the provision of hosting services through the Internet. It limits users to the type and quantity of IoT devices they can implement, but allows them to connect, collect and interpret data, since possible use cases can be determined and programmed a priori [31]. In this approach, the security management depends on the cloud service provider.

Actor-based middleware configures applications in a plug-and-play form. It exposes a variety of IoT devices as reusable actors to distribute them in the network in such a way that each of them can perform computational calculations. They do not have a particular standard for communication between IoT devices, which facilitates interoperability and scalability. However, it has as a disadvantage the low security of the system.

Fig 2. Generic architecture for IoT middleware

Some middleware use hybrid approach to have a better performance than those belonging to a single approach [22]. Taking into account all the requirements that the different types of approaches support, Fig 2 shows a generic architecture for IoT middleware.

2.3 Speech recognition systems for IoT

Speech is one of the most important communication methods that human beings have. Currently, human-computer interaction using voice without resorting to the implementation of interfaces such as keyboards or pointing devices is possible thanks to the Automatic Speech Recognition - ASR, which represents the function of modulating a voice signal to a series of words (phonemes) with the help of algorithms made by a computer program [32].

There are four types of ASR system [33], [34], as follows: (i) Isolated word recognition requires only one statement at a time. It is ideal for situations when the user must give answers or commands of a single word, but it is not natural for the entries of several words. (ii) Connected word recognition needs a minimum pause between word utterances to allow effortless voice flow. Its functionality is very similar to isolated words. (iii) Continuous speech recognition allows users to speak more or less naturally, while the computer decides the content. Recognizers with continuous speech skills are more difficult to generate since they employ machine-learning techniques to decide on emission limits. (iv) Spontaneous Speech Recognition allows recognizing spontaneous speech as that which occurs in interviews, debates, dialogues. The ASR system with the ability to speak spontaneously must handle a variety of natural speech characteristics.

Fig 3 shows a generic ASR architecture system [35], [36]. The pre-processing starts with Features extraction. During this step, the voice signal (that is, a set of acoustic waves) is transformed into a sequence of pre-phonetic symbols without linguistic meaning but containing characteristics with eigenvalues.

Acoustic modeling compares symbols with specific phonetic waveforms. In this step, it represents the audio signals by discriminating the classes of basic speech units and taking into account the variability of speech with respect to the speakers, channel, and environment. For this, a training speech data is required, which focuses on improving the model in aspects such as (i) Speech acoustics requires the recording of several speakers; (ii) Language requires a text corpus or sentence grammar and; (iii) Recognition lexicon is a list of recognizable tokens with one or more phonetic transcriptions.

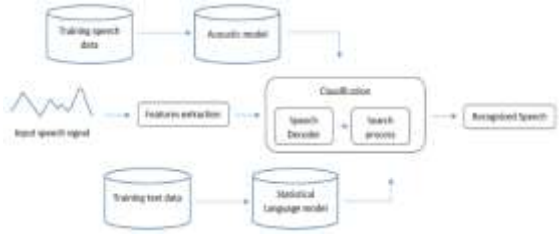

Fig 3. ASR Architecture

Language modeling imposes restrictions on the recognition hypotheses generated to model the structure, syntax, and semantics of the target language. Statistical language models are based on the empirical fact that a good estimate of the probability of a lexical unit can be obtained by observing it in amount text data.

2.4 Functional requirements for a generic IoT middleware

Functional requirements are functionalities that the system must provide, on how it should react to particular inputs and how it should behave in specific situations [37]. IoT has main requirements to satisfy the needs of the different application domains [22], [24], as follows.

Resource discovery: Every device must announce its presence and the resources it offers automatically. Discovery mechanisms also need to scale well, and there should be efficient distribution of discovery load, given the IoT's composition of resource-constrained devices.

Resource management: Facilitating potentially spontaneous resource re-composition, to satisfy application needs. The resource usage should be monitored, allocated or provisioned in a fair manner.

Data management: Providing data management services to applications. This data is

represented in different formats and various models. The data process has stage as acquisition, processing, filtering, compression, classification, aggregation, storage.

Event management: Transforming simple observed events into meaningful events. Managing real-time analysis of high-velocity data so that downstream applications are driven by accurate, real-time information, and intelligence.

Code management is necessary when the application requests it. However, it is not mandatory.

2.5 Non-functional requirements for a generic IoT middleware

Describe how the software will do its functionalities, based on quality attributes [38]. In this case, the quality of the software product is the degree to which the middleware satisfies the requirements of its users, thus contributing into values. These requirements are establish in the quality model of the International Organization for Standardization 25010 [39]. This model is divided in eight groups, as follows.

Functional suitability: A middleware provides functions that meet explicit and implied needs when used under specified conditions with completeness, correctness and appropriateness functionality.

Performance efficiency: Represents the performance relative to the amount of resources used under stated conditions. This performance is related with the time behavior of the middleware.

Compatibility: A middleware can exchange information with other products, systems or components, and perform its required functions, while sharing the same hardware or software environment (interoperability).

Usability: A middleware can achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context. This performance is given to the user by an easy-of-deployment feature.

Reliability: Every component or service in a middleware needs to be reliable to achieve overall reliability and fault tolerance, which includes communication, data, technologies, and devices from all layers.

Security: In IoT middleware, security needs to be considered in all the functional and nonfunctional blocks including the user level application. The availability property means the system is robust enough to be able to operate in adverse situations.

Maintainability: Represents the degree of effectiveness and efficiency with the middleware can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.

Portability: Degree of effectiveness and efficiency with the middleware can be transferred from one hardware, software or other operational or usage environment to another. It is necessary that the code management address this process.

Other non-functional requirements as scalability, context-aware, persistence, monitoring, real time, autonomous, stream processing are not explicitly described in the ISO. The literature mentioned them because they are specifically for an IoT middleware design.

3 Requirements for SWITCH

This section specifies the functional and non-functional requirements for SWITCH. Based on the requirements presented in section 2, we proposed a list of functional requirements shows in Table 1.

Any stakeholder in the IoT process (Domain expert, Software architect, Software development, network manager, database manager, Software administrator) can take these requirements for the IoT middleware development considering introduce into voice-based interfaces from design.

We considered SWITCH as service-oriented middleware with a layer architectural style. Each layer provides a set of services to the previous layer and uses the services of the next layer [40]. The design addresses some non-functional requirements, as known (i) Scalability where the number of components increases without affecting the upper layer. They are independent of each other. (ii) Security, since having isolated layers, when one of them has failures at the security level, does not imply that the others must be committed in the same way; (iii) Maintainability, since the fact of making corrections of software failures (bug), or simple maintenance tasks in one layer, does not imply that the upper layers must be re-implanted.

Table 1. Functional requirements for SWITCH

ID	Requirements
R1	The middleware should allow end users to configure IoT applications with speech recognition system
R2	The middleware must allow the "listen" status each time the instruction is given
R3	The middleware should provide mechanisms to transform the voice signal into phonetic symbols
R4	The middleware should compare the phonetic symbols with a recognition lexicon
R5	The middleware should compare the phonetic symbols with the text corpus
R6	The middleware must transform the voice signal into a command that is understood by the IoT applications
R7	The middleware must categorize the entities obtained from the transcription speech to text
R8	The middleware must transform simple events in significant events for all the system
R9	The middleware could allow the storage of algorithms for the features extraction of the voice signal
R10	The middleware should allow the storage of the voice command
R11	The middleware should allow the storage of the text corpus
R12	The middleware should allow the storage of a recognition lexicon
R13	The middleware should provide mechanisms to configure an ASR platform
R14	The middleware should show the user the voice command in text format
R15	The middleware should provide mechanisms to analyze the network connectivity
R16	The middleware must offer network connectivity control functions
R17	The middleware should storage of the voice command in a text format
R18	The middleware should provide mechanisms to configure of the an IoT application requirements
R19	The middleware must manage the mobility of the data in the system
R20	The middleware must show the user the recognized entities from the text categorization
R21	The middleware allows creating IoT applications
R22	The middleware should resource discovery automatically
R23	The middleware should facilitate the display of collected data at any time
R24	The middleware must be monitoring, resolve conflicts and fairly assign services because all the applications must have an acceptable quality of service.
R25	The middleware allows collecting data from sensors
R26	The middleware should facilitate the implementation of code for the development of IoT applications
R27	The middleware should allow the user to categorize (training) entities

4 Conclusions

A middleware is a software tool that hides complexity in the development of IoT applications and helps in the communication process between application and device layers.

SWITCH is a proposal that once implemented, contributes to the solution of the problems identified in the literature, as follows: (i) easy-deployment IoT applications with voice-based interfaces; (ii) speech recognition system for interacting human-machine in a non-intrusive way; (iii) provide digital interfaces to the devices

Acknowledgements

We thank all partners within the Center of Excellence and Appropriation on the Internet of Things (CEA-IoT), as well the Colombian Ministry for the Information and Communication Technologies (MinTIC), and the Colombian Administrative Department of Science, Technology and Innovation (Colciencias) through the project ID: FP44842- 502-2015 from the National Trust for Funding Science, Technology and Innovation Francisco José de Caldas.

References

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Comput. networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, “A gap analysis of Internet-of-Things platforms,” *Comput. Commun.*, vol. 89, pp. 5–16, 2016.

[3] E. Borgia, “The Internet of Things vision: Key features, applications and open issues,” *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[4] CISCO, “Internet of Things at a Glance,” 2016.

[5] Internet Society, “The Internet of Things (IoT): An Overview,” Geneva, Switzerland, 2015.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[7] F. Wortmann, K. Flüchter, and others, “Internet of things,” *Bus. Inf. Syst. Eng.*, vol. 57, no. 3, pp. 221–224, 2015.

[8] I. Lee and K. Lee, “The Internet of Things (IoT): Applications, investments, and challenges for enterprises,” *Bus. Horiz.*, vol. 58, no. 4, pp. 431–440, 2015.

[9] P. Patel and D. Cassou, “Enabling high-level application development for the Internet of Things,” *J. Syst. Softw.*, vol. 103, pp. 62–84, 2015.

[10] M. R. Abdmeziem, D. Tandjaoui, and I. Romdhani, “Architecting the internet of things: state of the art,” in *Robots and Sensor Clouds*, Springer, 2016, pp. 55–75.

[11] L. Atzori, A. Iera, and G. Morabito, “Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm,” *Ad Hoc Networks*, vol. 56, no. 1, pp. 122–140, 2017.

[12] IoT Analytics, “IoT Platforms: Market Report 2015-2021,” Hamburg, Germany, 2016.

[13] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, and D. Boyle, “IoT Architecture – State of the Art,” in *From Machine-To-Machine to the Internet of Things*, Elsevier, 2014, pp. 145–165.

[14] M. Weyrich and C. Ebert, “Reference architectures for the internet of things,” *IEEE Softw.*, vol. 33, no. 1, pp. 112–116, 2016.

[15] G. Banda, K. Chaitanya, and H. Mohan, “An IoT protocol and framework for OEMs to make IoT-enabled devices forward compatible,” in *Signal-Image Technology & Internet-Based Systems (SITIS), 2015 11th International Conference on*, 2015, pp. 824–832.

[16] T. Usländer and U. Epple, “Reference model of industrie 4.0 service architectures,” *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 858–866, 2015.

[17] IoT-A Project, “Requirements — IOT-A: Internet of Things Architecture,” *Requirements — IOT-A: Internet of Things Architecture*, 2016. .

[18] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and challenges for realising the Internet of Things*. 2010.

[19] N. Zhong, J. Ma, R. Huang, J. Liu, Y. Yao, Y. Zhang, and J. Chen, “Research challenges and perspectives on Wisdom Web of Things (W2T),” in *Wisdom Web of Things*, Springer, 2016, pp. 3–26.

[20] S. Hadim and N. Mohamed, “Middleware: Middleware challenges and approaches for wireless sensor networks,” *IEEE Distrib. Syst. online*, vol. 7, no. 3, p. 1, 2006.

[21] M.-M. Wang, J.-N. Cao, J. Li, and S. K. Dasi, “Middleware for wireless sensor networks: A survey,” *J. Comput. Sci. Technol.*, vol. 23, no. 3, pp. 305–326, 2008.

[22] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, “Middleware for internet of things: a survey,” *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, 2016.

[23] J. A. Manrique, J. S. Rueda-Rueda, and J. M. T. Portocarrero, “Contrasting Internet of Things and Wireless Sensor Network from a conceptual overview,” in *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*, 2016, pp. 252–257.

[24] S. A. Chelloug and M. A. El-Zawawy, “Middleware for Internet of Things: Survey and Challenges,” *Intell. Autom. Soft Comput.*, vol. 0, no. 0, pp. 1–9, 2017.

[25] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, “IoT middleware: A survey on issues and enabling technologies,” *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, 2017.

[26] R. Meier and V. Cahill, “Steam: Event-based middleware for wireless ad hoc networks,” in *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, 2002, pp. 639–644.

[27] R. Sanchez-Guerrero, F. Almenárez, D. Diaz-Sanchez, P. Arias, and A. Marin, “A model for dimensioning a secure event-driven health care system,” in *Wireless and Mobile Networking Conference (WMNC), 2012 5th Joint IFIP*, 2012, pp. 30–37.

[28] V. A. Immanuel and P. Raj, “Enabling context-awareness: A service oriented architecture implementation for a hospital use case,” in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, pp. 224–228.

[29] M. P. Papazoglou, “Service-oriented computing: Concepts, characteristics and directions,” in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, 2003, pp. 3–12.

[30] C. Perrot, G. Finnie, and I. Morrison, “Establishing context for software agents in pervasive healthcare systems,” in *2012 15th International Conference on Network-Based Information Systems*, 2012, pp. 447–452.

[31] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, L. Skorin-Kapov, and R. Herzog, “OpenIoT: Open Source Internet-of-Things in the Cloud,” in *Interoperability and Open-Source Solutions for*

the Internet of Things: International Workshop, FP7 OpenIoT Project, Held in Conjunction with SoftCOM 2014, Split, Croatia, September 18, 2014, Invited Papers, I. Podnar Žarko, K. Pripužić, and M. Serrano, Eds. Cham: Springer International Publishing, 2015, pp. 13–25.

[32] EY, "Internet of Things: Human machine interactions that unlock possibilities," United Kingdom, 2016.

[33] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Commun.*, vol. 56, pp. 85–100, 2014.

[34] A. H. Unnibhavi and D. S. Jangamshetti, "A survey of speech recognition on south Indian Languages," in *Signal Processing, Communication, Power and Embedded System (SCOPES), 2016 International Conference on*, 2016, pp. 1122–1126.

[35] H. Bouraoui, C. Jerad, A. Chattopadhyay, and N. Ben Hadj-Alouane, "Hardware Architectures for Embedded Speaker Recognition Applications: A Survey," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, p. 78, 2017.

[36] A. V. Haridas, R. Marimuthu, and V. G. Sivakumar, "A critical review and analysis on techniques of speech recognition: The road ahead," *Int. J. Knowledge-based Intell. Eng. Syst.*, vol. 22, no. 1, pp. 39–57, 2018.

[37] I. Sommerville, *Ingeniería del Software*. PEARSON, 2011.

[38] K. Adams, *Non-functional Requirements in Systems Analysis and Design*. Springer, 2015.

[39] International Organization for Standardization - ISO, *Software product quality*, vol. 1. 2011, p. 34.

[40] M. Richards, *Software architecture patterns*. O'Reilly Media, Incorporated, 2015.

Article received on 5/11/2018; accepted xx/xx/xxxx.
Corresponding author is jmanrique4@unab.edu.co