https://doi.org/10.51153/kjcis.v5i2.115

An Intuitive Architecture for DIY IoT Application Composition as Business Process Model

Muhammad Sohail Khan[1] DoHyeun Kim[2] Faiza Tila[2]

Abstract

Internet of Things (IoT) and the related technologies have changed the users' perspective of remote service utilization. Internet of Things is a network of billions of smart and connected devices, which expose their functionality in the form of sensing or actuation services. Currently, applications utilize those services to cater to the user's needs. However, a user's requirements are always changing which cannot be efficiently fulfilled by the domain specific applications. We propose the idea of using Business Process Modeling approach in order to enable the user to model their required processes based on the atomic services exposed by IoT devices. The approach requires no programming skills on behalf of the users so it can be used as a Do-It-Yourself (DIY) tool for the creation of IoT based processes. This article presents the architecture and detailed design of the proposed system.

Keyword: Business process modeling, Internet of Things, Architecture, DIY, Application.

1. Introduction

Recent years have shown the growth in business and research related to Internet of Things (IoT). According to IoT Analytics, by the year 2025, there will be 27.1 billion devices connected to the internet while currently it is around 13 billion, an unexpectedly low growth due to the COVID-19 pandemic and chip shortage [1]. The International Data Corporation (IDC) reported in in latest reports, that the IoT spending in the Asia Pacific region only will reach 437 billion USD till 2025 with an expansion of 9.6% as compared to the 2020 growth rate of 1.5% [21]. This encourages the academia and industry to strive for the realization of IoT vision. Several standardization efforts are underway and IoT architectures have been presented with a focus to standardize the way IoT systems and applications are developed.

Business Process Management (BPM) has been at the center of close collaboration between business and IT for a long time. This popularity of the BPM solutions is mostly due to the standardization of diagramming language known as the Business Process Modeling Notations (BPMN) [2]. The initial aim of BPMN was to enable the business

[1]*University of Engineering & Technology, Mardan, Pakistan | sohail.khan@uetmardan.edu.pk*
[2]*Jeju National University, South Korea | kimdh@jejunu.ac.kr, faizakhan797@gmail.com*

analyst to describe a business's desired process through a diagram and to accommodate the business agility through automated execution of the process model.

Service Oriented Architecture (SOA) [3] has since been at the forefronts of BPM solution implementations with the promise of service reuse. Service reuse is the encapsulation of a system's atomic functions as reusable service units with well-defined interfaces. These reusable services can provide easy and rapid integration of new composite processes hence making the IT to become more agile. BPMN can be utilized for more than just a means for business requirements gathering rather it is considered as a prominent solution for fast passed Industry 4.0 related process driven applications [4]. A recent survey by BpTrends [5] reveals that about 75 percent of the enterprises believe that BPM processes and technologies have helped them achieve their business goals, 73 percent reported an increasing interest in BPM since 2019 and 71 percent of the survey respondents reported the current digital transformation in the technology as the motivator towards BPM adoption. This indicates the importance and awareness of BPMN among businesses and its importance in the current technological scenario.

The current technologies in the form of sensors and actuators networks, web of things [6] and most of all the realization of the Internet of Things (IoT)[7] involves ever changing requirements and implementation within the ecosystem of resource constrained hardware, services and people. Until recently, accommodation of user desired change in applications associated with these paradigms was not easily possible due to the resource constraints, heterogeneous nature of the hardware and the lack of standardization in the communication strategies. With the recent improvements of security in REST [8], resource constrained protocols such as MQTT [9] and the recent CoAP [9] Protocol, a flexible service orientation has become possible for the things associated with IoT. Service composition and orchestration has been introduced to the recent developments in IoT and other associated paradigms. SENSEI [10] is a business driven IoT architecture for the scalability of large numbers of sensors and actuators associated with Internet of Things. It is focused on providing services for accurate retrieval of contextual information and interaction with physical entities.

MoCoSo[11] is another such project which is focused on the combination of various concepts such as identification of objects, contextual data and communication medium (Internet). The main idea is to integrate sensor networks into a larger Internet of Things. Systems focused towards the application of IoT in industry have also been initiated. Collaborative Business Items (CoBIs) project is a core project which aimed at enabling industrial objects such as machine parts and containers etc. to communicate with each other at their surroundings. This goal is achieved by making the items uniquely identifiable in the system and incorporating various sensors in the said items. The items exposed their behavior as services and the system provided an infrastructure for centralized service

deployment across the network.

Most of the above projects belong to the initial era of effort towards the realization of IoT vision and are at least 5 to 10 years old. A more recent project is the Compose project [20]. This project provides an open and scalable platform infrastructure enabling easy creation of application based on IoT. The main theme of the project is to represent the Internet connected smart objects as programming entities which can be utilized to program larger applications. This project, however, is composed of several open source technologies which form a complex integrated infrastructure in order achieve its goals and in there somehow lacks the intuitiveness for daily life, non-programmer users that a DIY type of system can provide.

Research community has embraced the potentials of a resource constrained device in the paradigm of business process management and hence efforts have been done to include things and services as part of the BPMN. In this regard, [12] provided the missing concept of "thing", as presented by the main components of IoT reference model [13], by extending the conventional meta-models for business process modeling notations. Similarly, [14] proposes the extension of the business process modeling lifecycle for the integration of IoT in it.

Despite the efforts to integrate IoT as part of the BPM lifecycle. Traditionally, business process modeling and the demand for rapid incorporation of change have always been based on service reuse and service composition. This practice, although proven effective at times, has not been always effective [4]. IoT is also not just some enterprise implementation of proprietary services for specific goals which can be composed upon to execute processes. In fact, IoT is a vast ecosystem of billions of devices which present themselves as atomic services on the Internet. These services must be available for masses to utilize for making their local solutions as well as to share them. This concept is also termed as the Do-It-Yourself (DIY) paradigm of development for the realization of IoT vision.

This idea has been advocated my many such as [15], [16] states that the end-users should be part of the creation process while having the power to discover things. Similarly, [16], [17] suggests that the end-users should be able to discover things and control them in order to effectively use the application for smart environments. The vision and motivations of Makers Revolution [18], the advancement in DIY prototyping platforms such as Arduino and Raspberry Pi etc. [19] and the ongoing standardization of communication protocols for constrained devices are all the right steps towards inculcating DIY culture in masses. However, the masses may not have the skills and the ability to program these embedded devices for their DIY implementations especially with the growing number of programming languages currently being used for IoT implementations. BPM and the associated diagramming language may prove useful as a solution to the problem.

BPM has always been envisioned as the tool to enable the managers and people with no programming skills to describe their needs and desired processes. We propose the utilization of BPM diagramming language for the IoT users to make their own desired processes and let the SOA enable those models to be executed in their environment. Our vision is to let the user model the process based on the available atomic services which have some level of composition and then integrate them with user defined rules to model their processes and directly execute those processes without the need to alter the underlying services according to the process model. Such an approach can enable the users to easily model and execute their desired processes and hence make IoT applications agile with respect to the user requirements.

This paper presents the architecture and design of the layered system for the realization of our vision. The rest of the paper presents a detailed description of the system architecture and design with the help of static and sequence models while the preliminary semantic model has been presented in the form of Protégé based implementation.

2 System Architecture

Fig. 1 shows the conceptual architecture for the system. The architecture consists of four layers each of which performs its own functions and communicates with the adjacent layers through a predefined communication scheme. A brief introduction of the system based on each layer is provided in the following paragraphs.

The physical layer consists of things associated with the Internet of Things. These things are capable of sensing data from their surroundings and/or controlling certain phenomenon happening around them. These things can simply be termed as sensing and actuating devices with the capability of communication through the internet.

The Virtual Object Layer (VOL) represents the physical layer things in the form of Virtual Objects. A Virtual Object (VO) is the in-system representation of a thing at physical layer. A virtual object encapsulates the information associated with a physical thing and enables the users to manipulate the VO inside the system environment, interact with it and through the VO interact with the environment of the physical things represented by the VO.

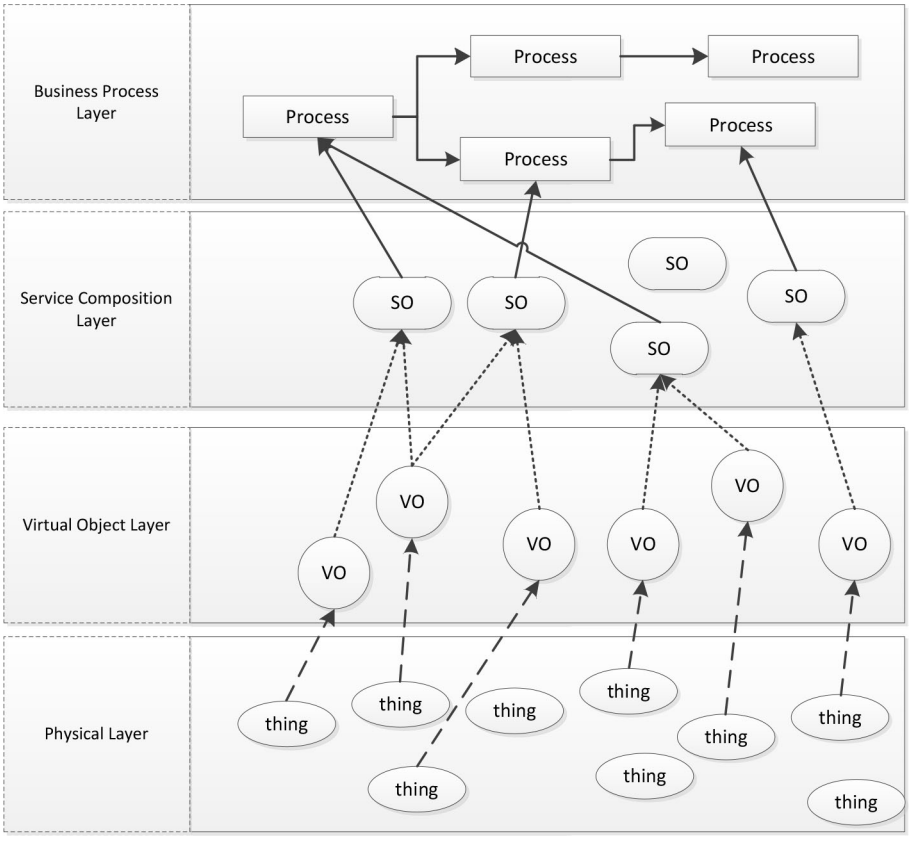

Figure 1: BPM based IoT system model

The virtual objects at the virtual object layer are utilized by the Service Composition Layer (SCL) in order to compose Service Objects (SO) by combining the functionalities offered by two or more virtual objects. A simple SO may thus contain an input VO joined with an output stream to display the data generated by the input VO. A more complex example of a service object would be to join a temperature sensor VO with an LED VO with the settings that when temperature value exceeds 40 degrees Celsius, the LED should start blinking. The acquisition of the temperature value and the blinking of the LED depend on the functionalities encapsulated by their corresponding VOs.

Once the SOs are created, these atomic service objects are utilized by the Business Process Layer (BPL) to formulate the flow of a user-desired process for a certain context. The business process layer utilizes the business process modeling notations (BPMN) to model the process based on the combination of SOs. The process flows are executed at the business process layer to carry out the functionality as defined by the individual service objects and actual interactions are carried out directly with the physical things based on their behavior encapsulated by their respective VOs.

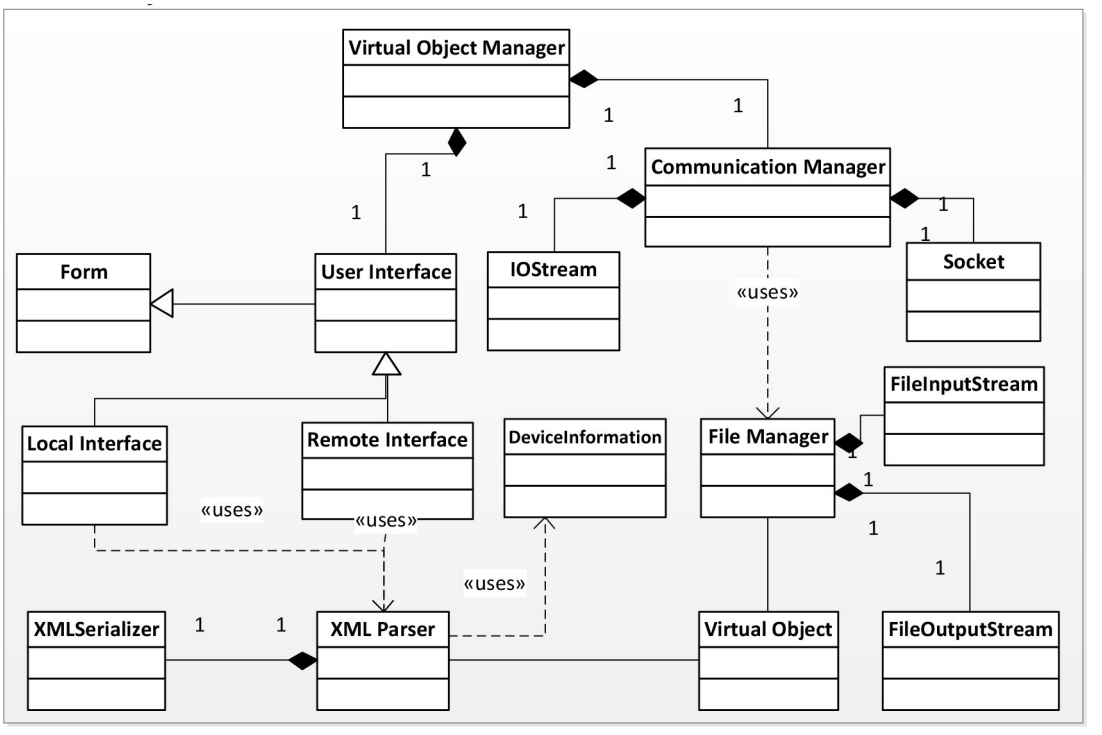

Figure 2: Static structure for virtual object manager

3. Detailed Design

This section presents the design details of the layers as presented in the previous section. The design of each layer includes the static structures and the interaction design for describing the main operations at each layer. The following sub-sections present the design details of each layer.

3.1 Object virtualization

The virtual object manager is the main component at the virtual object layer. It in collaboration with other classes such as the File Manager, Communication Manager and Parser etc., provides the implementation of all the functionality associated with the VOL. The static structure of VOM is shown in Fig. 2. VOM is the composition of the local and remote interfaces classes which enables the users to input information related to the physical things for which they want to register virtual objects. Similarly, the Communication Manager uses the File Manager for retrieving the XML version of the virtual objects from the local file system and to send it the client application. The client application in this scenario would be the service composition manager. The XML Parser works in collaboration with the File Manager and the interfaces to convert the information entered by users into xml elements representing the VO and vice versa. The

XML Parser uses the DeviceInformation class as a template for the creation of VOs.

Fig. 3 shows the internal process of the virtual object manager in the form of a sequence diagram. The sequence model shows the interaction of user with the interface component as well as the resultant interactions in the form of messages exchange among the other internal components of the system in order to fulfill the user commands. The sequence of interactions starts when the VOM is started and all the components including the user interface and the Communication Manager etc. are initialized.

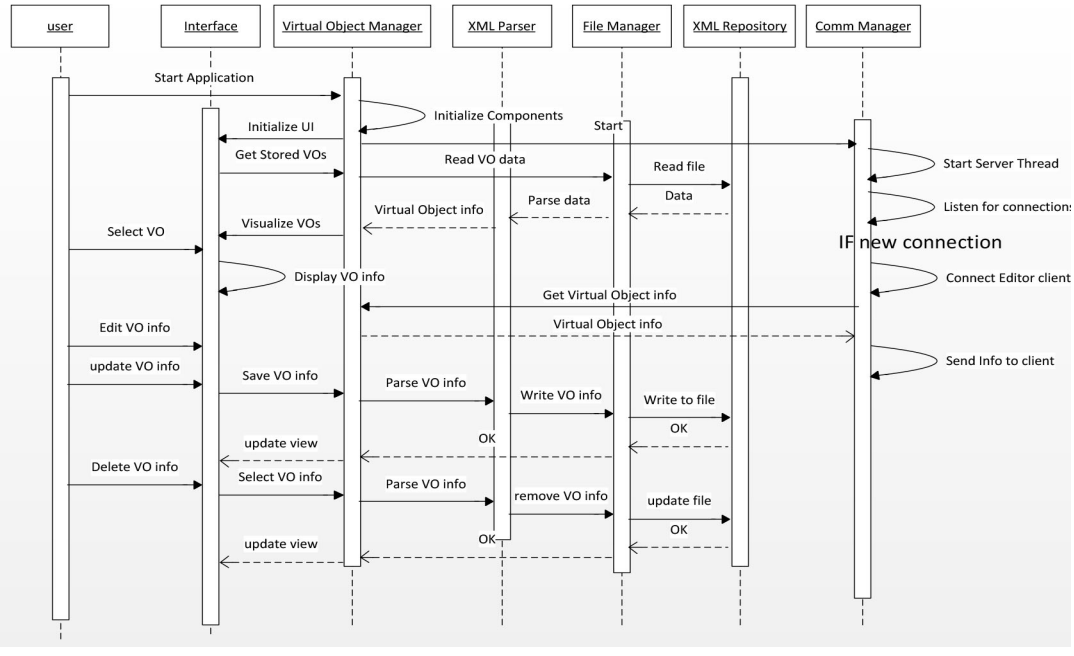

Figure 3: Virtual object manager operation sequence

The main interface provides a view for all the existing VOs so it requests the File Manager through the VOM to read the VO data from the XML repository. The data is parsed by the XML parser and the virtual object information is provided to the VOM in order to display it through the interface. Now the user is set to interact with the VOs through the interface. The VO related interactions that user can perform have been shown in the sequence model. The user selects a VO graphical representation and the VO information is displayed to the user through the view interface. The user can then choose to Edit and Update the VO if needed be. To save an edited VO, the information from the view interface is sent to the parser to convert it into proper format and the File Manager writes it to the XML Repository. The Delete Operation also works in the same fashion.

The Communication Manager acts as a server thread which listens for incoming connections from the remote client (SCM). Once is connection request is received, the

Communication Manager requests the VOM for VO information which is sent to the client.

3.2 Service Composition

Fig. 4 shows the static structure of the Service Composition Manager. It provides the overview of the main classes and the relationship, associations among these classes. The Form, TabControl and TabPage are the .Net built-in classes which act as displayable window and containers for visual controls respectively. The DeviceModule class provides the implementation of virtual representation for the input and output virtual objects. This is shown by the specialization relationship between the InputModule, OutputModule and the DeviceModule. The actual classes representing input devices such as a Pressure Sensor or an output device such as an LED are derived from the InputModule and OutputModule classes respectively. Each of these input or ouput device representation classes have associated custom attributes. The DeviceModule class implements the IDeviceModule interface for the implementation of core properties and methods related to devices modules. It also implements the ICloneable interface for making the virtual devices clone-able. This interface is used to clone the selected module when the user drags a module on the canvas.

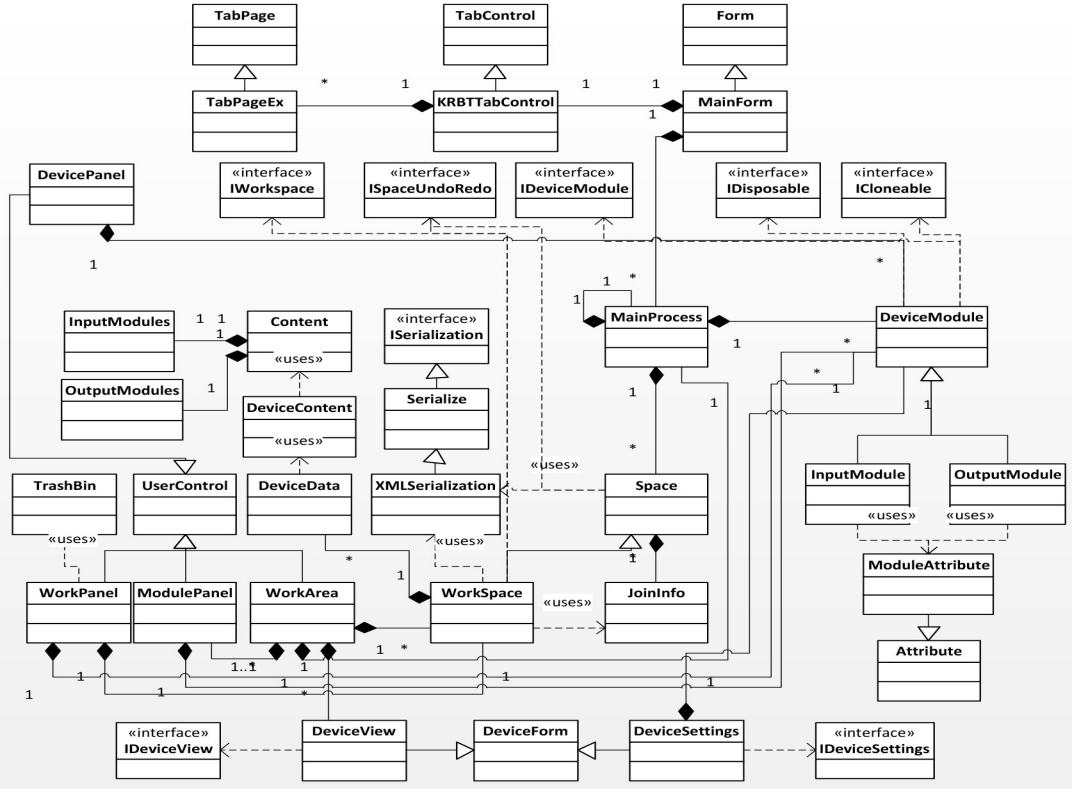

Figure 4: SCM static structure

Each device (VO) module such as the LED class has associated view and settings classes in the form of LEDView and LEDSettings classes. These classes are specializations from the DeviceView and DeviceSettings classes. The DeviceView class is associated with the WorkArea class to show the properties of a selected module in the form of Detail view tab in the editor. Similarly, the DeviceSettings class is a form for setting the properties or parameters and it is shown when a module is double-clicked in the editor's work area.

MainProcess class acts as the main back end process and it is implemented as a singleton class. All the other classes use the same instance of the MainProcess through a public interface. It maintains a list of DeviceModule class and WorkSpace class. Space class is super class of the Workspace class and it uses XmlSerialization class for the conversion of objects to XML data for storage purposes in the memory as well as the file system. The Space class implements the IWorkspace and IspaceUndoRedo interfaces which contains the interfaces related to the storage of data space and maintenance of the current space by providing Undo and Redo functions respectively.

In order to maintain information about the joins created between the input and output modules drawn in the functionality editor, the Space class has a list of JoinInfo class. The same JoinInfo class list is used by the Workspace class to know the joins associated with the current project. Similarly, each WorkSpace object has an object of the DeviceData class which uses the DeviceContents and the Contents class for representing the input and output modules drawn on the work area of a given service composition project. The DevicePanel, WorkPanel and WorkArea classes are derived from UserControl class. These classes are used to provide the graphical user interface for individual projects which are displayed as objects of the TabPageEx class as tabs in the KRBTabControl as an extended for of the TabControl class. TabPageEx is an extended version of the TabPage class which provides a close-able tabpage. The KRBTabControl is part of the MainForm class which is the main displayable container for visual controls and components. The Trash class is a graphical representation of a waste bin which works with the WorkPanel class to provide the functionality for deleting a module drawn on the work area of the editor.

Fig. 5 shows the sequence of steps for designing or composing a service flow using the service composition manager. The user starts the main GUI first and then creates a new project. The first part of the figure shows the sequence of interaction among various internal components of the service composition manager when the user initiates a new project. This sequence does not show the initialization of the MainProcess. It is assumed that the editor is already initialized and the FrmMain container is already displayed on the screen where the user can click the new project button to start the process shown in this figure. As the user clicks the new project button on the FrmMain, it calls the createWorkArea() function and it in turn sends CreateSpace() message to the MainProcess. The MainProcess then creates an object of the WorkSpace class and returns it back to the FrmMain. The FrmMain then adds this new WorkSpace object to the spaces

collection of the MainProcess and further creates an object of the WorkArea class by passing the newly created WorkSpace object in the message. The WorkArea class has an associated WorkPanel object which actually acts as the drawing canvas for the SCM. It also creates the input and output panels for displaying the device module blocks that will be used by the user to drag-n-drop to the work area for creating the service design. To display this WorkArea object on the FrmMain, it is added to the controls collection of an extended tabpage object called as TabPageEx. This tabpage object is then displayed as a new tab on the tabcontrol and all the toolbar controls are setup for the new project using the enableControls message. At this point the user is displayed with the new project tab where he/she can drag and drop virtual objects to create the functionality flows. Once a new project is initialized, the user can start to "drag n drop" input and output VO onto the work area. As the work area has an associated WorkPanel control, the graphical representations for each VO dropped by the user is drawn on the panel.

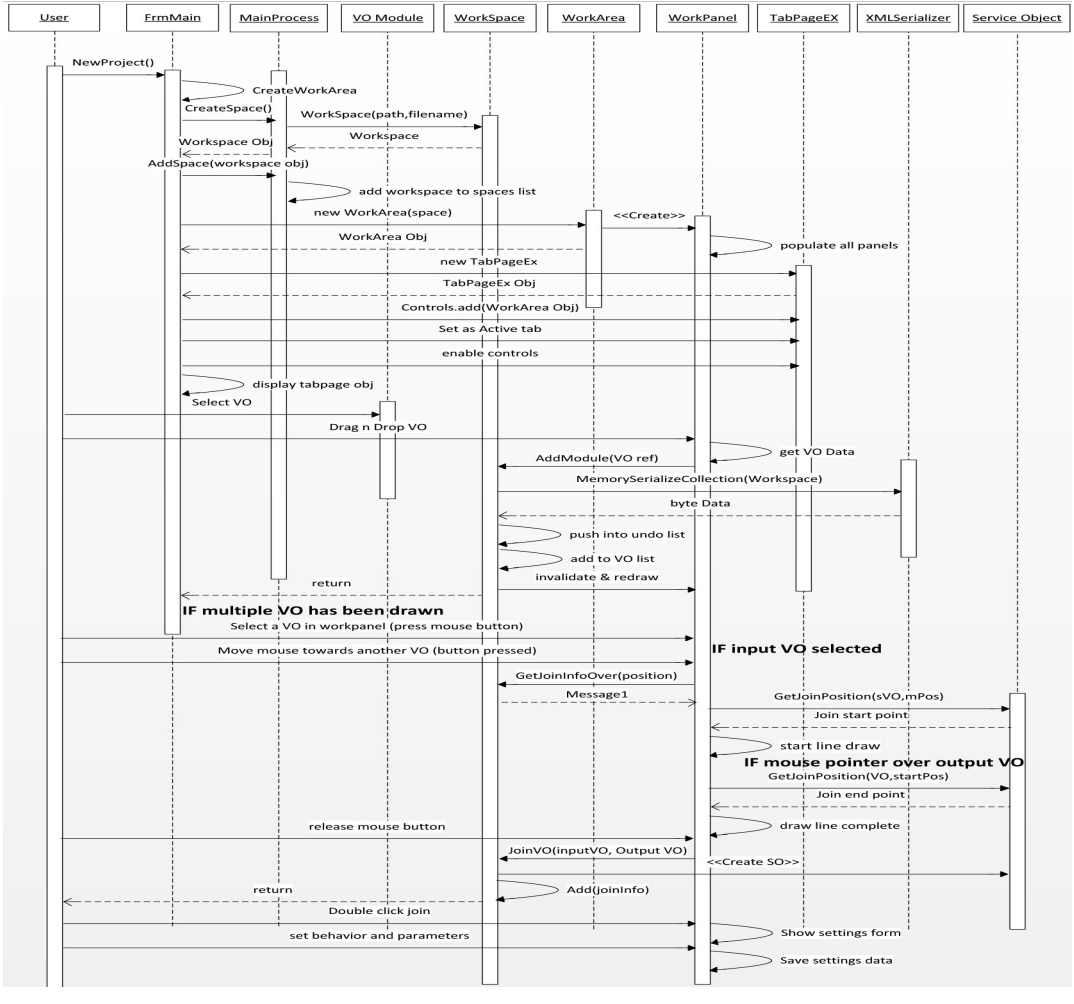

Figure 5: VO to SO mapping sequence at SCL

With each "drag n drop" on the panel, an event handler is executed which get the associated data of the dropped VO and creates a clone object from original one saved at the devices list at MainProcess. The clone object is then added to the Devices list maintained at each Workspace via the parent class Space. The parent class also has stack implementations for maintaining the Undo and Redo operations.

The Workspace class then creates an instance of the XmlSerialization class and calls the MemorySerializeCollection method with its own reference as parameter. The XmlSerialization class gets all the data associated with the Workspace object, converts it to XML format and saves it in a memory buffer as byte data. The reference to the byte data buffer is returned to the Workspace class. At this point, the byte data buffer is pushed into the Undo stack, the device list is updated and the WorkPanel is invalidated in order to draw the updated flow. This sequence is repeated for every new device module dropped onto the WorkPanel by the user. One thing is to be noted that each device module can be dragged to the WorkPanel only once in a project. This is an initial policy for the simplicity of the created flows and may be changed later on.

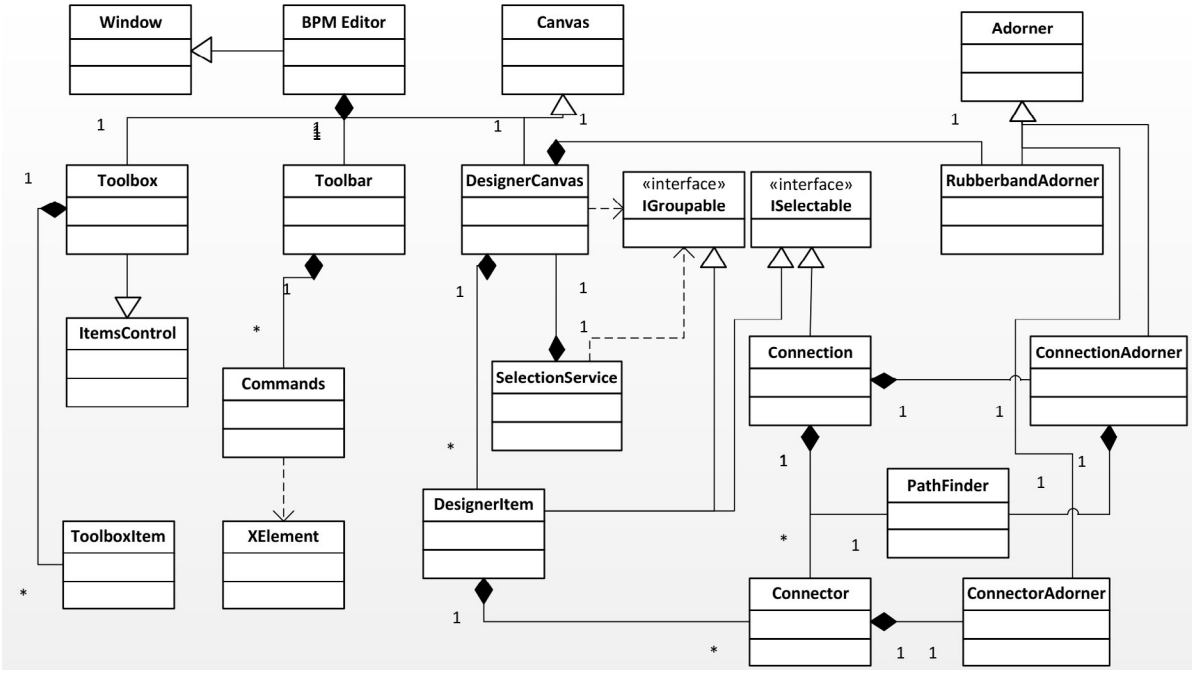

Figure 6: Static structure for business process design manager

The user can join an input device module to any number of output modules. As shown in the sequence, the user has to click and press the left mouse button on an input VO that is already drawn on the WorkPanel. If the user moves the mouse while the left button is pressed, the WorkPanel calls a static method of the JoinInfo class to get the starting

position from which the join should be drawn. To draw the join, a Bezier line is drawn from the starting point of the join to the mouse pointer. If the mouse pointer enters an output VO area, the static method is called again to calculate the end position of the join and the join is displayed on the WorkPanel. If the user releases the mouse button at this moment then the joinInfo object is created with the input and output device modules' information and the object is added to the Joins list maintained by the Workspace object. Otherwise, the drawn line is deleted. Once a join is created, the user can double click the join or the individual VO to set the behavior i.e. select an available function for the associated physical thing, and set other parameters. This data is saved as part of the SO in the form of JoinInfo and thus a complete SO is generated by combining input and output VOs.

3.3 Process Modeling

Fig. 6 shows the static structure of the main components at the business process layer. As the aim of the business process layer is to utilize the service objects created at the service composition layer and present them to the user in the form of business process modelling notations, the most important component at this layer is the business process design manager. It includes a BPM editor which is the main window containing the Toolbar, the Toolbox and the DesignerCanvas. The Toolbar is the component panel which is derived from the itemControl class. This panel is populated by the visual representations of the business process modeling notations in correspondence with the service objects acquired from the service composition layer.

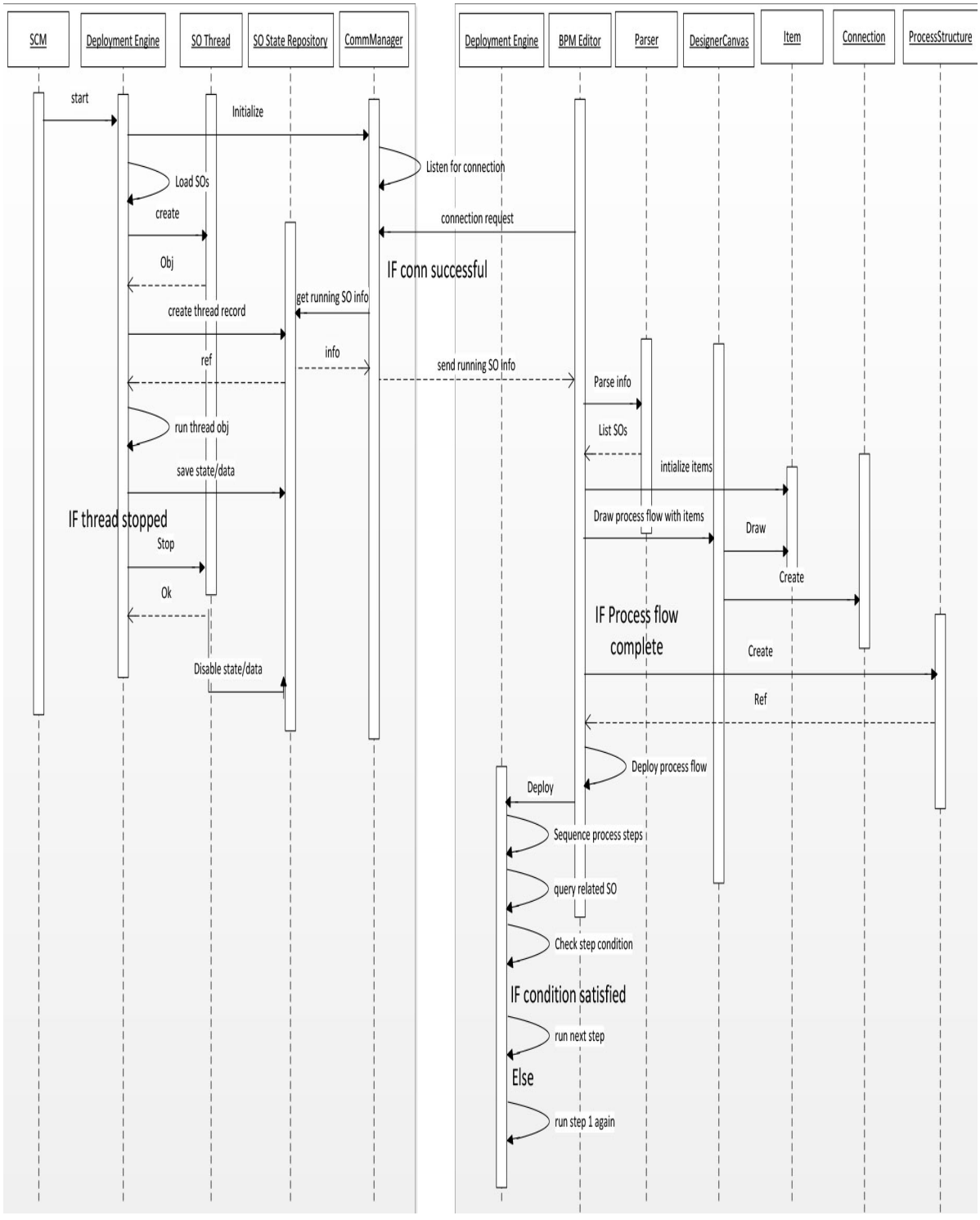

Figure 7: SO to process mapping and execution

The Toolbar component implements the basic drawing commands and operations which are required by a user for performing drag and drop designing. These commands include do, undo operations, copy, paste operation, group and ungroup operations etc. The connection class and its associated ConnectionAdorner class are used to draw connecting lines between the BPMN items representing the sequence of flow among process steps.
Fig. 7 shows the sequence of operation at the business process layer. The operations at business process layer include the acquisition of SOs from the service composition layer, mapping them into business process modeling notations for the user to convert them into a process model and finally to execute the process. The figure shows that the SCM communication manager listens for connection requests from the BPM editor, the main component at the business process layer. Once the connection is successfully established, the available Service Objects (SOs) from the SO repository at the service composition layer are acquired in the form of XML info objects and sent to the business process layer. The acquired SOs are then parsed through an XML parser presented to the user as business process modeling notations. The user then creates the process model by visual drag and drop operations applied to the visual representations of business process model notations. The user draws the process components and connects them via the connection notation along with the operational rules or conditions applied for the transitions between steps. When the process model is complete, a process object is created. The process object is a deployable entity which is deployed via the Deployment Engine at the business process layer.

The deployed process object is converted into a sequence of operations based on the services which compose the service objects of the said process. The deployment engine interacts with the services as part of the process object. The data is acquired from the services, evaluated against the user set conditions/rules and the actuating services are executed as a result. The process object is run as a separate thread so multiple process objects can be executed simultaneously.

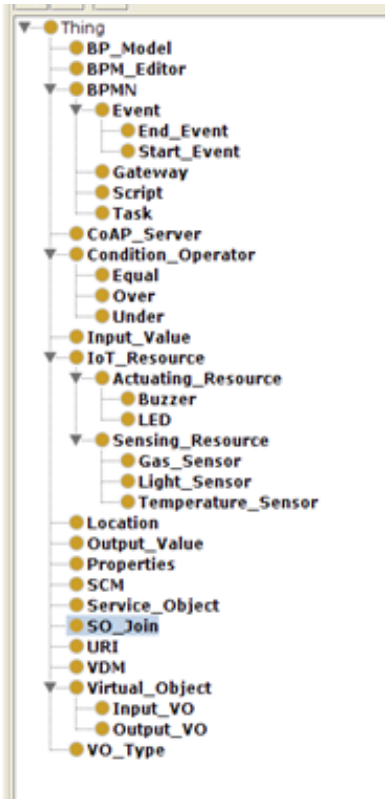

Figure 8: Semantic entities as part of the IoT application composition system

4. Semantic Modeling

This section provides the semantic modeling for the proposed system. The semantic models are used to allow a system to understand various concepts in the system's operation and enable the system to infer extra knowledge by reasoning based on the provided information and rules. Given below is a protégé based implementation of the conceptual semantic entities and the relation among those entities.

4.1. Protégé Implementation

Thing is the base concept in the implementation of Internet of Things as well as in the field of semantics. Fig. 8 shows the core entities of the proposed system as part of the semantic model which is derived from the based class 'Thing'. The semantic entities provide basic information in terms of specialization and generalization of entities. The relations are normally represented as 'is-a' relation.

.

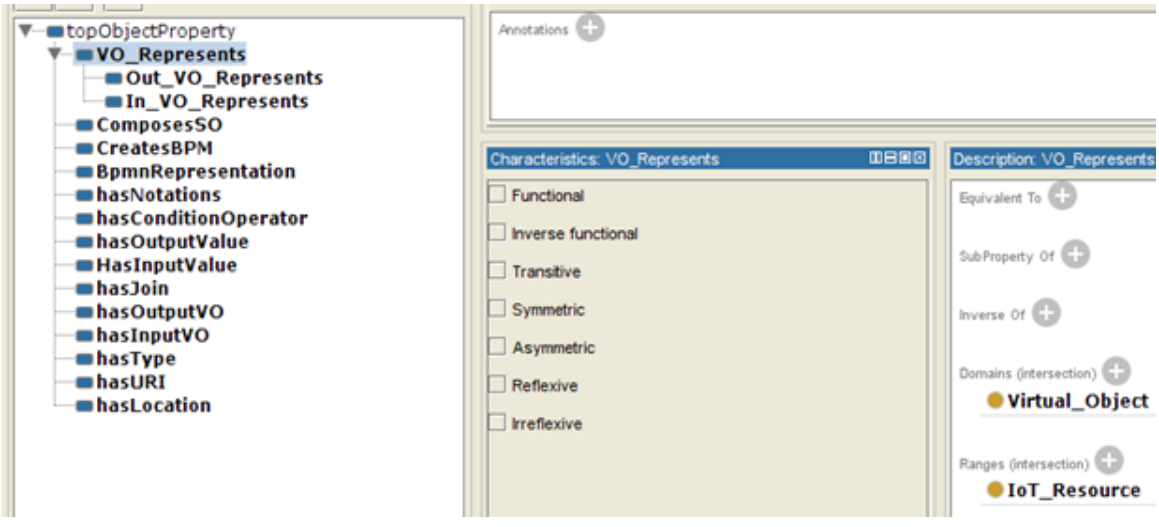

Figure 9: Object properties specifying semantic relations between

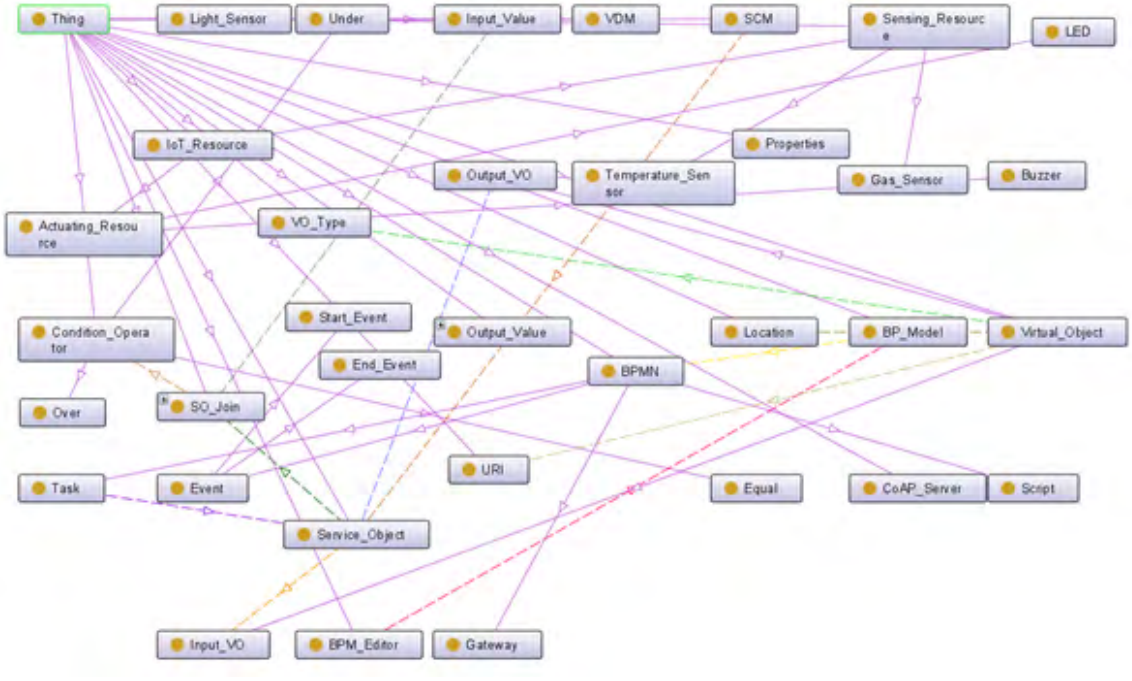

Figure 10: Protégé based semantic model for the proposed system

Fig. 9 provides a snapshot of the protégé object property tab. A list of the object properties defined based on the semantic entities has been shown on the left of the figure. These properties define relationships between various entities of the semantic model by specifying these entities as the domain and range of the properties. For example, the

VO_Represents object property specializes into two properties for the input and output VO concepts. The domain and range for generalized property has been specified to be the Virtual_Object and IoT_Resource entity respectively, providing the idea that a virtual object is a representation of an IoT resource. The specialization of the same object property then further divides this association between the input and output resources
The semantic model is the combination of semantic entities, relations among those entities and inference rules. Fig. 10 shows the semantic model for the proposed system as visualized by the protégé. Although the semantic model is still incomplete and further work is being done, it can provide semantic reasoning capabilities to the proposed system and based on the semantic inference, the system would be able to provide useful suggestions to the user while composing services as well as creating IoT applications through the BPM Editor.

4. Discussion:

The main focus of the architecture is the enablement of the end-users, with no or limited programming skills, to compose services/applications as per their own requirements with the help of the generic BPM notations. It is a layered architecture, which means that each layer can act on its own and only communicates with the upper or lower layers via well-defined interfaces. The layered architecture provides separation of concerns and avoid overburdening the constraint IoT devices. IoT devices exposes their functionalities as atomic services which are represented as virtual objects. These virtual object representations can then be grouped at the virtual object layer which makes the architecture scalable and the same can be achieved with the upper layers. The architecture can efficiently handle the security issues as well with the help of the separation of concerns. No user except the owner of the physical device is allowed to directly access and/or modify it. The users can only access virtual objects and service objects from different layers to fulfil their requirements and ultimately compose their own applications/services with the help of a generic drag and drop mechanism.
The only drawback of the layered architecture as evident from relevant literature is the performance degradation. When a message needs to pass through multiple layers in order to get across to the intended receiver, the extra translations and conversions at each layer degrades the overall performance. The future direction of the study is develop a prototype based on the presented architecture and test the performance at each layer and the performance of the composed services/applications.

5. Conclusion

This paper presented the idea of the utilization of business process modeling approach as a DIY tool for the IoT end users to design and execute IoT processes at IoT infrastructure. The IoT infrastructure may be in the form of a localized smart home or a more distributed implementation in the form of an agricultural IoT system. The paper visualizes the presented idea in the form of a layered architecture which consists of the physical layer,

service composition layer and the business process layer. A detailed description of the layered architecture and design detail of the layers has been presented in this paper. A brief description of the semantic model has also been presented. The development of the system is underway and a prototype implementation based on CoAP devices and services will be completed in the next phase of the development.

Acknowledgements

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.10043907, Development of high performance IoT device and Open Platform with Intelligent Software). And this research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1017) supervised by the IITP (Institute for Information & communications Technology Promotion), Corresponding author; DoHyeun Kim (e-mail: kimdh@jejunu.ac.kr).

Author Biographies

Muhammad Sohail Khan received his Bachelor of Engineering and Master's degrees from Computer Software Engineering Department, University of Engineering and Technology Peshawar in 2008 and 2012 respectively. Meanwhile, he had been a part of the software development industry in Pakistan as a designer and developer. From 2010 onwards, he has been working as a faculty member at the Department of Computer Software Engineering, UET Peshawar, Pakistan. In August 2016 he received his Ph.D. degree from Jeju National University, South Korea. The focus of his work is the application of software design strategies towards enablement of mass involvement in IoT application/service development through intuitive DIY development environments.

Do Hyeun Kim received the B.S., M.S. and P.D degrees in Electronics Engineering from Kyungpook National University, Taegu, Korea, in 1988 and 1990, 2000 respectively. He joined the Agency of Defense Development (ADD), Korea, in 1990. Since 2004, he is currently a professor at the Department of Computer Engineering at Jeju National University, Korea. His research interests include sensor web, optimization algorithm and context prediction.

Faiza Tila received the B.Sc. degree in Computer Software Engineering from University of Engineering and Technology Pakistan in 2012. She joined the Mobile Computing lab Jeju National University as a M.S. student in 2014. Her area of interest is Semantic Web Technologies and Internet of things.

References

[1] S. Sinha, “State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion” 2021. [Online] Available: https://iot-analytics.com/number-connected-iot-devices/#:~:text=In%20 2021 %2C%20IoT20Analytics%20expects,than%2027%20billion%20IoT%20 connections. [Accessed: 25-April 2022].

[2] V. Alpha, O. M. G. D. Number, and P. D. F. A. File, “BPMN 2.0 by Example,” vol. 8, no. June, 2010.

[3] N. Niknejad, I. Waidah, I. Ghani, B. Nazari, and M. Bahari, "Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation." Information Systems 91 (2020): 101491.

[4] E. Schäffer, V. Stiehl, P. K. Schwab, A. Mayr, J. Lierhammer, and J. Franke. "Process-driven approach within the engineering domain by combining business process model and notation (BPMN) with process engines." Procedia CIRP 96 (2021): 207-212.

[5] P. Harmon and C. Wolf, “State of Business Process Management – 2020,” 2020.

[6] M. R. Faheem, T. Anees, and M. Hussain. "The web of things: findability taxonomy and challenges." IEEE Access 7 (2019): 185028-185041.

[7] S. Kumar, P. Tiwari, and M. Zymbler. "Internet of Things is a revolutionary approach for future technology enhancement: a review." Journal of Big data 6, no. 1 (2019): 1-21.

[8] H. Garg, and M. Dave. "Securing iot devices and securely connecting the dots using rest api and middleware." In 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), pp. 1-6. IEEE, 2019.

[9] M. Dave, J. Doshi, and H. Arolkar. "MQTT-CoAP Interconnector: IoT Interoperability Solution for Application Layer Protocols." In 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 122-127. IEEE, 2020.

[10] B. M. Presser, P. M. Barnaghi, U. Kingdom, M. Eurich, and C. Villalonga, “G l o b a l The SENSEI Project : Integrating the Physical World with the The IEEE ComSoc Sister Society in India : The Institution of Electronics and Telecommunication Engineers,” no. April, pp. 1–4, 2009.

[11] T. S. LOPEZ, D. KIM, G. H. Canepa and K. Koumadi, “Integrating Wireless Sensors and RFID Tags into Energy-Efficient and Dynamic Context Networks,” vol. 52, no. 2, 2009.

[12] S. Meyer, A. Ruppen, and L. Hilty, "The Things of the Internet of Things in BPMN," Adv. Inf. Syst. Eng. Work., vol. 215, pp. 285–297, 2015.

[13] M. Bauer, M. Boussard, N. Bui, and F. Carrez, "Project Deliverable D1.2 – Final Architectural Reference Model for IoT," no. 257521, pp. 53–59, 2013.

[14] M. G., "Integrating the Internet of Things with business process management: A process-aware framework for Smart Objects," CEUR Workshop Proc., vol. 1415, pp. 56–64, 2015.

[15] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Comput. Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[16] K. Gama, L. Touseau, and D. Donsez, "Combining heterogeneous service technologies for building an Internet of Things middleware," Comput. Commun., vol. 35, no. 4, pp. 405–417, Feb. 2012.

[17] L. Atzori, A. Iera, and G. Morabito, "From 'smart objects' to 'social objects': The next evolutionary step of the internet of things," IEEE Commun. Mag., vol. 52, no. 1, pp. 97–105, Jan. 2014.

[18] C. Anderson, "Makers: The New Industrial Revolution," Compet. Rev., vol. 24, no. 2, pp. 147–149, Mar. 2014.

[19] J. G. Tanenbaum, A. M. Williams, A. Desjardins, and K. Tanenbaum, "Democratizing technology," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13, 2013, p. 2603.

[20] D. Carrera, "Collaborative Open Market to Place Objects at your Service," pp. 1–65, 2014.

[21] IDC, "Worldwide Internet of Things Spending Guide". 2022 [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=IDC_P29475. [Accessed: 25-April 2022].