ISSN 2347 - 3983
Volume 9. No. 3, March 2021
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter14932021.pdf
https://doi.org/10.30534/ijeter/2021/14932021

Towards a Software Architecture for Internet of Things based System of Systems

Muhammad Waqar Aziz[1], Usama Musharaf[2], Ali Sayyed[3]
Department of Computer Science, CECOS University of IT & Emerging Sciences Peshawar, Pakistan,
[1]waqar@cecos.edu.pk, [2]usamamusharaf1992@gmail.com, [3]alisayyed@cecos.edu.pk

ABSTRACT

The Smart City vision is becoming a reality with the widespread adaption of Internet of Things (IoT). In this context, several architectural styles like service-oriented and microservice architecture have widely been used in the development of IoT-based systems. However, less amount of work is done for IoT-based system of systems. The recognition of system of systems (SoS) as a system with its unique features such as operationally and administratively independence has been considered a new trend of distributed software systems. The collaboration of the SoS independent system helps to build a larger and more complex system. The characteristics and domain constraints of SoS make some quality attributes critical, especially when SoS is based on IoT. These quality attributes should be considered while designing such systems. To fill this gap, this article presents a novel software architecture based on microservices architectural style, while considering the important quality attributes required for IoT-based SoS. The applicability of the proposed architecture is demonstrated through a smart city case study. In addition, the design quality is evaluated in terms of scalability and maintainability. The results show that the design developed using the proposed architecture is better in terms of these two quality attributes than the existing approach.

Key words : Internet of Things, micro-services, software architecture, system of systems, smart city.

1. INTRODUCTION

The entire world is moving towards the era of intelligent technology. The key factor in bringing this revolutionary change is Internet of Things (IoT) [1]–[3]. The conversion of real-world objects into an interactive and intelligent decision-making systems that can be controlled from remote sites is only possible due to IoT [1]. Furthermore, IoT has become an important technology that promises to provide an intelligent life for humans, by enabling communication between objects, machines, and everything that exists with people. IoT represents a system that has sensors, actuators, and other embedded processing nodes communicating with real-world elements through a wired or wireless network structures. With the advent of the Internet of Things concept, the Internet is becoming more conducive to intelligent living in all its aspects. One of the key requirements for IoT is to develop software to automate tasks [2]. As the systems are becoming more and more software-intensive, there is a need to focus on software design and architecture [4].

The term "System of Systems" (SoS) has been used for the last few decades to describe a large complex system as a system, with its independent constituent systems (ICs), which execute together to achieve a common goal. SoS and IoT have many common properties such as their ICs are heterogeneous, independent, and often distributed. In addition, IoT and SoS achieve their goals through extremely run time cooperation that unites them. The recent exposure of IoT-based system of systems (IoT-SoS) [5], [6] has further speed up the design and development of such large-scale distributed complex systems. However, special attention is required to design the software architecture of such systems. Software architecture is the high-level design of a software system and the discipline of creating and connecting components with respect to different stakeholder views [7]. Software architectures have been recognized as the pillar of the success of any software system. In addition, they are responsible for aggregating quality attributes, such as scalability, interoperability, reliability, maintainability, and so on.

Smart City (SC) is an example of IoT-based SoS [8], where many IoT systems work together to provide various services to citizens. For example, in the case of disaster management, several ICs such as police, firefighting, and rescue work together to deal with a disastrous situation. There are several design approaches for smart cities [9], [10], [11] but an approach is missing for smart cities as IoT-based SoS. To fill this gap, a software architecture for IoT-based SoS is presented in this paper. On the one hand, the proposed architecture allows the development of IoT-based SoS as it deals with the major principles of SoS. On the other hand, the proposed architecture also considers quality attributes, such as scalability, interoperability, maintainability, and performance as needed. The proposed architecture is based on microservices, also known as microservice architecture. Microservices is an architectural style that is basically a variation of a service-oriented system [12]. By combining microservices style with cloud computing improves achieving of quality attributes in an architecture. To evaluate the applicability of the proposed architecture, it has been applied

in the smart city case study. Then, the quality of the developed design is evaluated in terms of two quality attributes (i.e., scalability and maintainability). The results show that the proposed architecture, while covering the principles of SoS, outperforms the existing approach.

The rest of the paper is organized as follows: Section 2 provides the background knowledge required to understand the core concepts, followed by the research methodology in section 3 and related work in section 4. The proposed software architecture is presented in Section 5. Its application in the smart city case study is demonstrated in section 6, whereas the evaluation of design quality is provided in section 7. Section 8 concludes this paper with our future work.

2. PRELIMINARIES

2.1 System of Systems

The concept System of Systems (SoS) involves the run time collaboration of distributed and heterogeneous systems to achieve its goals. In general, IoT and SoS have many common properties. Some of the IoT-based SoS principles are managerial independence, operational independence, devices heterogeneity, geographical distribution of components, and emergent behavior [13]. For these principles quality attributes such as interoperability, scalability, maintainability, and performance are very important. If these properties do not meet the criteria of the SoS or exceed from a well-defined threshold, then the system is useless or will fall into a failure state.

2.2 Microservice Architecture

The microservice architecture style is the approach to develop a single application in terms of small easy-to-test, loosely coupled services to perform their assigned task. The aim of the microservices architectural style is to improve the development process by creating highly independent services that scale better on-demand [14]. Communication in a microservices architecture is done via HTTP REST. The essential features of microservices are fine granularity and loose coupling which means that each microservice can be developed in a different framework, such as programming languages, thus improving interoperability. In contrast to microservices, the monolithic architecture has no flexibility to support continuous development and deployment which is essential in today's highly heterogeneous environment such as in IoT based SoS.

2.3 Cloud Computing

Cloud computing is a distributed approach [15] with the capabilities of providing services such as Infrastructure, Platform, and Software as a Service. The remotely driven computing infrastructure of the cloud provides hardware and software resources on-demand, which helps the cloud world and IoT to grow rapidly and freely. These worlds are very different from each other, but their features often complement each other. In fact, IoT can take advantage of unlimited cloud capabilities and resources such as storage, processing, and communication. Cloud can provide effective solutions for managing and authoring IoT services [16]. In many cases, clouds can be the middle layer between objects and applications. Cloud makes it easy for IoT applications to collect and process data, and to configure and integrate new elements quickly while maintaining complex data processing at a better cost.

3. METHODOLOGY

This research work has started with the observation and analysis of existing architectures from the literature to set a base for the development of software architecture for IoT based SoS. Based on observation and analysis, requirements are obtained which highlight the important challenges and set a path to develop the software architecture. These requirements are obtained after a detailed analysis of architectures from two different domains (IoT and SoS). Research has already been done on IoT and SoS separately, but very few research is found where IoT and SoS are combined. In addition, we have noticed that these works are in their initial stages. Based on these requirements, the architectural style and state of the art technologies are identified that are then used to develop the proposed architecture. The proposed architecture is applied to the smart city case study [17] to check its applicability. Moreover, the design quality is evaluated in terms of scalability and maintainability, as these two quality attributes are important from the IoT-based SoS point of view. The scalability is measured in terms of response time using the Palladio simulator [18], whereas the ripple effect analysis is performed to measure the maintainability. The complete research methodology of developing the proposed software architecture can be seen in Figure 1.

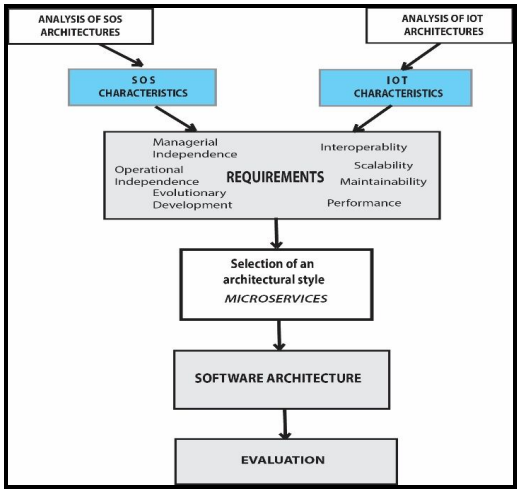

Figure 1: Research Methodology

4. RELATED WORK

There is literature available for smart cities but very limited work is available for SoS based smart cities. Alkhabbas et al. [5] have presented an overview of SoS based on IoT, highlighting some identical properties by showing a relation between IoT and SoS. There are many common characteristics between IoT and SoS, For example, their ICs are heterogeneous, autonomous, distributed, operational, and managerial independent. This study is sufficient to understand the concept of IoT-based SoS. The study [19] presents a conceptual model for the integration of existing systems for the development of smart cities. This research presents a smart city as a distributed system with an emphasis to achieve collaboration among the existing individual systems in order to avoid developing systems from scratch. This SoS based approach strengthens the vision of a smart city by integrating existing individual systems of a city. In [20] a systematic mapping study about design approaches of IoT-based SoS is presented with some research opportunities. The study concludes that most of the efforts are associated with middleware platforms development due to its significance in dealing with issues regarding interoperability between different embedded devices and heterogeneity of ICs. According to this study smart cities might become a reference scenario of IoT-based SoS.

In [21], the authors presented an SOA-based reference architecture for an SoS based smart city to counter the problem of integrating independent systems of smart cities. This study also presents smart city requirements in the context of SoS principles. The reference architecture is enough to understand the concept of SoS based smart cities but lacks an adequate mechanism for the presentation and validation of its architecture in a meaningful way. Similarly, Mohammed et al. [22] present a framework for the integration of a cyber-physical system of systems for smart city mobility applications. The study adds a good body of knowledge for integrating existing cyber-physical systems to present a holistic view of smart city mobility applications. The focus of the study is mainly on the service composition of existing CPS systems.

Conclusion

It is observed from the literature review that the SoS based smart cities are at their initial development stages. The researchers of smart cities are aiming to develop smart cities on the basis of SoS principles but a proper architecture for smart cities in the context of SoS is still missing.

5. THE PROPOSED SOFTWARE ARCHITECTURE FOR IOT BASED SOS

The software architecture proposed in this paper consists of three layers: Device, Cloud, and Service. Figure 2 illustrates the proposed software architecture in which the Cloud layer is the main layer that provides services after getting data from the Device layer. The layers in the proposed architecture are described in detail below.

5.1 Edge or Device Layer

This layer contains all physical hardware devices in the system. This ranges from sensors, actuators to other processing nodes (Raspberry pi / Arduino). It is assumed that each device in the intelligent system has a memory (to store the operating system), a microcontroller, a network interface, and software-accessible device control interfaces. The integrated processing nodes (Raspberry pi / Arduino) have GPIO pins (general purpose input / output) to interface with other nodes (cloud). This layer basically sends data to the cloud layer for analysis.

5.2 Cloud Layer

To perform a high-speed calculation and for large storage, the cloud layer is used. IoT based SoS is composed of a large number of independent information systems, which produce large amounts of data. As a result, IoT requires the collection, access, processing, and sharing of large amounts of data. The cloud offers unlimited storage capacity, low cost, making it ideal and is the most cost-effective way to manage the data generated by IoT systems. Data stored in the cloud can be viewed and accessed from anywhere via APIs.

5.3 Service Layer

This layer contains the services provided by the cloud. In order to achieve interoperability, the service layer allows communication with other services through HTTP REST (a form of JSON). The service is categorized into two types namely simple service and composite service as shown in Figure 2. The simple service is a service that provides a single functionality. A composite service is a group of services combined together to provide functionality. In the proposed architecture, the services have direct interaction with other services without any middleware.

6. APPLICABILITY ON SMART CITY CASE STUDY

This article utilizes the smart city case study [17], whose ultimate purpose is to uplift the living standard of citizens by providing various services such as crime prevention, incident management, and traffic management. In the smart city, various cameras/sensors are mounted on different sites of the city to collect data and then send it to the concerned authority for monitoring. In the crime prevention system, when a crime is reported by a citizen or detected by the system through image analysis, the police are informed to reach the crime spot. The traffic system provides traffic information to citizens and other concerned authorities. The incident management system provides a rescue service with the aim of preventing incidents or any other disaster by taking necessary actions with the help of police, ambulance, and so on.

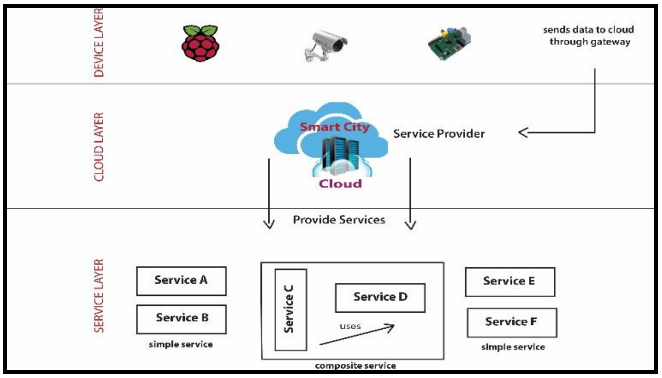

Figure 2: Proposed Software Architecture

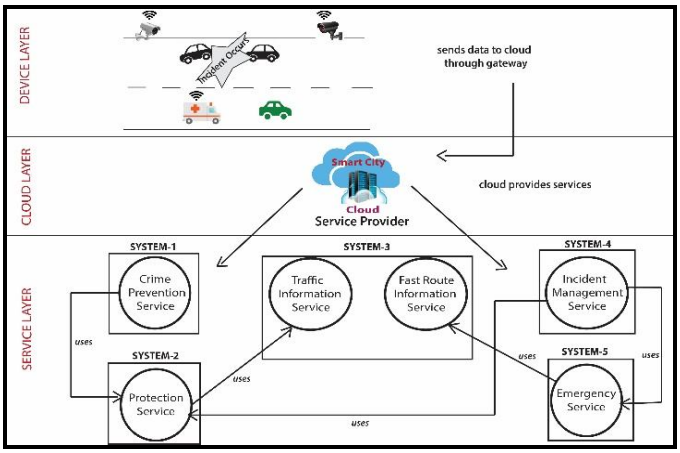

Figure 3: Illustration of the Proposed Architecture on Smart City Case Study

In Figure 3, the case study is mapped using the proposed architecture. It can be seen that the sub-systems of the smart city (crime prevention, incident management, traffic management) work independently and collaboratively to provide various services to the citizen to improve their way of life. Each system has its own set of IoT devices such as camera sensors, actuators, and other processing nodes (Raspberry pi / Arduino, etc). The devices send data to the cloud for analysis. The cloud is the main service provider in the proposed architecture where several services are deployed such as crime detection, incident monitoring, and so on. The service uses other services in order to provide the composite functionality to users. In the case of crime detection, the crime detection service uses the service of police to prevent crime. The police car (equipped with embedded devices) then uses the service of the traffic system to reach the spot immediately.

7. DESIGN QUALITY MEASUREMENT

This section evaluates the quality of the proposed architecture by analyzing its scalability and maintainability.

7.1 Scalability Analysis

Scalability can be defined as the ability of the system to maintain its performance when the size of the system grows. Scalability is an essential quality attribute as it helps in extending the system functionality. In order to analyze the scalability of the proposed architecture, the Palladio simulator [18] is used.to predict software performance at the design level. The simulation is performed in order to get response time. The steps for measuring scalability are defined below.

Step-1:
Initially, we have modeled a crime prevention scenario and incident management scenario in the Palladio simulator by adding components and their interfaces which can be observed in Figure 4. In the crime prevention scenario and incident management, the camera sensor captures images and then send it to the cloud for analysis. In case of any abnormal situation, the city police or ambulance get notified to reach the crime spot. Both scenarios represent a composite service-based scenario with the involvement of more than one service as shown in Figure 4.

Step-2:
In the second step, we set the simulation parameters of services with the help of stochastic expressions. The stochastic expression in the internal action (*service effect specification SEFF*) of a component is used to represent component behavior in terms of its complexity. To predict exact resource demand at the design time is often difficult. In this regard, a stochastic expression can help in order to set a value for resource demand. For example, the expression (CPU: DoublePMF[(5;0.5)(7;0.3)(10;0.2)]) represents the resource demand of a component. This demand translates to as in 50% of the cases the component or more specifically a service requires 5 CPU work units (normal case); in 30% of the cases it requires 7 CPU work units (average case) and in 20% of the cases it requires 10 CPU work units (worst case). In this way, a service resource demand covers all aspects of complexity. The details of service with respect to resource demand can be observed in Table 1.

Step-3:
In the third step, the physical resources are assigned to the components. Some more parameters such as processing speed, and linking resource (network) are set in this stage which can be observed in Figure 5. The components are distributed on different servers like the working of a cloud-based environment.

Step-4:
In the last step, a workload is induced in the system to measure scalability. The workload here refers to the requests made by users or objects (in our case various sensors/cameras are mounted on different sites sends data regularly). In Table 2, two usage models are implemented in the simulator with the best and worst cases.

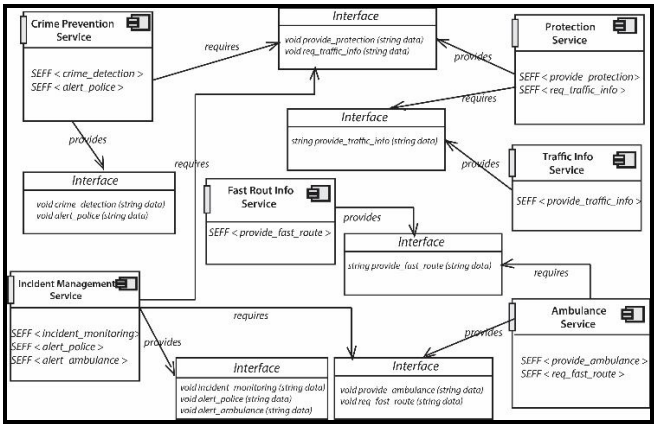

Figure 4: Component Model in Palladio Simulator

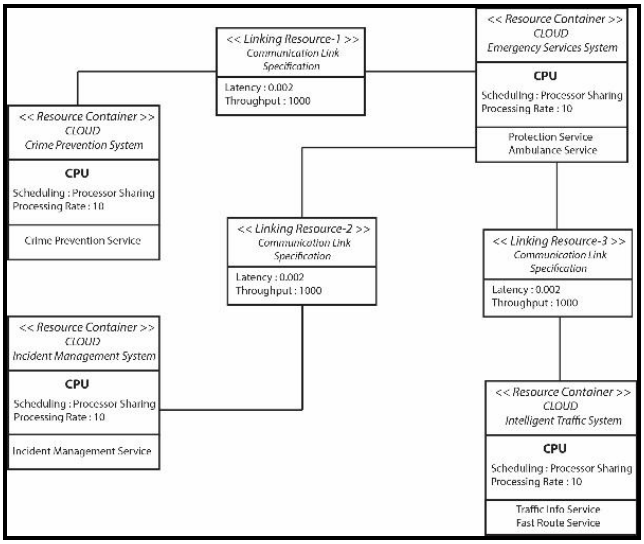

Figure 5: Deployment Model

Table 1: Simulation parameters of services in proposed architecture

Proposed Architecture Services	**Stochastic Expression**
Crime Prevention Service	DoublePMF[(1.0; 0.7)(2.0; 0.2)(3.0; 0.1)]
Protection Service	DoublePMF[(1.0; 0.6)(3.0; 0.4)]
Traffic Information Service	DoublePMF[(1.0; 0.7)(2.0; 0.2)(3.0; 0.1)]
Ambulance Service	DoublePMF[(1.0; 0.6)(2.0; 0.4)]
Fast Route Information Service	DoublePMF[(1.0; 0.7)(2.0; 0.2)(3.0; 0.1)]
Incident Management Service	DoublePMF[(1.0; 0.7)(2.0; 0.2)(3.0; 0.1)]

Table 2: Usage Models

Scenarios	**Best Case**	**Worst Case**
Crime Prevention	After every 60 seconds, a crime happens.	After every 20 seconds, a crime happens 4 times.
Incident Management	After every 60 seconds, an incident happens.	After every 20 seconds, a crime happens 4 times.

7.2 Maintainability Analysis

Software maintainability is an important quality attribute in present-day software systems. This is because changes need to be incorporated in the system with the passage of time to keep the software meet the growing demands. In general, more than 60% of the cost in the software development life cycle is related to the maintainability of the software [23]. In order to validate the proposed architecture in terms of maintainability, ripple effect analysis (REA) [24] is used. REA is a technique to observe the impact of changes made in one software component into other components. In this work, the REA of the proposed architecture is performed through features to service association and service to service association matrix.

1. Feature to Service Association Matrix

In feature to service association matrix, a feature to service mapping is done to observe the changes in services with respect to features. The features can be elicited from user requirements but in our case, it is elicited from the services of a smart city case study which can be observed in Figure 6.

2. Service to Service Association Matrix

In a microservices-based system, there are a number of services. Although the services are often independent, any modification in one service can have an impact on other services. For this reason, we use service to service mapping in order to analyze the effect of one service on another.

7.3 Comparison and Discussion

In this section, we have compared the proposed architecture with smart city architecture namely InterSCity [9] which is based on microservices. In order to compare the results of the proposed architecture with InterSCity, we first model InterSCity architecture on the smart city case study. The working of services coordination in InterSCity can be observed in Figure 7.

1. Scalability Results Comparison

The scalability is evaluated on the basis of two usage scenarios as shown in Table 2. The result of scalability is shown in Figures 8 to 11. The horizontal axis in these figures shows the probability while the verticle axis shows the response time. It can be observed that the proposed architecture is more scalable than InterSCity architecture. In the proposed architecture, the service coordination (orchestration) is done at the system level, thus eliminating middleware. The InterSCity architecture is based on the middleware (orchestrator) which is responsible for the coordination among services. The use of middleware in InterSCity added one extra service as being responsible for service coordination.

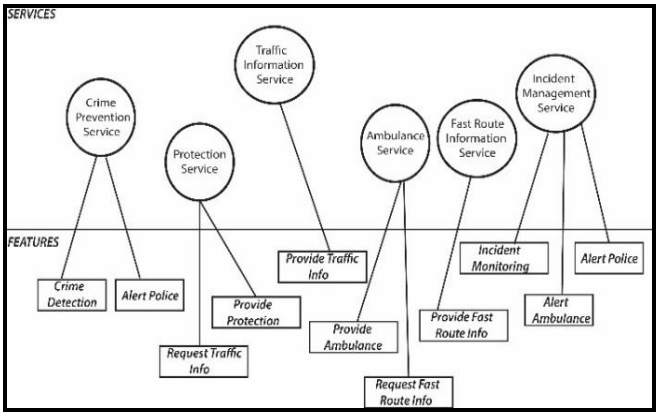

Figure 6: Features Elicited from Smart City Case Study

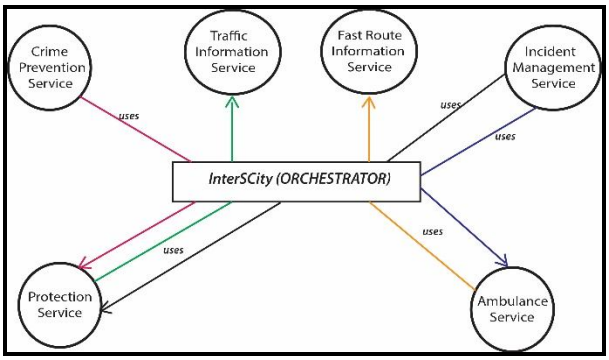

Figure 7: Services Coordination in InterSCity

In this context, the resources demand of services in "InterSCity" is lesser than the proposed architecture because there is one additional service (orchestrator) for the coordination of services. In the modeling of the InterSCity architecture, the simulation parameter (stochastic expression) is set as by keeping resource demand of services idle in 20 to 30% cases that are enough for the justification of not having coordination at service or system-level like in the proposed architecture. The simulation parameters of InterSCity are shown in Table 3.

2. Maintainability Results Comparison

The maintainability is evaluated through REA on the basis of observing changes. A new service (Traffic Violation Service) is introduced in a case study in order to observe the changes as shown in Table 4 and 6. In this way, we are now able to extract those services which are affected by these newly added features and service (shaded green). In Tables 5 and 7, the change can be observed in red shaded color. The proposed architecture is more maintainable than InterSCity architecture as adding new services have no impact on other services. In InterSCity, the services coordinate with other services through middleware (orchestrator). Adding new services would require making changes in middleware and any change in middleware would affect all services as shown in Table 7.

Table 3: Simulation parammeters of services in InterScity architecture

InterScity Architecture Services	**Stochastic Expression**
Crime Prevention Service	DoublePMF[(1.0; 0.6)(2.0; 0.2)(0.0; 0.2)]
Protection Service	DoublePMF[(1.0; 0.6)(2.0; 0.2)(0.0; 0.2)]
Traffic Information Service	DoublePMF[(1.0; 0.6)(2.0; 0.2)(0.0; 0.2)]
Ambulance Service	DoublePMF[(1.0; 0.7)(0.0; 0.3)]
Fast Route Information Service	DoublePMF[(1.0; 0.6)(2.0; 0.2)(0.0; 0.2)]
Incident Management Service	DoublePMF[(1.0; 0.6)(2.0; 0.2)(0.0; 0.2)]
Orchestrator (Middleware)	DoublePMF[(1.0; 0.5)(2.0; 0.2)(3.0; 0.3)]

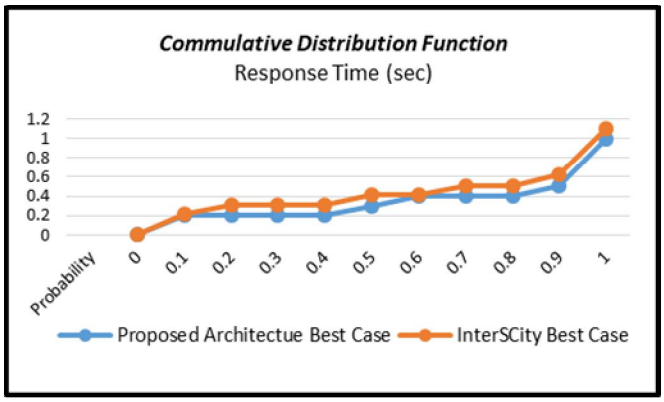

Figure 8: Crime Prevention Scenario Best Case Comparison

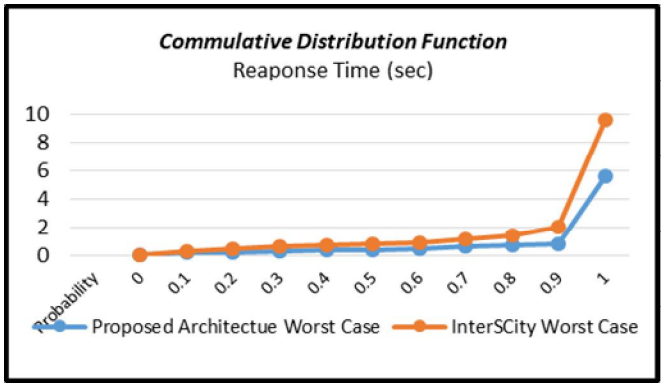

Figure 9: Crime Prevention Scenario Worst Case Comparison

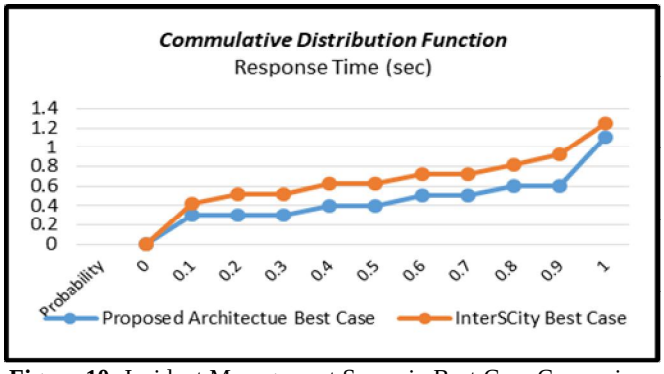

Figure 10: Incident Management Scenario Best Case Comparison

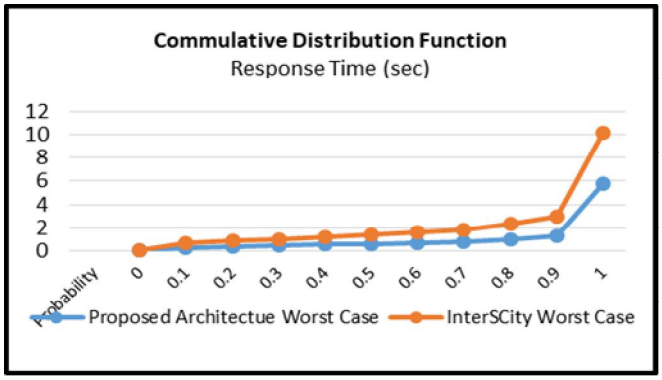

Figure 11: Incident Management Scenario Worst Case Comparison

Table 4: Feature to Service Association Matrix of Proposed Architecture

Services / Features	Traffic Violation Service	Crime Prevention Service	Protection Service	Traffic Information Service	Ambulance Service	Fast Route Information Service	Incident Management Service
Crime Detection	0	1	0	0	0	0	0
Alert Police	0	1	0	0	0	0	0
Provide Protection	0	0	1	0	0	0	0
Request Traffic Info	0	0	1	0	0	0	0
Provide Traffic Info	0	0	0	1	0	0	0
Provide Ambulance	0	0	0	0	1	0	0
Request Fast Route Info	0	0	0	0	1	0	0
Provide Fast Route Info	0	0	0	0	0	1	0
Incident Monitoring	0	0	0	0	0	0	1
Alert Police	0	0	0	0	0	0	1
Alert Ambulance	0	0	0	0	0	0	1
Violation Detection	1	0	0	0	0	0	0
Alert Police	1	0	0	0	0	0	0

Table 5: Service to Service Association Matrix of Proposed Architecture

Services / Services	Traffic Violation Service	Crime Prevention Service	Protection Service	Traffic Information Service	Ambulance Service	Fast Route Information Service	Incident Management Service
Traffic Violation Service		0	0	0	0	0	0
Crime Prevention Service	0		0	0	0	0	0
Protection Service	0	1		0	0	0	1
Traffic Information Service	0	0	1		0	0	0
Ambulance Service	0	0	0	0		0	1
Fast Route Information Service	0	0	0	0	1		0
Incident Management Service	0	0	0	0	0	0	

Table 6: Feature to Service Association Matrix of InterSCity Architecture

Services / **Features**	Traffic Violation Service	Crime Prevention Service	Protection Service	Traffic Information Service	Ambulance Service	Fast Route Information Service	Incident Management Service	Orchestrator
The rest of the features are the same as in Table 4.								
Identifies Request	0	0	0	0	0	0	0	1
Load Balancing	0	0	0	0	0	0	0	1
Coordinates Request	0	0	0	0	0	0	0	1
Violation Detection	1	0	0	0	0	0	0	0
Alert Police	1	0	0	0	0	0	0	0

Table 7: Service to Service Association Matrix of InterSCity Architecture

Services / **Services**	Traffic Violation Service	Crime Prevention Service	Protection Service	Traffic Information Service	Ambulance Service	Fast Route Information Service	Incident Management Service	Orchestrator (Middleware)
Traffic Violation Service		0	0	0	0	0	0	1
Crime Prevention Service	0		0	0	0	0	0	0
Protection Service	0	1		0	0	0	1	0
Traffic Information Service	0	0	1		0	0	0	0
Ambulance Service	0	0	0	0		0	1	0
Fast Route Information Service	0	0	0	0	1		0	0
Incident Management Service	0	0	0	0	0	0		0
Orchestrator (Middleware)	1	1	1	1	1	1	1	

8. CONCLUSION

In this paper, a microservice based layerd architecture for smart city is proposed. The proposed architecture provides a way to model the smart city in terms of services. We have shown a smart combination of SoS principles with microservices to propose a software architecture for smart cities with several quality attributes including salability, interoperability, maintainability.The proposed architecture is applied on a smart city case study. The quality of proposed architecture is evaluated in terms of scalability and maintainability. In future we will improve our architecture by adding some more quality attributes like evolvability, security etc.

REFERENCES

1. Z. Kamal. Aldein Mohammed and E. Sayed Ali Ahmed. **Internet of Things applications, challenges and related future technologies**, *World Scientific News*, Vol. 2 (67), pp. 126-148, 2017.
2. K. Kaiva and G. Atkinson. **What the Internet of Things (IoT) needs to become a reality**, *White Paper, FreeScale and ARM*, pp. 1-16, 2013.
3. A. Whitmore., A. Agarwal,. and L. Da Xu. **The Internet of Things—A survey of topics and trends**, *Information Systems Frontiers*, Vol. 17(2), pp. 261-274, 2015.
4. P. Kruchten. **An ontology of architectural design decisions in software intensive systems**, in *2nd Groningen workshop on software variability*, pp. 54-61. 2004.
5. F. Alkhabbas, R. Spalazzese, and P. Davisson. ***IoT-based Systems of Systems***, 2016.
6. J. Lukkien. **A systems of systems perspective on the internet of things**, *ACM SIGBED Review*, Vol. 13(3), pp. 56-62, 2016.
7. E.Woods. **Software architecture in a changing world**, *IEEE Software*, Vol. 33(6), pp. 94-97, 2016.
8. E. Cavalcante, N. Cacho, F. Lopes, T. Batista, & F. Oquendo. **Thinking smart cities as systems-of-systems: A perspective study**, in *Proc. of the 2nd International Workshop on Smart Cities*, pp. 9:1-9:4. ACM 2016.

9. A. D. M. Del Esposte, E.F. Santana, L. Kanashiro, F.M Costa., K. R. Braghetto, N. Lago, & F. Kon. **Design and evaluation of a scalable smart city software platform with large-scale simulations**, *Future Generation Computer Systems*, Vol. 93, 427441, 2019.
10. N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, & S. Mahmoud. **A service-oriented middleware for cloud of things and fog computing supporting smart city applications**, in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* pp. 1-7, IEEE, 2017.
11. A. Krylovskiy, M. Jahn, & E. Patti. **Designing a smart city internet of things platform with microservice architecture**, in *3rd International Conference on Future Internet of Things and Cloud*, pp. 25-30, IEEE, 2015.
12. D. Namiot, and S. S. Manfred. **On micro-services architecture**, *International Journal of Open Information Technologies* Vol. 2, No. 9, pp. 24-27, 2014.
13. M. W. Maier. **Architecting principles for systems-of-systems**, *Systems Engineering: The Journal of the International Council on Systems Engineering*, Vol. 1(4), pp. 267-28, 1998.
14. D. Shadija, D., Rezai, M., & Hill, R. **Towards an understanding of microservices**, in *23rd International Conference on Automation and Computing* (ICAC), pp. 1-6, IEEE, 2017.
15. T. Erl, R. Puttini, and Z. Mahmood. ***Cloud computing: concepts, technology & architecture***. Pearson Education. 2013.
16. Y. Marcelo, R. Miltio, R. Serral Gracia, D. Montero, and M. Nemirovsky. **Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing**, in *IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 325-329, IEEE, 2014.
17. S. K. Lee, H.R. Kwon, H. Cho, J. Kim,, & D. Lee. **International Case Studies of Smart Cities: Anyang, Republic of Korea**. *Inter-American Development Bank* 2016.
18. S.Becker, H. Koziolek, & R. Reussner. **The Palladio component model for model-driven performance prediction**, *Journal of Systems and Software*, Vol. 82 (1), pp. 3-22, 2009.
19. V. Javidroozi, H. Shah, A. Cole., & A. Amini. **Towards a City's Systems Integration Model for Smart City Development: A Conceptualization**, in *International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 312-317, IEEE, 2015.
20. P. Maia, E. Cavalcante, P. Gomes, T. Batista, F. C. Delicato, & P. F.Pires. **On the development of systems-of-systems based on the internet of things: A systematic mapping**, in *Proc. of the 2014 European Conference on Software Architecture Workshops*, pp. 1-8, ACM, 2014.
21. S.J. Clement., D.W. McKee, & J. Xu. **Service-oriented reference architecture for smart cities**, in *IEEE symposium on service-oriented system engineering (SOSE)*, pp. 81-85, IEEE, 2017.
22. M. Elshenawy, B. Abdulhai, & M. El-Darieby. **Towards a serviceoriented cyber–physical systems of systems for smart city mobility applications**, *Future Generation Computer Systems*, Vol. 79, pp. 575-587, 2018.
23. G. Alkhatib. **The maintenance problem of application software: An empirical analysis**, *Journal of Software Maintenance: Research and Practice*, Vol. 4 (2), pp. 83-104, 1992.
24. S. Anwar, S. Haleem, T. A. Syed, A. Adnan, T. A. Tanveer, M. Alam, M. Ali & A. Rauf. **A FODA Oriented Approach to Architecture Based Ripple Effect Analysis**, *INFORMATION-AN INTERNATIONAL INTERDISCIPLINARY JOURNAL*, Vol. 14 (6), pp. 2139-2149, 2011.