A process model for Service-Oriented Development of Embedded Software Systems

Muhammad Waqar Aziz
Department of Computer Science, CECOS University of IT and Emerging Sciences, Peshawar, 25000, Pakistan

Najeeb Ullah
Department of Computer Science, CECOS University of IT and Emerging Sciences, Peshawar, 25000, Pakistan

Muhammad Rashid
Department of Computer Engineering, College of Computer and Information Systems, Umm Al-Qura University, Makkah, 21955, Saudi Arabia

Abstract—**The concepts of Service-Oriented Computing (SOC) have previously been used in the embedded systems domain, mostly at the device level in ad hoc manners to achieve advantages of device integration. However, SOC has not been used for the development of embedded software systems (ESS), i.e., software controlling the embedded devices. The lack of attention devoted to the application of SOC concepts during the analysis and design of ESS not only hampers the benefits of adopting SOC but also reduces the overall quality of these systems. To fill this gap, a process model is proposed in this paper that allows the systematic development of embedded software systems based on SOC concepts. The proposed process consists of analysis and design phases of embedded software development. The analysis phase is concerned with the collection of system information and preparation for the system design. Based on this, the service-based software architecture is developed in the design phase. The effectiveness of the proposed process model is demonstrated through its application in the Smart Home case study. Experimental results show that the proposed process can reduce coupling and improve cohesion in the software design and, thus, contribute to improving the overall quality of the ESS.**

■ **AN EMBEDDED SYSTEM** is a specialized computing system, which performs a specific function and works as a part of a larger system. Unlike general-purpose computers, embedded systems usually have limited physical resources and have to work under tight timing constraints [1]. The use of embedded systems is increasing in different domains such as aerospace, building and environmental control, critical infrastructure, process control, factory automation, health care and so on [2]. Although an embedded system may roughly be divided into hardware and software parts, the amount of the software in these systems is increasing faster than Moore's law [1]. To control the embedded systems, hardware components are being increasingly replaced by software systems. Consequently, the application of established software engineering practices is needed to cope with the increasing complexity of these software-based

embedded systems. In this context, researchers have been investigating several software engineering paradigms, such as Model-Based Software Engineering (MBSE), Component-Based Software Engineering (CBSE) and Service-Oriented Computing (SOC), for embedded system development [3].
SOC [4] provides several advantages over other paradigms, which reduce the development complexity. For example, SOC provides a higher level of abstraction, enhanced reusability of code, loose coupling, autonomy and dynamic reconfiguration [4]. In the embedded system domain, the capabilities of physical entities can be wrapped as services to enable the use of SOC concepts [5]. By providing a logical service-based view of physical devices, SOC can offer numerous benefits, such as an adaptation of a unifying technology for all levels of the enterprise (from sensors/actuators to enterprise business processes) [6], integration of resources from different levels [7], replacing traditional vendor-specific solutions with popular open standards [8]. Based on SOC, Microservices has recently emerged as an architectural style that fits perfectly well with the use of cloud technology and infrastructure [33]. Microservices style allows engineering new software applications using a set of autonomous small services, which interoperates through message-based communication. It creates an application into a set of conserving and easy-to-test, loosely coupled, reliable units organized around the business features.
SOC concepts have been previously used in embedded systems domain at the device-level in ad hoc manners without any focus on software analysis and design [5, 6, 7, 8]. Consequently, not only the software related issues such as maintainability and reuse are compromised, but it may also hamper the benefits of SOC adoption in this domain. Microservices, in particular, has not previously been used in the analysis and design of ESS to the best of authors' knowledge. The aim of this research work is to systematically develop embedded software systems using microservices. This article presents a service-oriented process model for developing embedded software systems (SOPES). The proposed process model provides descriptive guidance for the systematic development of embedded software systems solely in terms of software services. To be precise, SOPES defines the analysis and design phases of the development of ESS, based on the features and concepts of microservices. The analysis phase deals with the identification of the group of services to be built, whereas the design phase is mainly concerned with building the software architecture using the identified services.
The applicability of the proposed process model is demonstrated via the Smart Home case study. The MBSE [9] approach is followed for accomplishing platform-independent development to reduce the amount of reengineering required by the fast-changing hardware. MBSE allows the development of a system using abstract models [10, 11]. Similarly, SOPES allows the development of high-level design models of ESS, which can be transformed into low-level models and executable code for a specific platform. The details of model transformation and code generation are considered outside the scope of this article. Since service is treated as an "analysis and design concept" in this work, it is intended that SOPES would produce high-quality software systems compare to the ad hoc use of SOC concepts in embedded systems domain. In addition, the definition of systematic analysis and design process and following the MBSE practice would reduce the development complexity of embedded systems.
The remainder of this article is organized as follows: The next section presents a review of different methods used for embedded systems development. Section 3 explains the proposed SOPES and its phases, which is followed by the application of SOPES in Section 4. The results and discussion are provided in Section 5, followed by the conclusion in the last section.

2 RELATED WORK

The development of embedded software is complex and different from enterprise software development due to the specialized characteristics of embedded systems, e.g., hard to change, safety, long operation required, short time-to-market, work in real-time and resource-constrained in terms of memory, bandwidth and power. Furthermore, in recent years the increasing shift of functionality and complexity from hardware to software, have made embedded software development as one of the biggest challenges in the embedded systems domain. As a consequence, a lot of research has been carried for engineering of these software systems. Based on the software development approaches used, we have classified existing methods as component-based, model-driven and service-oriented methods.
With respect to CBSE, some component technologies for embedded systems exist for quite a long time, such

as PECOS, Koala, and ROBOCOB [12]. There also exist some component models for embedded systems development, such as ProCom (PROGRESS Component model) [13], SOFA HI [14] and BlueArX [15]. ProCom addresses explicit separation of concerns at different levels of granularity. SOFA HI, an extension of SOFA 2 component model, is targeted at high integrity real-time embedded systems. SOFA 2 is an advance distributed component system that provides complete support for all the stages of application development and deployment [12]. BlueArX has been developed for the traditional automotive domain that focuses on design time component models to support resource constraints and non-functional requirements. This also provides different views of a developed system [12].

Regarding model-driven development, Harmony/ ESWTM [16] is an effective MBSE process for the development of embedded real-time applications. Harmony is an incremental development process, consisting of analysis and design phases. Besides providing several advantages, the Harmony process is focused on solving the process and management issues of embedded real-time system development and does not provide the advantages offered by SOC. Similarly, the UML MARTE profile [17] is proposed as model-based description method of embedded real-time systems. MARTE provides support for modeling of time, resources, NFP and concepts for software and hardware resources. Some other research works on model-driven development for embedded systems are also based on UML, for instance [18].

Both CBSE and MBSE approaches lack in providing features like loose coupling, automatic discovery and dynamic composition that SOC provides. Additionally, SOC offers other advantages that are missing from the previous development paradigms, such as, commonality of functionality among several clients, publish/discover paradigm, dynamic composition, and exchange of documents between services. Due to providing these advantages, SOC has been applied in embedded systems development [8, 19, 20], for producing intelligent manufacturing systems [5, 6, 7, 21] and in developing robotic systems [22]. Additionally, SOC has been used in several European research projects related to industry automation, such as SIRENA [23] and SOCRADES [24]. But, in all of these works, service is used as an implementation concept either to achieve the interoperability between the devices or integrating devices with the enterprise software.

Despite a large number of studies on using SOC in the embedded domain, very few have a focus on proposing a systematic process for service-oriented development of embedded systems. Ermagan et al. [25] presented a systematic software development process for service-oriented development of distributed embedded systems. Although service is considered as a first-class modeling concept in [25], it is defined as the interaction between the entities. Moreover, the process depends on the underlying component model for architecture deployment. A service model is used just to abstract the underlying component model. Finally, the process is supposed to be for software engineering in the automotive domain only. More recently, a software process for designing software architectures of service-oriented robotic systems is presented [22]. Yet, all the phases of the process target the development of software architecture for robotic systems. It would be difficult to apply the process to other types of embedded software systems.

Due to this gap produced by the lack of a systematic service-oriented process model, the embedded systems have been developed in an ad hoc manner. Therefore, a systematic process for service-oriented development of embedded software systems is still necessary and can potentially contribute to the embedded systems domain.

3 PROPOSED PROCESS MODEL

SOPES is a systematic process that allows the service-oriented development of embedded software systems. In this research, an embedded system is treated as a system composed of a variety of physical entities (called devices in this article) providing diverse functionalities. These device functionalities are wrapped (and hence termed) as services, to enable the use of SOC technologies. The proposed process consists of two phases to explicitly consider the analysis and design of the embedded software. The analysis phase allows identifying the embedded devices, the services they provide and their interactions. In design phase using the identified services the software architecture is built. The details of these phases are provided as follow:

3.1 Analysis Phase

The analysis phase comprises of the following activities, as shown in **Figure 1**.

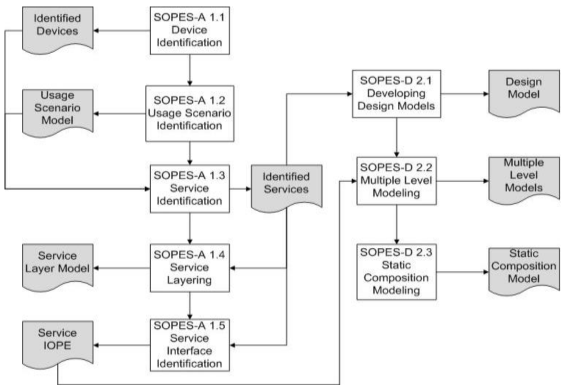
Identified Devices
SOPES-A 1.1 Device Identification
Usage Scenario Model
SOPES-A 1.2 Usage Scenario Identification
SOPES-A 1.3 Service Identification
Identified Services
Service Layer Model
SOPES-A 1.4 Service Layering
Service IOPE
SOPES-A 1.5 Service Interface Identification
SOPES-D 2.1 Developing Design Models
Design Model
SOPES-D 2.2 Multiple Level Modeling
Multiple Level Models
SOPES-D 2.3 Static Composition Modeling
Static Composition Model

Figure 1. Activities and products of analysis phase (left) and design phase (right) of SOPES

SOPES-A 1.1 – Device Identification: All the devices (and their components) present in the system are identified, whether they are providing or using the service(s) or both.

SOPES-A 1.2 – Usage scenario definition: The usage scenarios of the devices are defined where each scenario describes the interaction among the physical entities involved in the usage. The usage scenarios further specify the order (workflow) in which the activities would take place.

SOPES-A 1.3 – Service Identification: The services provided by the devices are identified using the service identification guideline for embedded systems [26]. The guideline provides profound descriptions of identifying the services in embedded systems. The output of this activity is the list of the identified services. The resulting identified services not only portray the devices' functionalities in terms of services but also classify them into atomic and composite types. An atomic service represents a basic functionality, which can be combined with other services to build a composite service.

SOPES-A 1.4 – Service Interface Identification: In this activity, inputs and outputs of the service internal operations are identified to specify its behavior. In addition, the requirements of devices that need to be fulfilled for executing a process (Preconditions) and the results obtained after process execution (Effects) are recorded. A service interface is defined in terms of Inputs, Outputs, Pre-Conditions, and Effects (IOPE) of the service.

3.2 Design Phase

The design phase of SOPES is aimed at producing the software architecture of ESS. This software architecture is basically a Platform Independent Model – in MBSE jargon, which does not concern with the platform-specific details. Thus, the focus of development shifts to function-based instead of code-based engineering. This phase comprises of the following activities.

SOPES-D 2.1 – Developing Software Architecture: The software architecture is developed using the services identified in the analysis phase. This development follows the Domain-Specific Modeling Language (DSML) for cyber-physical systems [28] that provides the means to model the structure and behavior of embedded systems in terms of SOC concepts. In addition, DSML also allows the modeling of temporal and other quality of service characteristics. This DSML is defined formally in terms of a meta-model and implemented as a UML profile [29]. In this way, it allows using the existing UML tools to develop the software architecture.

SOPES-D 2.2 – Multiple Level Modeling: The software architecture (abstract high-level model) of the ESS can be detailed into low-level design models using the multiple levels of abstraction modeling for embedded systems [30]. The multiple levels consist of four types of models: Device-Level Design Model (representing a set of interacting devices only), Service-Level Model (highlighting the services and their providers), Interface-Level Model (show the interfaces of the provided services) and Service-Detail Model (displaying the elements of the service) [30]. This level-by-level modeling simplifies the design of embedded systems, as only the relevant information is exposed at a particular level and reduces the design complexity by representing the system at different levels of abstraction.

SOPES-D 2.3 – Static Composition Modeling: Service composition is a process where different services are combined together to build a more value-added complex service [4]. Although service composition is a vast research topic on its own, SOPES facilitates the modeling of the static composition just to provide completeness towards the SOC concepts.

In static composition, the participating services, their providers and the workflow of activities are known and based on which atomic services are composed together at the design time [4]. In SOPES, static composition can be modeled based on the usage scenarios of the devices, as defined in the analysis phase. The atomic services can be composed together using these usage scenarios and workflows at the design time. The entire composition process can be modeled using the service-oriented design models for embedded systems [30]. Dynamic service composition is performed at run-time, which is beyond the analysis and design scope of SOPES.

4 SOPES APPLICATION IN SMART HOME

The Smart Home case study is used, in this work, as smart homes have emerged as a focused application area of embedded systems [31]. The case study consists of heterogeneous embedded devices distributed across a home, which communicate in real-time (having hard and soft temporal requirements). The devices used in the case study were limited to a certain number in order to have a better understanding of service-oriented concepts in general and the proposed approach in particular. The devices include white goods, consumer electronics, building automation, and environmental sensors. Each device in the case study has a built-in micro-controller, flash program memory, and internal Random Access Memory (RAM). Most of the actions in the case study are event-based and involve human interaction.

4.1 Analysis Phase

During this phase, the communicating devices in the Smart Home and their processes were identified. The following services were identified by applying the service identification guideline [26].

- *Device Services*: Telephone
- *Composite Services*: Cooking, Food order, Check Food, Temperature control
- *Functional Services*: LowHigh Volume, LowHigh Light, Display, Read cooking Instructions, Weight Food item, Place Order, Read expiry date, Check Temperature, LowHigh Temperature
- *External Service*: Order Processing and
- *Application Services*: Send SMS, Send e-mail

The service interfaces were defined in terms of IOPE of the services. **Table 1** presents the inputs and outputs of each of the identified services. Similarly, the preconditions and effects of each service were identified and recorded (as tabulated in **Table 2**).

4.2 Design Phase

The design models (software architecture and low-level models) for the Smart Home were developed using the UML tool Papyrus [32]. The Smart Home structural model is presented in **Figure 2**, where the service design model clearly distinguishes the composite services.

Contrary to the ad hoc approach, our proposed method SOPES is a systematic approach. The proposed approach, using autonomous service identification and better utilization of service-oriented concepts, helps in attaining loose coupling. With the help of our proposed approach SOPES quality attributes are incorporated in the software. For instance, low coupling among services and among service operations are obtained, when SOPES is used. Because of the loose coupling more service cohesion is achieved. Consequently maintainability of the system will be improved. Furthermore, the special attention given to service identification process (step-by-step service identification guideline) in SOPES helps in attaining low complexity. The reason behind this is the low number of services (as more services mean more complex system). Hence portability of the system will be enhanced. Thus, our proposed approach helps in developing better quality system as compared to the ad hoc approaches. The proposed approach may help practitioners in saving a lot of re work, efforts and resources.

Table 1. Inputs/Outputs of the identified services.

Service	Inputs	Outputs
Low Light	Intensity Level	-
High Light	Intensity Level	-
Low Volume	Volume Level	-
High Volume	Volume Level	-
Display Text	Text	-
Read RFID Tag	-	Cooking Instructions
Cook Food	Cooking Instructions	Food Ready Message
Send SMS	Food Ready Message	-
Display Teletext	Food Ready Message	-
Weight Food Item	-	Food Weight
Order	Item Code; Quantity	Receipt
Read Expiry Date	-	Expiry Date
Low Temperature	Temperature Level	-
High Temperature	Temperature Level	-

Table 2. Pre-condition/Effects of the identified services.

Service	Inputs	Outputs
Low Light	Light = ON ; Input intensity level < current intensity level	current intensity level = Input intensity level
High Light	Light = ON ; Input intensity level > current intensity level	current intensity level = Input intensity level
Low Volume	TV = ON ; Input volume level < current volume level	current volume level = Input volume level
High Volume	TV = ON ; Input volume level > current volume level	current volume level = Input volume level
Display Text	TV = ON	Text displayed
Read RFID Tag	Oven = ON ; Food item at proper place-	RFID tag read Instructions
Cook Food	Oven = ON ; Temperature set according to cooking instruction	Food Ready Message
Send SMS	Food is cooked	SMS send
Display Teletext	Food is cooked	Text displayed
Weight Food Item	Fridge = ON ; Weight machine is working ; Food item on weight machine	Food item weighted
Order	User acceptance	Order placed
Read Expiry Date	Fridge = ON ; Food item at proper place	Expiry date read
Send SMS	Expiry date = current date + 7	SMS send
Low Temperature	AC = ON ; Input temperature < current Temperature	Current Temperature = Input temperature
High Temperature	AC = ON ; Input temperature > current Temperature	Current Temperature = Input temperature

5 CONCLUSION

The amount of software and its development complexity is increasing in embedded systems. To handle this, a systematic process model for the development of embedded software systems is presented in this paper. The proposed process model (termed as SOPES) is based on service-oriented computing concepts. SOPES defines the analysis and design phases of software development in a systematic way, with details of activities involved. By using the service concept at system analysis and design level, it is intended that more benefits of service-oriented computing would be achieved, compared to its use at the device level in ad hoc manners only. This would lead to producing high quality embedded software systems.

On the other side, the systematic definition of the analysis and design phases would streamline the development of embedded software systems and thus would reduce the development complexity. This eventually would increase the productivity of embedded systems and reduce the time-to-market and the development cost. Although the applicability of the proposed process model was demonstrated in the smart home case study to check the soundness of the presented concepts, it is general enough to be applied to the development of any embedded software system. In the future, it is planned to apply SOPES for the development of more complex embedded systems.

■ REFERENCES

1. Oshana R, *Software Engineering of Embedded and Real-Time Systems*: Elsevier, 2013.
2. Rajkumar R, Lee I, Sha L, and Stankovic J. “Cyber-physical systems: The next computing revolution,” *47th ACM/IEEE Design Automation Conference (DAC)*, pp. 731–736, 2010.
3. Vyatkin V. “Software engineering in industrial automation: State-of-the-art review,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1234–1249, 2013.
4. Erl T. *Service-Oriented Architecture: Concepts, Technology, and Design*: Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
5. Jammes F, Smit H. “Service-oriented paradigms in industrial automation,” *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 62–70, 2005.
6. Cannata A, Gerosa M, and Taisch M. “A Technology Roadmap on SOA for smart embedded devices: Towards intelligent systems in manufacturing,” *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, pp. 762-767, 8-11 December, 2008.
7. Mendes J M, Bepperling A, Pinto J. Leitão P, Restivo F, and Colombo A W. “Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project,” *33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC '09)*, Seattle, Washington, pp. 452-459, 20-24 July, 2009.

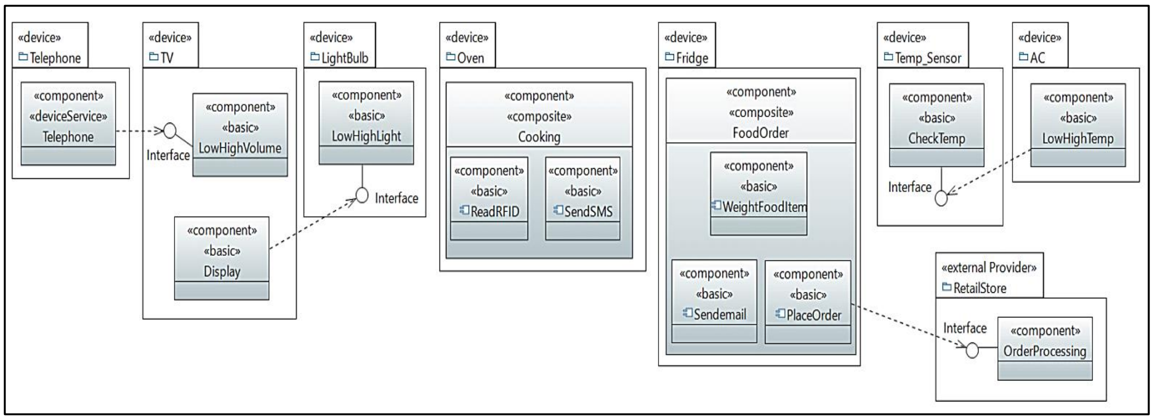
«device» Telephone
«component» «deviceService» Telephone
«device» TV
Interface
«component» «basic» LowHighVolume
«component» «basic» Display
«device» LightBulb
«component» «basic» LowHighLight
Interface
«device» Oven
«component» «composite» Cooking
«component» «basic» ReadRFID
«component» «basic» SendSMS
«device» Fridge
«component» «composite» FoodOrder
«component» «basic» WeightFoodItem
«component» «basic» Sendemail
«component» «basic» PlaceOrder
«device» Temp_Sensor
«component» «basic» CheckTemp
Interface
«device» AC
«component» «basic» LowHighTemp
«external Provider» RetailStore
Interface
«component» OrderProcessing

Figure 2. High-level model of embedded software system for Smart Home

8. Shopov M P, Matev H, and Spasov G V. "Evaluation of Web Services Implementation for ARM-Based Embedded System," *Proceedings of ELECTRONICS '07*. 19-21 September. Sozopol, Bulgaria, pp. 79-84, 2007
9. Beydeda S, and Book M. Model-driven software development, vol. 15. Heidelberg: Springer, 2005.
10. Herrera F, Posadas H, Peñil P, Villar E, Ferrero F, Valencia R, and Palermo G. "The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems," *Journal of Systems Architecture*, vol. 60, no. 1, pp. 55-78, January 2014.
11. Rashid M, Anwar M W, and Khan A M. "Towards the Tools Selection in Model Based System Engineering for Embedded Systems-A Systematic Literature Review," *Journal of Systems and Software*, vol. 106, pp. 150-163, August 2015.
12. Hošek P, Pop T, Bureš T, Hnětynka P, Malohlava M. Comparison of Component Frameworks for Real-Time Embedded Systems. In Component-Based Software Engineering 6092, Grunske L, Reussner R, Plasil F (Eds.), Berlin, Heidelberg: Springer, 2010, pp. 21-36.
13. Bures T, Carlson J, Crnkovic I, Sentilles S, and Vulgarakis A. ProCom - the Progress Component Model Reference Manual: Mälardalen Real-Time Research Centre, Mälardalen University, June 2008.
14. Prochazka M, Ward R, Tuma P, Hnetynka P, and Adamek J. "A component-oriented framework for spacecraft on-board software," *Proceedings of the Data Systems In Aerospace, DASIA*. Noordwijk, Netherlands, August 2008.
15. Kim J E, Rogalla O, Kramer S, and Hamann A. "Extracting, specifying and predicting software system properties in component based real-time embedded software development," *31st International Conference on Software Engineering - Companion Volume (ICSE)*, Vancouver, BC, pp. 28-38, 16-24 May 2009.
16. Douglass B P. Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development (1st ed.): Prentice Hall, 2009.
17. OMG. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems (Vol. 1.0): Object Management Group, Inc. 2009.
18. Ito K, Matsuura S. "Model driven development for embedded systems," *Proceedings of the 9th WSEAS international conference on Software engineering, parallel and distributed systems*, Cambridge, UK, pp. 102-108, 20-22 February 2010.
19. Hoang D D, Paik H-Y, Kim C-K. "Service-Oriented Middleware Architectures for Cyber-Physical Systems," *International Journal of Computer Science and Network Security*, vol. 12, no. 1, pp. 79-87, 2012.
20. Rodrigues D, de Melo Pires R, Estrella J C, Marconato E A, Trindade O, Branco K R L J C. "Using SOA in Critical-Embedded Systems," *International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing (iThings/CPSCom)*, Dalian, pp. 733-738, 19-22 October 2011.
21. Giret A, Garcia E, Botti V. "An engineering framework for service-oriented intelligent manufacturing systems," *Computers in Industry*, vol. 81, pp. 116-127, 2016.
22. Oliveira L B R, Leroux E, Felizardo K R, Oquendo F, Nakagawa E Y. ArchSORS: A Software Process for Designing Software Architectures of Service-Oriented Robotic Systems. The Computer Journal, 2017, 1-19.
23. Jammes F, and Smit H. "Service-oriented architectures for devices – the SIRENA view," *3rd IEEE International Conference on Industrial Informatics (INDIN '05)*, Perth, Australia, pp. 140-147, 10-12 August 2005.
24. Cannata A, Gerosa M, and Taisch M. "SOCRADES: A framework for developing intelligent systems in manufacturing," *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, pp. 1904-1908, 8-11 December 2008.
25. Ermagan V, Huang T-J, Krüger I, Meisinger M, Menarini M, and Moorthy P. "Towards Tool Support for Service-Oriented Development of Embedded Automotive Systems," *Proceedings of the Dagstuhl Workshop on Model-Based Development of Embedded Systems (MBEES)*, Schloss Dagstuhl, Germany, 15-18 January 2007.
26. Mohamad R, Aziz M, Jawawi D N, et al. "Service identification guideline for developing distributed embedded real-time systems," *IET Software*, vol. 6, no. 1, pp. 74–82, 2012.
27. Aziz M W. "Service-oriented layered architecture for smart home," *International Journal of Smart Home*, vol. 7, no. 6, pp. 409–418, 2013.
28. Aziz M W, Rashid M. "Domain Specific Modeling Language for Cyber Physical Systems," *IEEE International Conference on Information Systems*

Engineering (ICISE), Los Angeles, CA, pp. 29-33, April 2016.

29. Muhammad W A, Radziah M, Dayang N A J. “SOA4DERTS: A Service-Oriented UML profile for Distributed Embedded Real-Time Systems,” *IEEE Symposium on Computers & Informatics (ISCI)*, pp.64-69, 18-20 March 2012.
30. Aziz M W, Mohamad R, Jawawi D N. “Multiple levels of abstraction modeling for service-oriented distributed embedded real-time software design,” Informatics Engineering and Information Science, Springer, 2011, pp. 517–528.
31. Balta-Ozkan N, Amerighi O, Boteler B. “A comparison of consumer perceptions towards smart homes in the UK, Germany and Italy: reflections for policy and future research,” Technol. Anal. Strat. Manag. 26, 2014, 1176–1195. http://dx.doi.org/10.1080/ 09537325.2014.975788.
32. Gérard S, Dumoulin C, Tessier P, Selic B. “Papyrus: A UML2 Tool for Domain-Specific Language Modeling,” *Model-Based Engineering of Embedded Real-Time Systems*, LNCS: Springer, 2010, pp. 361-368.
33. Nadareishvili I., Mitra R., McLarty M., & Amundsen M. Microservice architecture: aligning principles, practices, and culture. O'Reilly Media, Inc. 2016.

Muhammad Waqar Aziz is Associate Professor at CECOS University of IT and Emerging Sciences, Pakistan. He received his Ph.D. (Computer Science) from Universiti Teknologi Malaysia in 2013, MS-Software Engineering from City University of Science and Technology in 2009 and MSc-Computer Science from University of Peshawar in 2001. Previously, he has worked in Umm Al-Qura University, Saudi Arabia for more than three years, where he was involved in different research projects and successfully completed them. Before that, he has almost eight years of teaching experience as Lecturer at Institute of Management Studies, University of Peshawar. He published more than 20 research articles in indexed and well reputed journals and Conference Proceedings. The research areas of his interest are Real-Time Systems and Smart Environments. Contact him at waqar@cecos.edu.pk.