



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

	<p>Enhanced IoT Composition Architecture based on DIY Business Process Modeling Approach</p> <p>Sohail Khan</p> <p>2016</p>	
--	---	--

A Thesis

For the Degree of Doctor of Philosophy

Enhanced IoT Composition Architecture based on DIY  
Business Process Modeling Approach

Muhammad Sohail Khan

Department of Computer Engineering

Graduate School

Jeju National University

June 2016

# Enhanced IoT Composition Architecture based on DIY Business Process Modeling Approach

Muhammad Sohail Khan  
(Supervised by: Professor Do-Hyeun Kim)

A thesis submitted in partial fulfillment of the requirements for the degree  
of Doctor of Philosophy in Computer Engineering.

2016. 06.

This thesis has been examined and approved.

---

Thesis Committee Chair  
Sang-Joon Lee, Professor, Jeju National University

---

Thesis Committee Member  
Yun-Jeong Lee, Professor, Jeju National University

---

Thesis Committee Member  
Jeong-Won Jo, Professor, Jeju National University

---

Thesis Committee Member  
Seong-Dong Kim, Korean Electronics Technology Institute

---

Thesis Supervisor,  
Do-Hyeun Kim, Professor, Jeju National University

Department of Computer Engineering  
Graduate School  
Jeju National University

*Dedicated to my parents and my  
family for their love and support.*

# Acknowledgment

All praise to *Allah Almighty* for bestowing upon me his countless blessings. He became my strength when I felt weakened, He gave me courage when I felt defeated, and he blessed me with the patience to finally achieve success in this huge endeavor.

I would like to express my sincere gratitude to my supervisor Prof. Do-Hyeun Kim for his continuous support of my Ph.D. study and related research, for his patience, motivation, and encouraging attitude. He has not only been an excellent scientific mentor, but is a great human being. Besides my advisor, I would like to thank my thesis evaluation committee for their insightful comments and valuable suggestions during the thesis defense. Their input helped me in elevating the quality of this thesis. I really appreciate the support and encouragement given to me by my former lab mates and dear friends Dr. Rashid Ahmad and Dr. Safdar Ali. I would like to mention my current lab mates Jin Wenquan, Ajaya, Hang le, Israr Ullah and Muhammad Fayaz for their kind support and help during this endeavor.

Words cannot express my gratitude to my mother who won't even understand a word of this thesis but nevertheless prayed endlessly for my success. I wish I could celebrate this huge achievement with my late father whose selfless efforts made me what I am today. I am deeply thankful to my siblings for their support, prayers and love towards my success. Finally, I would like to appreciate the undiminishing and unconditional support, love and care of my lab mate and my life mate Faiza Sohail Khan for being there for me each and every step of this journey. Without her support and encouragement I would not have achieved my goals.

**Muhammad Sohail Khan**

# Table of Contents

<b>Acknowledgment</b>	<b>i</b>
<b>Table of Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>vii</b>
<b>Abbreviations</b>	<b>viii</b>
<b>Abstract</b>	<b>1</b>
<b>1. Introduction</b>	<b>4</b>
<b>2. Related work</b>	<b>15</b>
2.1. Existing IoT composition research .....	15
2.2. Issues with the existing solutions from the DIY perspective .....	21
2.3. Existing IoT protocols.....	24
<b>3. Proposed DIY IoT System Architecture</b>	<b>27</b>
3.1. Virtual Object Layer.....	27
3.2. Service Composition Layer .....	29
3.3. Business Process Layer .....	31
3.4. Development Process .....	32
3.5. Data representations .....	34
<b>4. BPM based DIY IoT System Design</b>	<b>36</b>

4.1.	Virtual Object Layer.....	36
4.2.	Service Composition Layer .....	40
4.3.	Business Process Layer .....	47
5.	<b>BPM Based DIY IoT System Implementation</b>	<b>57</b>
5.1.	Virtual Object Manager .....	57
5.2.	Service Composition Manager .....	59
5.3.	BPM Editor .....	63
5.4.	BPM Deployment Manager .....	67
5.5.	Performance Analysis .....	70
6.	<b>Usability Study for Robotic Arm Use-case</b>	<b>75</b>
6.1.	IoT in Industrial Robotics .....	75
6.2.	Implementation architecture for robotic arm use-case .....	76
6.3.	Robotic Arm prototype implementation .....	78
6.4.	Usability study for Robotic Arm use-case.....	81
6.4.1.1.	SCM usability assessment for Robotic Arm use-case .....	84
6.4.1.2.	SCM usability score based on SUS .....	87
6.4.1.3.	SCM usability from DIY perspective .....	88
6.4.2.	BPM Editor usability assessment for Robotic Arm use-case .....	93
6.4.2.1.	BPM Editor usability score based on SUS .....	96
6.4.2.2.	BPM Editor usability from DIY perspective .....	97
7.	<b>Usability Study for Smart-Space use-case</b>	<b>105</b>



<b>7.1. IoT enabled Smart-Spaces.....</b>	<b>105</b>
<b>7.2. Implementation architecture for smart-space use-case .....</b>	<b>106</b>
<b>7.3. Smart-space prototype implementation .....</b>	<b>109</b>
<b>7.4. Usability study for Smart-Space use-case .....</b>	<b>112</b>
<b>7.4.1. SCM usability assessment.....</b>	<b>114</b>
<b>7.4.1.1. SCM usability score based on SUS .....</b>	<b>116</b>
<b>7.4.1.2. SCM usability from DIY perspective .....</b>	<b>118</b>
<b>7.4.2. BPM Editor usability assessment .....</b>	<b>123</b>
<b>7.4.2.1. BPM Editor usability score based on SUS .....</b>	<b>125</b>
<b>7.4.2.2. BPM Editor usability from DIY perspective .....</b>	<b>127</b>
<b>8. Conclusion</b>	<b>135</b>
<b>References</b>	<b>137</b>

# List of Figures

<b>Figure 1: Comparison of the existing and proposed DIY IoT architecture</b> .....	10
<b>Figure 2: Generic architecture of the Glue.Things project</b> [28] .....	16
<b>Figure 3: Conceptual architecture of IoT MAP</b> [30] .....	17
<b>Figure 4: High-level architecture of the IoTLink</b> [32] .....	18
<b>Figure 5: Layered architecture of the SSC platform</b> [33] .....	18
<b>Figure 6: High-level architecture of Ambient Flow</b> [34] .....	19
<b>Figure 7: BPM based DIY IoT composition system architecture</b> .....	28
<b>Figure 8: Development process through DIY IoT composition system</b> .....	33
<b>Figure 9: Metadata representation at each layer of the system</b> .....	35
<b>Figure 10: Virtual Object Manager startup and operational configuration</b> .....	37
<b>Figure 11: Static structure for Virtual Object Manager</b> .....	38
<b>Figure 12: Virtual Object Manager operation sequence</b> .....	39
<b>Figure 13: Service composition Manager basic configuration</b> .....	41
<b>Figure 14: Service Composition Manager static structure</b> .....	43
<b>Figure 15 : VO to SO mapping sequence at SCL</b> .....	46
<b>Figure 16: BPM Editor startup and operational configuration</b> .....	49
<b>Figure 17: Static structure for BPM Editor</b> .....	51
<b>Figure 18: SO to process mapping and execution</b> .....	52
<b>Figure 19: BPM Deployment Manager startup and operational configuration</b> .....	53
<b>Figure 20: BPM based DIY IoT application development system collaboration from the BPL perspective</b> .....	55
<b>Figure 21: Virtual Object Manager Interface</b> .....	58
<b>Figure 22: XML representations for virtual object storage at VOL</b> .....	59
<b>Figure 23: Service composition manager interface</b> .....	60
<b>Figure 24: XML representation of service objects at SCL</b> .....	62
<b>Figure 25: BPM Editor interface</b> .....	64
<b>Figure 26: XML representation of a stored BPM model</b> .....	66
<b>Figure 27: Deployment manager interface with a loaded BPM</b> .....	68
<b>Figure 28: Deployment manager execution results</b> .....	69
<b>Figure 29: Performance analysis graph at SCM</b> .....	71
<b>Figure 30: Performance analysis graph for BPM Editor</b> .....	72
<b>Figure 31: Performance analysis graph for BPM Deployment Manager</b> .....	73
<b>Figure 32: Robotic arm use-case implementation in the proposed architecture and modifications for usability study</b> .....	77
<b>Figure 33: Intel Edison based finalized robotic arm prototype</b> .....	79
<b>Figure 34: Test scenario for usability assessment of robotic arm use-case</b> .....	82
<b>Figure 35: Flow of SCM usability assessment experiment for robotic arm use-case</b> .....	85
<b>Figure 36: Results of SCM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Robotic arm use-case)</b> .....	87

<b>Figure 37: Sample comparison of the time taken by both groups to complete SCM task without any prior training.....</b>	<b>89</b>
<b>Figure 38: Comparison of percentage for successful task completion by Programmers vs Non-Programmers .....</b>	<b>89</b>
<b>Figure 39: Sample comparison of the time taken by both groups to complete SCM task after the training session.....</b>	<b>91</b>
<b>Figure 40: Flow of BPM Editor usability assessment experiment for robotic arm use-case .....</b>	<b>94</b>
<b>Figure 41: Results of BPM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Robotic arm use-case) .....</b>	<b>96</b>
<b>Figure 42: Sample comparison of the time taken by both groups to complete BPM task without any prior training (Robotic arm use-case).....</b>	<b>98</b>
<b>Figure 43: Comparison of percentage successful completion of simple BPM task by Programmers vs Non-Programmers without any prior training (Robotic arm use-case) ..</b>	<b>99</b>
<b>Figure 44: Sample comparison of the time taken by both groups to complete BPM task after training (Robotic arm use-case).....</b>	<b>101</b>
<b>Figure 45: Sample comparison of the time taken by both groups to complete BPM complex task (Robotic arm use-case).....</b>	<b>103</b>
<b>Figure 46: Comparison of percentage successful completion of BPM complex task by Programmers vs Non-Programmers (after training).....</b>	<b>104</b>
<b>Figure 47: Smart space use-case implementation in the proposed architecture and modifications for usability study.....</b>	<b>107</b>
<b>Figure 48: Intel Edison based finalized Smart-Space prototype .....</b>	<b>109</b>
<b>Figure 49: Test scenario for usability assessment of smart space use-case.....</b>	<b>113</b>
<b>Figure 50: Flow of SCM usability analysis experiment for smart space use-case.....</b>	<b>115</b>
<b>Figure 51: Results of SCM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Smart space use-case).....</b>	<b>117</b>
<b>Figure 52: Sample comparison of the time taken by both groups to complete SCM task without any prior training (smart space use-case).....</b>	<b>118</b>
<b>Figure 53: Percentage for successful SCM task completion by Programmers and Non-Programmers (Smart space scenario) .....</b>	<b>119</b>
<b>Figure 54: Sample comparison of the time taken by both groups to complete SCM task after the training (smart space use-case).....</b>	<b>121</b>
<b>Figure 55: Flow of BPM Editor usability experiment for smart space use-case .....</b>	<b>123</b>
<b>Figure 56: Results of BPM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Smart space use-case).....</b>	<b>126</b>
<b>Figure 57: Sample comparison of the time taken by both groups to complete BPM simple task without any prior training (Smart space use-case) .....</b>	<b>127</b>
<b>Figure 58: Comparison of percentage successful completion of simple BPM task by Programmers vs Non-Programmers without any prior training (smart space use-case)..</b>	<b>128</b>
<b>Figure 59: Sample comparison of the time taken by both groups to complete BPM simple task after the training (Smart space use-case).....</b>	<b>129</b>
<b>Figure 60: Sample comparison of the time taken by both groups to complete BPM complex task (smart space use-case).....</b>	<b>132</b>

**Figure 61: Comparison of percentage successful completion of BPM complex task by  
Programmers vs Non-Programmers (smart space use-case) ..... 132**

# List of Tables

<b>Table 1: System configuration for performance analysis</b> .....	70
<b>Table 2: Device implementation summary for robotic arm use-case</b> .....	80
<b>Table 3: System Usability Scale items</b> .....	83
<b>Table 4: Descriptive Statistics of SUS Scores for Adjective Ratings [55]</b> .....	84
<b>Table 5: SCM usability assessment experimental setup based on robotic arm use-case</b> .....	86
<b>Table 6: Two-Sample t-Test for unequal variances (Activity 1 before training)</b> .....	90
<b>Table 7: Two-Sample t-Test for unequal variances (Activity 1 after training)</b> .....	92
<b>Table 8: BPM Editor usability assessment experimental setup based on robotic arm use-case</b> .....	95
<b>Table 9: Two-Sample t-Test for unequal variances (Simple BPM task without prior training, robotic arm use-case)</b> .....	100
<b>Table 10: Two-Sample t-Test for unequal variances (Simple BPM task after training)</b> ...	102
<b>Table 11: Two-Sample t-Test for unequal variances (Complex BPM task)</b> .....	103
<b>Table 12: Input device implementation summary for smart space use-case</b> .....	110
<b>Table 13: Output device implementation summary for smart space use-case</b> .....	111
<b>Table 14: SCM usability assessment experimental setup based on smart space use-case</b> .	116
<b>Table 15: Two-Sample t-Test for unequal variances (Activity 1 before training)</b> .....	120
<b>Table 16: Two-Sample t-Test for unequal variances (Activity 2 after training)</b> .....	122
<b>Table 17: BPM Editor usability assessment experimental setup based on smart space use-case</b> .....	124
<b>Table 18: Two-Sample t-Test for unequal variances (Simple BPM task without prior training, smart space use-case)</b> .....	129
<b>Table 19: Two-Sample t-Test for unequal variances (Simple BPM task after training, smart space use-case)</b> .....	130
<b>Table 20: Two-Sample t-Test for unequal variances (BPM complex task, Smart space use-case)</b> .....	133
<b>Table 21: Mann-Whitney-U Test for non-parametric equivalence (BPM complex task, Smart space use-case)</b> .....	134

# Abbreviations

API	Application Programming Interface
BPL	Business Process Layer
BPM	Business Process Modeling
BPMN	Business Process Modeling Notations
CoAP	Constrained Application Protocol
CQELS	Continuous Query Evaluation over Linked Stream
DIY	Do-It-Yourself
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
MQTT	MQ Telemetry Transport
POJO	Plain Old Java Object
REST	Representational State Transfer
SCL	Service Composition Layer
SCM	Service Composition Manager
SO	Service Object
SOA	Service Oriented Architecture
SPARQL	Simple Protocol and RDF Query Language
SUS	System Usability Scale
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
UED	End-User Development
URI	Uniform Resource Identifier
VO	Virtual Object
VoIP	Voice over IP

VOL	Virtual Object Layer
VOM	Virtual Object Manager
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol)

# Abstract

Tremendous leaps in technological advancements have been taken recently and specifically the advancements in the sensing and communication technologies keep on adding new dimensions to the field of information and communication technologies. These drastic advancements led to vision of a new hyper-connectivity based technological paradigm called as the Internet of Things (IoT). Internet of Things has been the focus of research and development in the recent years and although, the goal of complete realization of the IoT vision is still to be achieved, it is already playing a major role in transforming our social, commercial and personal spheres. The increased attention from academia and industry has resulted in various approaches for the realization of the IoT vision that every physical entity should be a part of global network of things.

For a global adaptation and realization of the IoT vision, one approach is to utilize general population towards the adaptation of IoT vision and for the development of IoT resources and applications. The approach is suggested to be achieved via the representation of real world entities as virtualized entities in the cyber world where the behavior of the virtual objects is exposed as services and it can be accessed and manipulated like real world objects. The Maker movement and the growing number of MakerSpaces all around the world is a testament to the utilization of this approach. The problem with this approach is that most part of the general population lacks the necessary programming skills to utilize the services of these connected objects to create their own applications.

Do-It-Yourself (DIY) paradigm of development is one of the most popular candidate solutions for the general public programming skills problem. State of the art studies have suggested DIY interfaces for their IoT implementations. It has been identified that most of these



implementations are application/domain specific or too complex for a non-technical user to deploy in order to utilize them for the customization/development of IoT applications and services. To bridge this gap between the utilization of masses for IoT development and the programming skill requirements on behalf of the masses, an enhanced IoT composition architecture based on DIY Business Process Modeling approach has been presented in this dissertation.

Business Process Modeling Notations (BPMN) is a standardized graphical language that has been utilized to gap the bridge between non-technical managers and the technical staff since long. This study utilizes the existing concepts of virtual objects and service-orientation to provide an intuitive DIY IoT composition architecture to enable end-users to visualize IoT service objects as standardized graphical notation and interact with them via simple actions like drag-n-drop and clicks etc. Complete design and implementation details for the BPM based DIY IoT composition architecture have been provided along with the performance evaluation of the major functional units. In order to assess the usability of the proposed architecture from the perspective of multi-domain applicability, two separate use-cases have been developed.

The first use-case targets the Industrial Robotics domain which is a growing target for the IoT implementation. A prototype robotic arm has been developed for this purpose. To assess the usability of the proposed architecture, an experimental study has been conducted to allow participant from various age groups and ethnic background to rate the system based on the System Usability Scale. The second use-case is taken from the Smart-Space domain which is another focus of IoT implementations. A prototype scenario of a smart space has been developed to assess the usability of the proposed architecture in the domain.

The experimental study allows participants to use the proposed architecture for service composition and BPM based process development for the prototype smart space and then use the

System Usability Scale (SUS) to rate the system. As the SUS score only indicates a system's usability from users' perspective based on their subjective ratings, thorough statistical analysis has been conducted on data collected during the experiments for both the use-cases to assess the usability of the proposed architecture from DIY perspective. The SUS scoring and the statistical analysis provided positive support for the claims made in this study.

# 1. Introduction

For more than a decade now, the Business Process Modeling (BPM) has been at the center of close collaboration between business and IT. The term 'Business Process Modeling' was coined in the 1960s in the field of systems engineering by S. Williams in his 1967 article 'Business Process Modeling Improves Administrative Control' [1]. His idea was that techniques for obtaining a better understanding of physical control systems could be used in a similar way for business processes. The term became popular in 1990 and since then it has been utilized in almost every business including the recent software engineering industry. It has become an integral part of successful business operations. 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 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 and can also be utilized for process driven applications [4].

The current technologies in the form of sensors and actuators networks, web of things [5] and most of all the realization of the Internet of Things (IoT) [6] 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 introduction of REST [7] and resource constrained protocols such as MQTT [7] and the recent CoAP [8] Protocols, 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.

Research community has embraced the potentials of a resource constrained devices in the paradigm of business process modeling and management, hence efforts are being done to include things and services as part of the BPMN. In this regard, Meyer [9] provided the missing concept of “thing”, as presented by the main components of IoT reference model [10], by extending the conventional meta-models for the Business Process Modeling Notations. Similarly, [11] proposed 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 [12] 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. The concept is also known as Do-It-Yourself (DIY) paradigm and it has been utilized since long for reducing the dependence of common people on proprietary products. DIY is the technique or method for creating, modifying or repairing something without the direct aid of experts or professionals. According to research “individuals engage raw and semi-raw

materials and component parts to produce, transform, or reconstruct material possessions, including those drawn from the natural environment (e.g. landscaping)" [13].

According to Avula [14], in an age where mass production has become a practice, DIY is a way to differentiate your product. From the perspective of a global Internet of Things (IoT) realization, DIY IoT development via the end-users has been suggested by De Roeck [15] and the DIY IoT vision is presented in such a way that end-user is able to create applications for smart environments. The same idea has been supported by Bannon [16] who states that in order for ubiquitous technologies to get full exploration of the design space, the concepts of how technology works and what boundaries there are should be forgotten in the design strategy.

According to the DIY manifesto presented by De Roeck [15], The DIY paradigm of development in the IoT is of special importance because IoT is all about context-aware applications. However, the existing context-aware applications lack the ability to uniquely distinguish the meaning of context for individual users of the applications. To solve this, end-user should be enabled to decide what context means to them in a particular environment. Therefore, DIY development in IoT can prove very effective if users can develop IoT applications according to their own environmental setups.

DIY paradigm of development inspires non-technical people to create things on their own. It eliminates the needs for extensive training. In conventional scenarios, the motivation to achieve training goals sometimes touches the boundaries of harassment. DIY systems capture the attention of its users by offering a playground like environment where the user can utilize and experiment with their own created components or the ones created by others hence increasing the possibility of application development with unconventional thinking and logics.

This idea has been advocated by many such as [12, 13] stating that the end-users should be part of the creation process while having the power to discover things. Similarly, Gamma et al., [36] and Atzori et al., [37] 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 [20], the advancement in DIY prototyping platforms such as Arduino and Raspberry Pi etc. [21] 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.

As IoT envisions a global ecosystem of connected devices providing services to the people, DIY paradigm of development has been suggested as the only solution towards the global realization and implementation of IoT. In the DIY scenario, the same general population which utilizes the services provided by IoT implementation will be able to develop IoT application according to their own needs and requirements. It is however, a concern that the general population lacks the necessary technical experience and specifically the programming skills to develop IoT related applications. This is why alternative development strategies are being investigated to provide people with DIY type development environments through which anyone, regardless of their programming skills level, can develop applications based on their own requirements. The work presented in this thesis is based on the same idea.

The global adaptation and implementation of IoT vision is still far from realization and serious efforts are underway for achieving this goal. However, this global adaptation and implementation cannot be restricted to only governments or technical development

organizations. It has been suggested that mass involvement of general population can prove effective towards the global adaptation of IoT vision. The mass involvement of general public in IoT adaptation means that people should be able to create and customize IoT related services for their own uses. It, however, requires considerable amount of technical skills and knowledge on behalf of the general population to program and customize IoT services.

It is, therefore, necessary to provide the general population with platforms which uses the DIY approach to program and customize IoT services and applications. Following the goal of enabling mass involvement towards the realization of IoT vision, this work proposes the use of standardized Business Process Modeling Notations (BPMN) based enhanced DIY IoT composition architecture. The proposed architecture is aimed at enabling non-technical people to easily and intuitively utilize virtual representations of IoT devices to compose services and to visualize those services as business process modeling notations to graphically model their own IoT applications or customize the existing ones according to their needs and requirements.

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. 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. The concept of DIY development paradigm is to relieve the end-user of any technical complexities associated with application development. This concept proves even more important in the case of IoT application development because the number of IoT enabling technologies is very high and it is still growing with time. In such a scenario if mass involvement of general population with lesser or no technical/programming skills is required, the DIY concept must be implemented in its entirety.

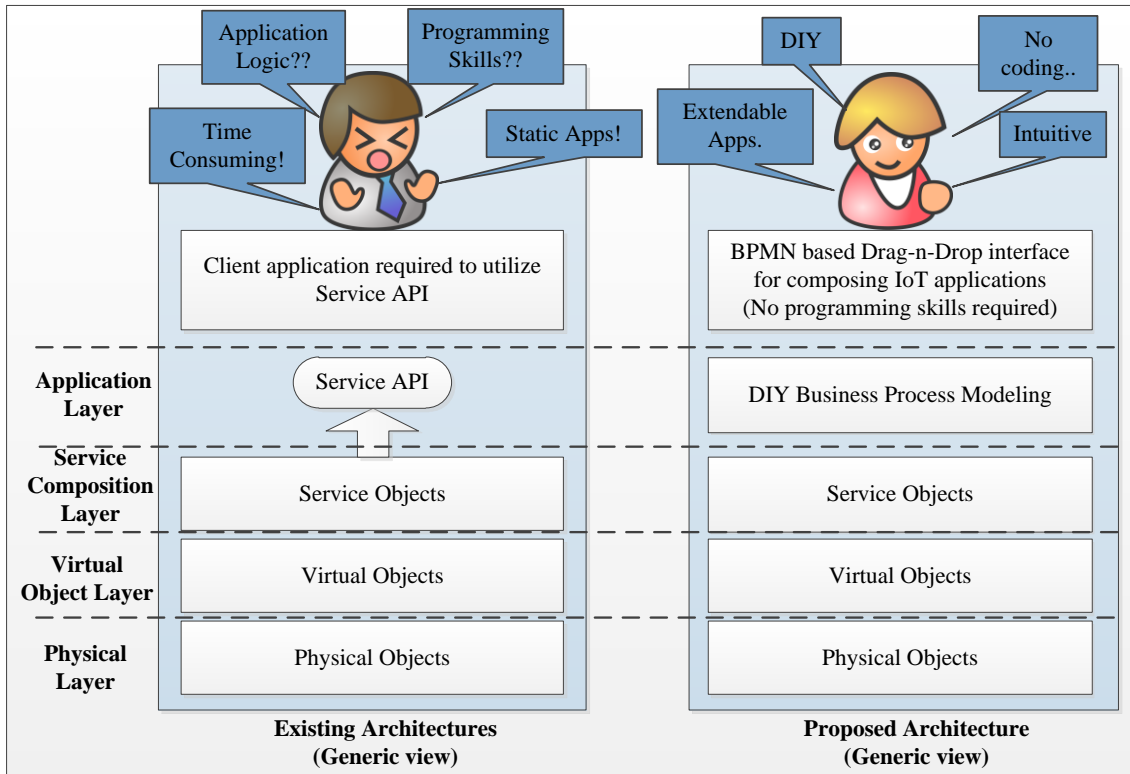
Figure 1 presents a generic comparison between the existing architectures which claims to provide a DIY development environment for IoT applications and the proposed IoT Composition Architecture based on DIY Business Process Modeling Approach. The first two layers of both the architectures have the same goals of representing the physical devices as virtual objects. In different systems, these virtual objects may be represented using different encoding technologies but nevertheless, the main objective of the virtual objects remains the same. Hence, the general functionality of the first two layers in the existing and the proposed architecture is same.

The third layer represents the Service Composition Layer where the virtual objects are utilized to create service objects. The main aim of this layer in the existing and the proposed architecture is the same and that is to provide a DIY approach to enable use to create the services of their own choice and according to their own requirements. However, the existing systems mostly expose these user-created services as Application Programming Interfaces (APIs) and let the users create their own client applications by utilizing these exposed APIs. This allows for the creation of more requirement specific applications but it violates the basic concept of DIY development paradigm. The client application development with application logic at the client side is again a developer's job and people with no programming skills may not be able to create their own custom applications. At the end such application development is time consuming and the resulting applications cannot be modified easily.

Here, a question arises which states that why we needed to utilize Business Process Modeling for the representation of an IoT application? The answer lies in the basic definition of 'Process' and 'Business Process' itself. The word 'process' has been defined by Havey [22] in terms of its verbal meaning as to handle, as in processing of an error, or processing a message while the noun meaning referring to a program running in an operating system, and to a procedure, or a set of procedures, for accomplishing a goal. According to Havey, the



implications of the term ‘process’ are movement, work, and time; a process performs actions over some interval of time in order to achieve, or to progress to, some objective.



**Figure 1: Comparison of the existing and proposed DIY IoT architecture**

From the perspective of Business Process Modeling (BPM) a business process is defined as a series of steps performed by a group of stakeholders to achieve a concrete goal. These steps are often repeated many times, sometimes by multiple users and ideally in a standardized and optimized way. A business process can be manual or automated. If manual, the process is achieved without the aid of automation or assisting technology. If automated, a technology aid has been put into place which assists users in implementing the process in a more accurate, standardized or optimized manner. Examples of business processes include receiving orders, invoicing, shipping products, updating employee information, or setting a marketing budget.

From the perspective of IoT implementations, the same definition of business process fits to the processes or applications that run on top of an IoT implementation defining its behavior via the interactions of sensing and actuating agents in order to provide useful services to the end-users. The example of such an implementation would be a Smart Space, where the process running as the application would define the behavior of the Smart Space sensing (temperature, humidity, illumination, proximity etc.) and actuating (Air conditioning, heating, lightings, door locks etc.) agents based on pre-defined events (resident's arrival, time schedule, emergency state etc.). In this example, the IoT application for the Smart Space basically acts a business process model for the particular resident of the Smart Space. Through the personalized process model, the resident can define the behavior of her space according her specific needs.

Apart from the fact that an IoT application can be represented via a business process model, BPM has already been a part of the software development industry for more than twenty years now. In software development, BPM has been originally used a method for requirement analysis and specification drawing as a better communication medium between the clients and the developers. According to Barjis [23], one of the leading causes for the high failure rate is still poor process modeling (requirements' specification). Therefore both researchers and practitioners recognize the importance of business process modeling in understanding and designing accurate software systems. This is the main reason it has an importance in the IoT application development paradigm where end-users will be developing software according to their own requirements. According to Deepak Singh [24], it aligns a process execution with actual operation activity and redesigning the process is made simple by business process modeling. From software development perspective, it is beneficial for application redesigning. Hence it makes the system agile and easy to incorporate changes. As agility is the core factor in

IoT related application development via the end-user, BPM can play an important role from this perspective.

Graphical notations associated with business process modeling such as BPMN are standardized notations [25] which can be interpreted globally by anyone with basic knowledge of the standard. As IoT is envisioned as a global network, these standardized notations can become a global language for the IoT application development and hence aid in the mass utilization of general public in IoT development. The standardized notations are easy to learn and it is ideal for training of new people and rapid knowledge transfer. These factors can reduce the development time and ultimately the cost of development which a major concern in the software industry. The same benefits can be availed in the IoT paradigm hence this study investigates the applicability and usability of BPM approach to provide a better DIY IoT application composition environment.

The proposed IoT composition architecture enhances the existing architecture by providing a Business Process Modeling approach for a DIY application development platform. The service objects created at the Service Composition Layer are represented as standardized Business Process Modeling Notation (BPMN) along with notations to help create the application logic in an intuitive visual manner. This is the reason we have represented the Application Layer as the Business Process Layer in the proposed architecture because at this layer, the business logic of an IoT implementation is defined using business process modeling approach. The simple action such as Drag-n-Drop, mouse clicks etc. are well known to general population of modern age, hence enabling anyone to create their own IoT applications. No coding is required on behalf of the application developer (end-user) and the created applications can be easily modified through the manipulation of the business process model representing the application. BPMN is a standardized set of notations designed to allow non-technical users to express their requirements

to technical people, hence it can provide a better DIY programming platform for IoT related applications.

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. The main contribution of this work include the utilization of BPM diagramming language for the IoT users to make their own desired processes and let IoT protocols i.e. CoAP enable those models to be executed in their environment. DIY interfaces have been developed for enabling the users to create virtual objects for their IoT devices, visually compose services from those virtual objects and finally combine the composed services into IoT applications or processes in the form of business process models. The architecture has been developed following the layered approach so each layer complements the ability of the users to create complex programming structure without the need to learn any programming languages.

The next chapter of the dissertation provides a thorough analysis of the related projects and the limitations of the previous works from the DIY perspective. Chapter 3 provides the details of the proposed architecture by explaining the major component and functionalities at each layer. Chapter 4 provides detailed design of each layer in terms of static structure, sequence models and configuration models of each layer. Chapter 5 provides some insight into the implementation of the proposed system and provides basic performance analysis of the major functions at each layer. In order to evaluate the utilization of the proposed architecture in different IoT scenarios, two use-cases have been developed from the fields of industrial robotics and smart spaces. Prototype implementations have been developed in order to evaluate the performance of the proposed architecture in each use case from the perspective of its usability. For the usability analysis of the proposed enhanced IoT composition architecture based on DIY business process modeling approach, System Usability Survey (SUS) has been implemented as part of each use

case. Participants from various age groups and technical backgrounds have been involved in the usability analysis of the proposed architecture. The SUS data gathered from the usability experiments has been utilized for SUS based usability scoring of the system. Statistical analysis of the data recorded for the participants' interactions with the system has been carried out to assess the proposed system's DIY behavior. This is done by dividing the participants into two groups of programmers and non-programmers respectively and t-test analysis of the user performance during the usability studies for the two use-cases has been performed to show if the system provides a better DIY environment to all participants regardless of their programming skills level. Chapter 6 and Chapter 7 provide the details of the use-cases and the usability study based on the use-cases respectively.

## 2. Related work

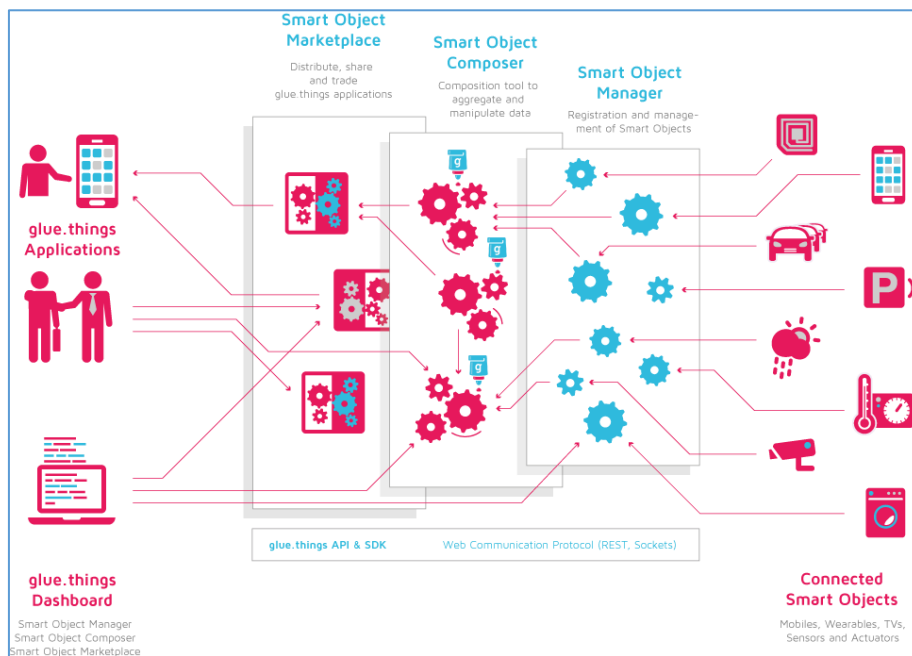
The IoT vision is the realization of a worldwide network of connected and interoperable smart devices which can provide services to the people. Although the individual technologies in terms of communication and devices have improved tremendously, the implementation and realization of such a complex network is still in its nascent stage. The main hurdles include the coping with the heterogeneity of the hardware and mass involvement of general public in the IoT development and adaptation process. The first issue is being abstracted out with the help of middleware based solution and service-orientation. The second issue is of greater importance because in our view, the realization of a successful worldwide IoT implementation is not possible without the involvement of masses in the development of IoT. According to [26] “there is a reason people are excited about the IoT: It feels like a big opportunity to improve how we design and build products”. With the proliferation of digital technologies, user led innovation is being considered to have significant commercial value [27].

### 2.1. Existing IoT composition research

Here we discuss some of the recent IoT related projects which, in some way, provide a unique and alternate interface for end users to get involved in the design and development of IoT in our daily life.

Glue.things [28], is a recent project which implements the concepts of device integration and real-time communication using the recent technologies of Web Sockets, MQTT and CoAP. The protocols are utilized on real-time data streams networks to allow mashups of the data streams, add actions etc. The final mashups are deployable in a distributed environment. The system specially focuses on the composition of data streams from Web services and IoT devices with

Web interfaces. The main aim of Glue.things as shown by the architecture in Figure 2 is to utilize web technologies for providing interoperability platform with REST APIs, JSON data models and Web sockets etc. The Mashup interface is based on Node-RED [29] which is a browser based visual data stream aggregation tool. Another important aspect of the project is the utilization of well-established Open Source technologies.



**Figure 2: Generic architecture of the Glue.Things project[28]**

IoT Mashup Application Platform (MAP) [30] is an effort towards flexible interoperability of smart things with smart phones in users' personal pervasive environments. IoT Map decouples the development of mobile application from the static model chosen by the designer/developer of the application which is a great hurdle in the way of application adaptability to the user's needs and/or the surrounding environment. IoT MAP utilizes the concept of abstracted service objects for the development of IoT applications. This is achieved through a set of application programming interfaces (API) exposed for functions like discovery and retrieval of the service objects etc. The business logic is written in POJO [31] while

abstracting out the details of connectivity and implementation of smart things. As shown in the conceptual architecture presented in Figure 3, the users can also utilize a graphical authoring tool based on NodeRed to compose an IoT application which can be converted into API calls and executed as an application.

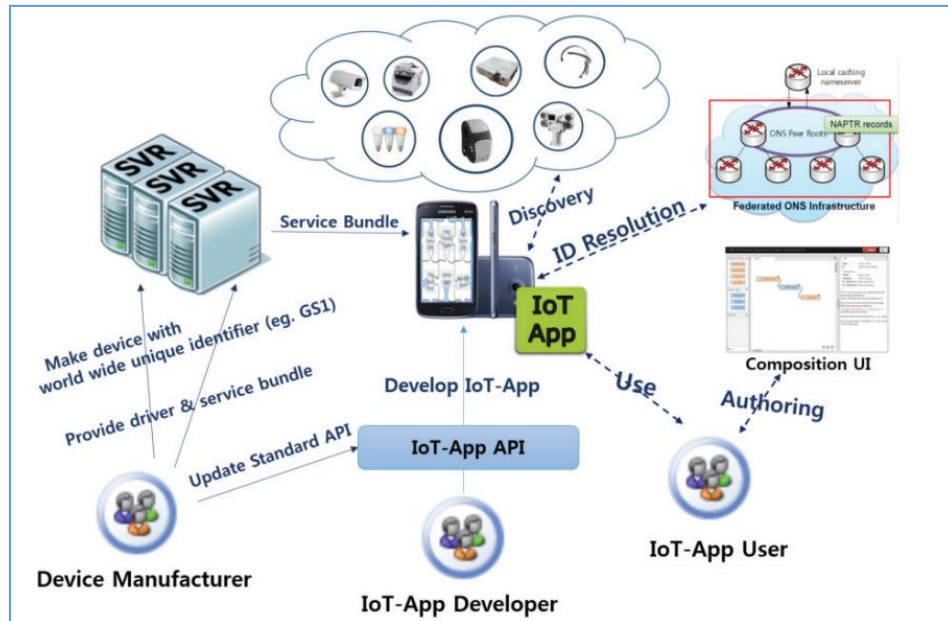


Figure 3: Conceptual architecture of IoT MAP [30]

From the perspective of reducing the complexity of IoT application development, IoTLink [32] has been presented as a development toolkit. The IoTLink toolkit is based on a model driven approach, which utilizes a domain-specific graphical programming language to allow inexperienced developers to compose their own IoT applications. It encapsulates the underlying complexities of interaction with IoT devices and services into visual components. These visual components represent the IoT devices as virtual objects which can be accessed through multiple communication technologies. The high level architecture of the IoTLink project is shown in Figure 4. The project utilized IBM post-study system usability questionnaire to get user feedback regarding the system usability.



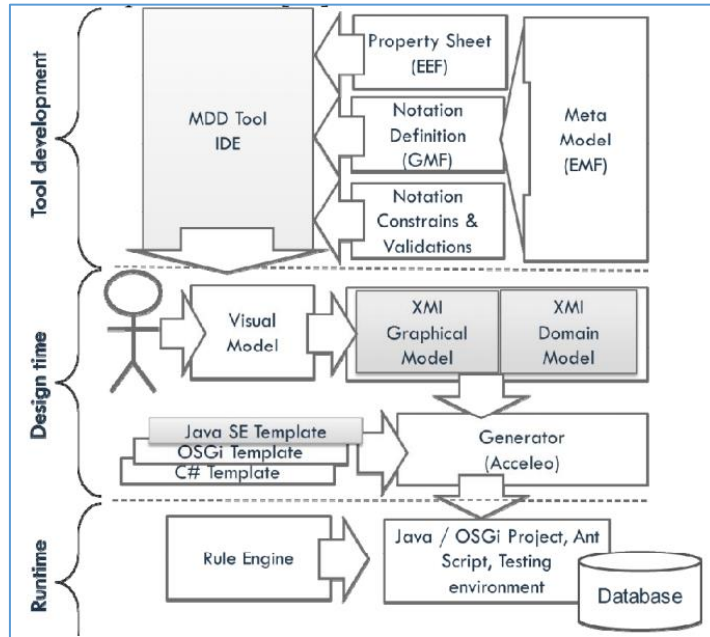


Figure 4: High-level architecture of the IoTLink [32]

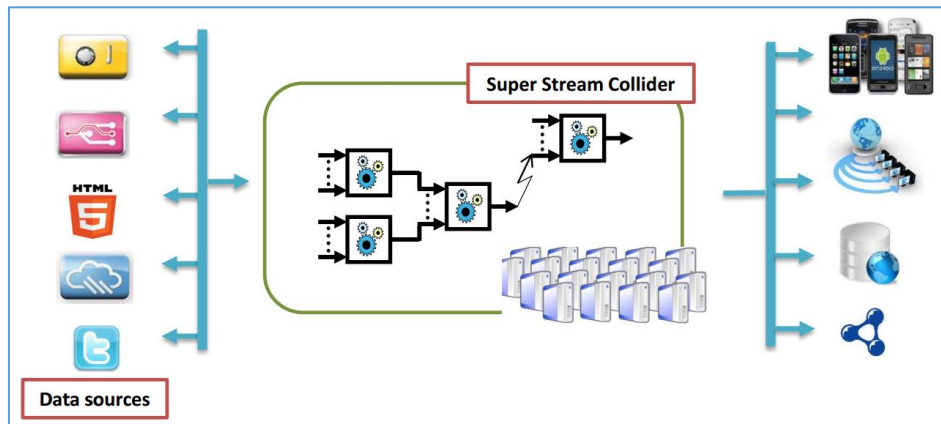


Figure 5: Layered architecture of the SSC platform[33]

Super Stream Collider (SSC) [33] as shown in Figure 5 is another platform which helps enable everyone, from novice IoT users to expert programmers, to develop IoT applications in the form of near-real-time data streams. The web-based interface for SSC enables anyone to create their own mashups by combining linked data sources and linked streams to create resources which can be used as applications for IoT scenarios. The system supports drag-n-drop

technique with a SPARQL/CQELS editor. As the platform is intended for large data acquisitions through streams, it utilizes cloud infrastructure for fetching the data, processing and dissemination of data.

Ambient Flow [34] is another recent effort towards making the IoT devices interoperable and to provide an intuitive interface for enabling the daily life users of smart devices to remix the functionalities of their devices in a fun and potentially innovative ways. The project utilizes the user's smartphone as the gateway to communicate with heterogeneous smart devices. It provides a flow-graph model based visual interface through which the design can be easily created and these designs are then sent over a network to the smartphone in order to be executed. An architectural overview of the system is presented in Figure 6.

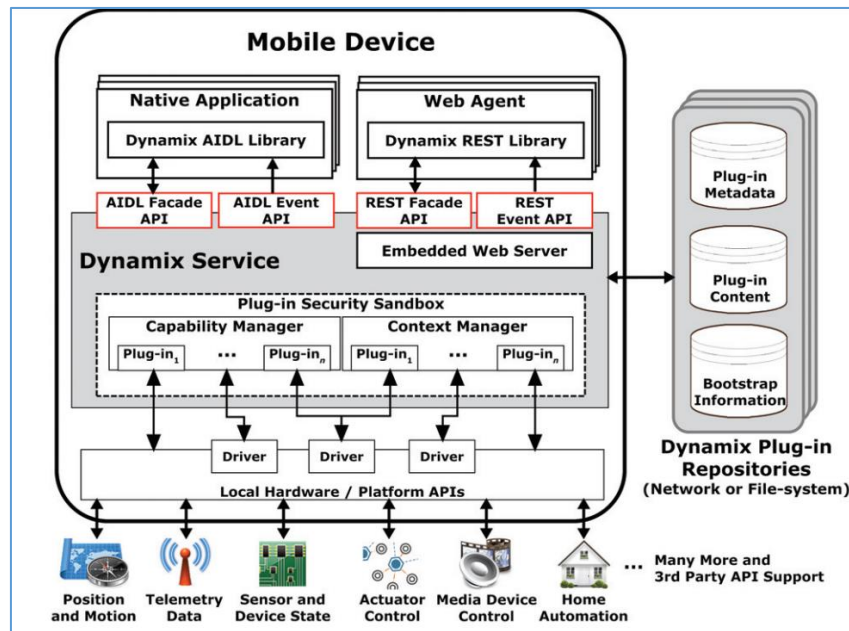


Figure 6: High-level architecture of Ambient Flow [34]

As part of the OpenIoT project, a visual development approach has been presented by Kefalakis et al. [35]. The visual development tools are intended to be used as an integrated development environment (IDE) for the support of IoT application development lifecycle. The

tools presented are based on a semantic IoT architecture and claims to be a minimal programming environment for IoT application development. It uses a node-based user interface theme to allow the user to model service graphs and then convert them into SPARQL queries.

AppsGate [36] is one of the latest efforts towards an end-user development (EUD) environment for smart homes. An EUD empowers the people with tools to create things from scratch and enables them to test, debug, maintain and customize the functional coverage of a system [37]. AppsGate provides direct operation as well as programming opportunity to smart home users. Visual representation of smart devices lets the user directly turn on/off switches etc. while programming smart home is viewed as a creative activity which is defined by residents according to their environment and preferences. AppsGate also provides a timeline based temporal snapshot interface through which the users can monitor various levels of details regarding the smart home.

To reduce the development complexity of IoT applications in terms of communication with heterogeneous device, [38] presents a high level domain specific development language. The language and IDE resulted in their efforts is termed as DSL-4-IoT Editor which is basically a high level visual programming language. The editor has been developed using JavaScript and for the execution engine, an open source project OpenHAB [39], has been utilized. Other similar Model-driven domain specific languages include PervML [40] which promotes the idea of role separation in order to categorize the IoT application developers. DiaSuite [41] is based on Sense/Compute/Control logic and provides Java based programming framework in conjunction with graphical renderer to simulate the applications along with a deployment framework.

## 2.2. Issues with the existing solutions from the DIY perspective

This section subsection provides a brief analysis of the existing systems providing alternative development approaches in order to be used by wider group of people. The issues mentioned in this section are based on review of the systems/projects from the DIY perspective only.

Although the project considers the utilization of existing open-source technologies, it does not consider the users skills to utilize those technologies. The system itself is composed of several open-source components which even make the deployment of the system complex for most skilled users. Node-RED is a powerful tool for composition IoT mashups but it still requires some level of programming skills on behalf of the developers as the composed services must be utilized in a Web application in order to be utilized. It is therefore, less suitable for end-user development especially for common people with no programming skills.

The project utilizes POJO for describing the business logic of objects by the users. POJO is a programming style and the user must learn it before creating their customized objects for utilization in this platform. Users are still dependent on the manufacturer for providing the ID resolution procedure and the driver bundles for the smart things [30]. The APIs intended for the manufacturers to implement in their smart devices are not standard interfaces. From a DIY perspective, the platform still requires considerable level of programming skills on behalf of the application developers and hence may not be suitable for ordinary users without much programming skills. From the evaluation of the IoT MAP project, it is evident that the final composed application is run fully on the remote device which makes it suitable for latest smartphones only and may not prove efficient for IoT constrained devices.

IoTLink utilizes graphical box notations for the representation of every entity in the system including virtual objects and services instead of graphical icons or some sort of standardized notations. The lack of icons or standardized graphical notations may make the graphical language challenging for the non-experienced developers specially users with no computer background. Even the testing and usability analysis of the system is performed by experienced programmers with a median experience of 7.5 years. This shows that programming skill is a requirement for using the IoTLink and hence not suitable from the DIY point of view.

Although the toolkit provide a better development environment for users with less or no programming skills but the graphical editor used for the visual programming requires its users to have an understanding of programming structures such as if-else and loops etc. This can be a challenge for people who have less or no interaction with computers especially in a programming sense. The other main attribute of the SSC is representing IoT resources as streams of data and those streams can be utilized for querying data for a user's applications. This again requires programming skills and hence makes is rather unsuitable for adoption in daily life IoT applications where the non-technical user needs to customize the behavior of their sensing and actuating equipment.

The Ambient Flow project is strictly a domain specific implementation with focus towards providing a better experience to users with a set of smart space design tools. The limitations are evident because the flow-graph model can use only a small set of graphical shapes to represents every element that could be used in a design hence limiting the diversity and intuitiveness of the interface. The smartphone based execution of user's smart space design also limits the applications to personal environments and may not be applicable to wide range of IoT scenarios.

The request definition module for visual development is web based tool which can operate only in the presence of a cloud connection and hence cannot be used for standalone applications.

The services compose-able through this interface must be OpenIoT platform based services thus limiting its application. The Request Presentation module of the system is a Web application which utilizes the SPARQL based composed services to create visualization dashboards which can only be used to acquire data and visualize it. This limits the domain of the project and makes it unsuitable for application development and deployment for IoT constrained resources.

AppsGate project is specifically designed from the perspective of smart homes and hence is a domain specific implementation. The IoT vision is a global network of multi-domain devices, services and applications. A project can only contribute towards the global realization of IoT vision if it can be easily applied to multiple domains which is not true for the AppsGate project.

Although domain specific languages support and assist the developers of IoT applications, the programming skills on behalf of the developers cannot be ruled out. Hence, such visual development interfaces/languages cannot be utilized for DIY development environments in the IoT scenario.

Given above are a few of the prominent efforts towards intuitive and alternative development strategies for the Internet of Things applications and services. The goal of this section was to highlight these efforts and to identify the shortcomings or limitations of these systems when truly considered from the perspective of a DIY scenario where the users of such systems are not mandatorily programmers. As described, most of these systems are either too complex to deploy and/or program by a non-programmer person or intended for a very limited application domain. It is therefore, necessary for the realization of IoT to develop a standardized interface/development environment which can easily be adapted to multiple domains while providing an intuitive approach towards the development of IoT application.

## 2.3. Existing IoT protocols

This section provides an overview of the prevailing IoT protocols. We only discuss the most recent and popular protocols which are best suited for the IoT implementations.

HTTP is the foundation of the client-server model used for the Web. The more secure method to implement HTTP is to include only a client in your IoT device, not a server. In other words, it is safer to build an IoT device that can only initiate connections, not receive. Although HTTP can be utilized as a reliable protocol for IoT implementations but due to its heavyweight protocol stack, it is not suitable for the resource constrained IoT devices.

WebSocket is a protocol that provides full-duplex communication over a single TCP connection between clients and servers. It is part of the HTML 5 specification. The WebSocket standard simplifies much of the complexity around bi-directional Web communication and connection management.

XMPP (Extensible Messaging and Presence Protocol) is a good example of an existing Web technology finding new use in the IoT space. XMPP has its roots in instant messaging and presence information. It has expanded into signaling for VoIP, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. It is a contender for mass scale management of consumer goods such as washers, dryers, refrigerators, and so on.

MQ Telemetry Transport (MQTT) is an open source protocol for constrained devices and low-bandwidth, high-latency networks. It is a publish/subscribe messaging transport that is extremely lightweight and ideal for connecting small devices to constrained networks. MQTT is bandwidth efficient, data agnostic, and has continuous session awareness. It helps minimize the resource requirements for your IoT device, while also attempting to ensure reliability and some degree of assurance of delivery with grades of service. MQTT targets large networks of small

devices that need to be monitored or controlled from a back-end server on the Internet. It is not designed for device-to-device transfer and it is not designed to “multicast” data to many receivers. MQTT is extremely simple, offering few control options.

The Constrained Application Protocol (CoAP) was designed by the IETF for use with low-power and constrained networks. CoAP is a RESTful protocol. It is semantically aligned with HTTP, and even has a one-to-one mapping to and from HTTP. CoAP is a good choice of protocol for devices operating on battery or energy harvesting.

As CoAP uses UDP, some of the TCP functions are reproduced in CoAP. For example, CoAP distinguishes between confirmable (requiring an acknowledgement) and non-confirmable messages. Requests and responses are exchanged asynchronously over CoAP messages. All the headers, methods and status codes are binary encoded, which reduces the protocol overhead. Unlike HTTP, the ability to cache CoAP responses does not depend on the request method, but the Response Code. CoAP fully addresses the need for an extremely lightweight protocol and the ability for a permanent connection.

CoAP is an open standard and not proprietary like some of the earlier protocols for networked embedded systems [42]. The open standard means that the standardization process is open to public and that it is free to be used by anyone without any royalty. This fact alone makes it perfect for the global implementation of the Internet of Things.

Secondly, the CoAP protocol has been designed with the focus towards resource constrained devices associated with IoT. Thus it is light in comparison to other IoT protocols. It is based on the same principles Like HTTP thus it is very easy to use. It provides datagrams based asynchronous communication which is suitable from the perspective of constrained devices because it is very lightweight in terms of resource consumption.



CoAP runs over IP where IPv6 is the future of IoT. This feature enables the future IoT to easily integrate with the current IP based IT infrastructure of organizations and personal spaces. Using the IP based communication infrastructure, CoAP can be utilize for interconnecting the IoT devices with the HTTP and RESTful web and this can be done through simple proxies.

Lastly, as CoAP is still in the standardization phase, new features are constantly being added to the protocol stack. The emerging CBOR encoding for CoAP has proved to be a better suit for REST than the conventional JSON and HTTP [43]. This study does not claim that CoAP alone can be utilized to fulfill all the requirements of the future IoT but it seemes a better choice based on the facts described above to use CoAP for the prototype implementations related to this study. Later on other IoT protocols can be studied and implemented as part of the proposed architecture.

## 3. Proposed DIY IoT System Architecture

This chapter is dedicated to the description of the proposed DIY IoT architecture. The details of each layer are divided into shared components and layer specific or application specific components of the system. Each layer is described in terms of components and the activities performed by those components. The following text provides the description of each layer. Figure 7 shows the detailed architecture of the system.

### 3.1. Virtual Object Layer

The application specific components at the Virtual Object Layer (VOL) include the Virtual Object Manager (VOM), VO information acquisition interface and the VO repository. The VOM represents the physical things in the form of VO Behavior, VO attributes and VO visualization. The VO Behavior is the services or functions which can be utilized by the system to interact with the physical thing represented by the VO. A simple example would be the name of CoAP service which can be called remotely to interact with the physical thing. The VO Attributes are the other information in the form of complete URI and Location etc. which collectively describes the existence of the physical thing through its virtual representation. Finally, VO Visualization is the graphical representation of VO depicting the type of the physical thing represented through it. It is an icon or string of characters visually representing the underlying physical entity such as thermometer icon to represent a temperature sensor.

The VO information acquisition interface is used to register physical things as virtual objects. This interface can be a local data entry interface for a user or administrator to register the available physical things or it can be exposed as an online service to enable users to remotely

access and register their devices. After the registration, the information related to VOs is stored at the VO Repository. VO Repository holds the information about virtual objects in XML document for so that it can be transferred easily over the Internet.

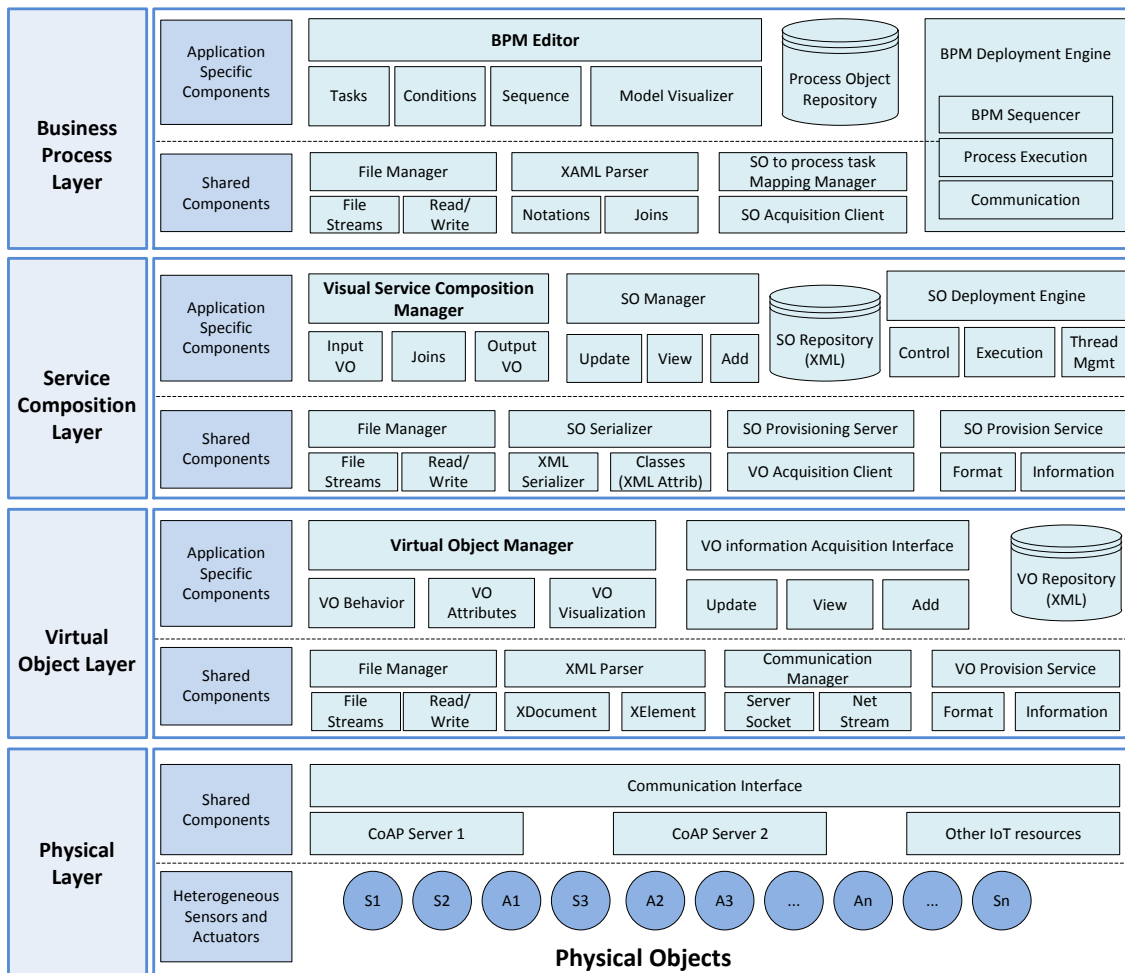


Figure 7: BPM based DIY IoT composition system architecture

The shared components at the VOL include File Manager, XML Parser, Communication Manager and VO Provisioning Scheme. The File Manager is responsible for reading and writing files to the file system of the computer on which the VOM is deployed. It uses the basic file streams to perform all the I/O operations. The XML Parser, as evident from the name, provides parsing service for the conversion of VO information into xml documents so it can be saved in

the repository and vice versa. The XElement and XDocument classes have been used to perform the parsing tasks. The Communication Manager is responsible for providing connectivity to the VOL with adjacent layers of the system. It acts as a server for the upper layer in order to communicate and provide the VO information based on Sockets and Streams. In order to provide the VO information to other layers, an information exchange scheme is needed. For this purpose, the VO Provision Scheme provides the format of messages to exchange the information and the data structure needed to encapsulate that information.

### 3.2. Service Composition Layer

The Service Composition Layer (SCL) is the part of the system where the unit services are composed based on the information obtained from the virtual objects provided by the VOL. The application specific components of the SCL include Visual Service Composition Manager, Service Object (SO) manager, the SO Repository and the SO Deployment Engine. Each of these components is explained in the following paragraphs.

The Service Composition Manager (SCM) is responsible for providing an intuitive and easy to use visual environment where the VOs obtained from VOM are rendered as graphical module, enabling the users to drag-n-drop modules onto the main canvas and join them to create service objects. A Service Object consists of an Input VO joined to an Output VO through the Join element. The join element also provides an interface for the user to create conditions for the execution of the Service Object. Each SO is represented as a combination of Module elements and a join element in an XML document which is stored at the SO Repository.

The SO Manager provides the functionality of viewing the previously stored service objects, adding new SOs and updating the existing ones in the repository. The SO Deployment Engine is a standalone entity at the SCL which is responsible for the execution of individual services. This

individual service execution capability can also be used for testing the SO objects. The Deployment Engine provide multithreaded execution environment with a control interface.

The shared components at the SCL include File Manager, SO Serialization, Acquisition Client and Provisioning Server along with the SO Provision Scheme. The File Manager is responsible for the actions related to file system interaction in the form of file I/O operations. This component is used by the application specific components to perform their required I/O operations with the file system of the host machine. It uses the basic file streams to perform all the I/O operations. The XML Serializer component is used to convert the VO implementation classes to an xml format based on the XML attribute class developed as part of the system. This technique allows the conversion of individual VO's attributes and selected behavior, as part of a service object, into an xml format which can be easily transferred over the Internet.

The SO Provision Server provides a listener for the connections from the business layer for the provisioning of services which can be used to compose a process model. The VO Acquisition Client enables communication with the VOL for the acquisition of all the available virtual objects. This layer also defines a SO Provisioning Scheme which defines the message exchange formats and the data structures used to hold information related to Service Objects.

The Service Objects created at the Service Composition Layer are stored as an XML repository at the SCL. The SOs are designed in two types, one type is the independent SO which contains all the required entities to be runnable as an isolated process. The second type of SO are the one which define partial functionality (Unit SO) and can be combined into sequences to create larger processes.

### 3.3. Business Process Layer

Business Process Layer (BPL) basically represents the Application Layer in the existing architectures. As the the proposed system uses BPM approach for providing a DIY IoT application development environment, the layer has been termed as the BPL. At the BPL, the application specific components include the BPM Editor, The Process Object Repository and the Process Execution Engine. The BPL utilizes the service objects composed at the service composition layer and represents them as business process modeling notations. Normally a service object is represented as a BPMN task notation while other notations such as Gateways are defined at the BPL for functions such as condition evaluation or setting multiple paths in a process model, Script notations represents data processing and generalized actions while Events represent the start and end of process models. Events can be further utilized for message passing among tasks and other notifications but the current implementation utilizes them only as the demarcating elements in the IoT process models.

The BPM Editor application enables the user to create a functionality flow for an IoT environment in the form of a graphical business process model. The user creates rules and applies various scripts according to the conventions of the BPMN. This model can be saved as a process object in the repository at BPL and reloaded into the editor application for update at any time. An optimized version of the same file is created which can be used by the Deployment Manager at the BPL.

The Deployment Manager is implemented in such a way that it can directly communicate with the remote IoT devices. The basic information for communication is extracted from the optimized version of the business process model file. The file is further parsed to extract the information related to individual service objects as presented in the BPM by the user. Each

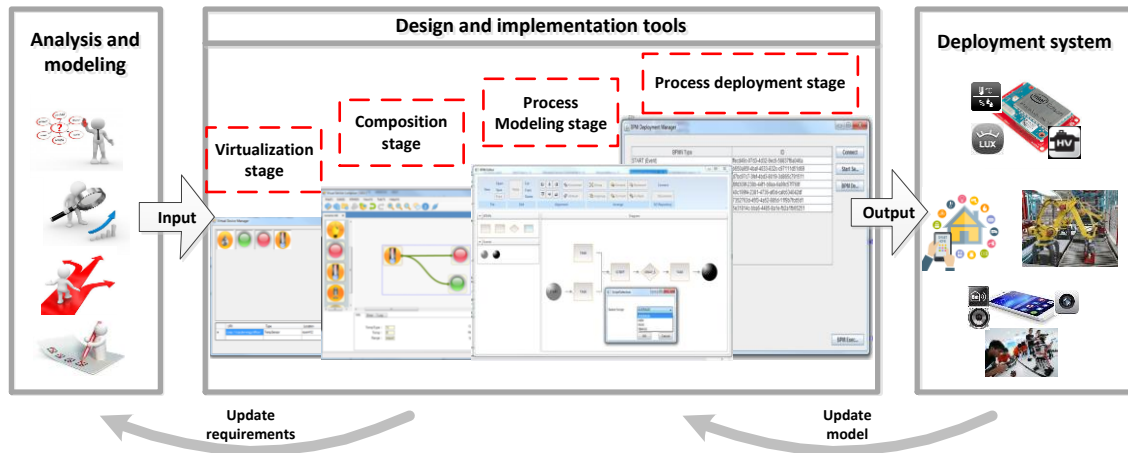
service object is retrieved in the order specified by the user and sent as XML document to the primary IoT device associated with the service object. This information is sent to the device via a generic CoAP post which is a default implementation in every IoT devices interacting with the proposed architecture. The remote IoT device then parses and executes the service object and sends back a CoAP response which is utilized by the Deployment Manager to further execute the business process model.

### 3.4. Development Process

Figure 8 shows the development cycle based on the proposed BPM based IoT system. As in any development process, the user must first identify the requirement for the system/process that needs to be developed and deployed. Hence, as shown in the figure, requirement analysis is the first step in which the user specifies what functions the system will be performing and how it will perform these functions.

Once the functionality of the system/process is decided, the next step is to utilize the tools of the proposed system to create virtual objects. These virtual objects are basically the software representation of the physical devices (sensors and actuators) utilized by the system/process being developed. These devices are termed as IoT resources in this document and certain information regarding these resources such as URI, location, type etc. are provided by the user to create virtual objects. The next stage is to create service objects (SO) from the available virtual objects. A service object is basically a combination of an input and output VO with certain restriction applied by the user for its operation. For example, a temperature range set on a thermistor to operate on a switch or led. A service object is a complete operation and can be executed in isolation. Unit service is another term used by the system where only the input or output VO along with its operation restriction is composed into a service object. Unit SO can be

utilized in situations where a complete definition of a service object in terms of input device, operation rule and output device is not possible and a service object is required only for a data acquisition purpose or for an actuation purpose.



**Figure 8: Development process through DIY IoT composition system**

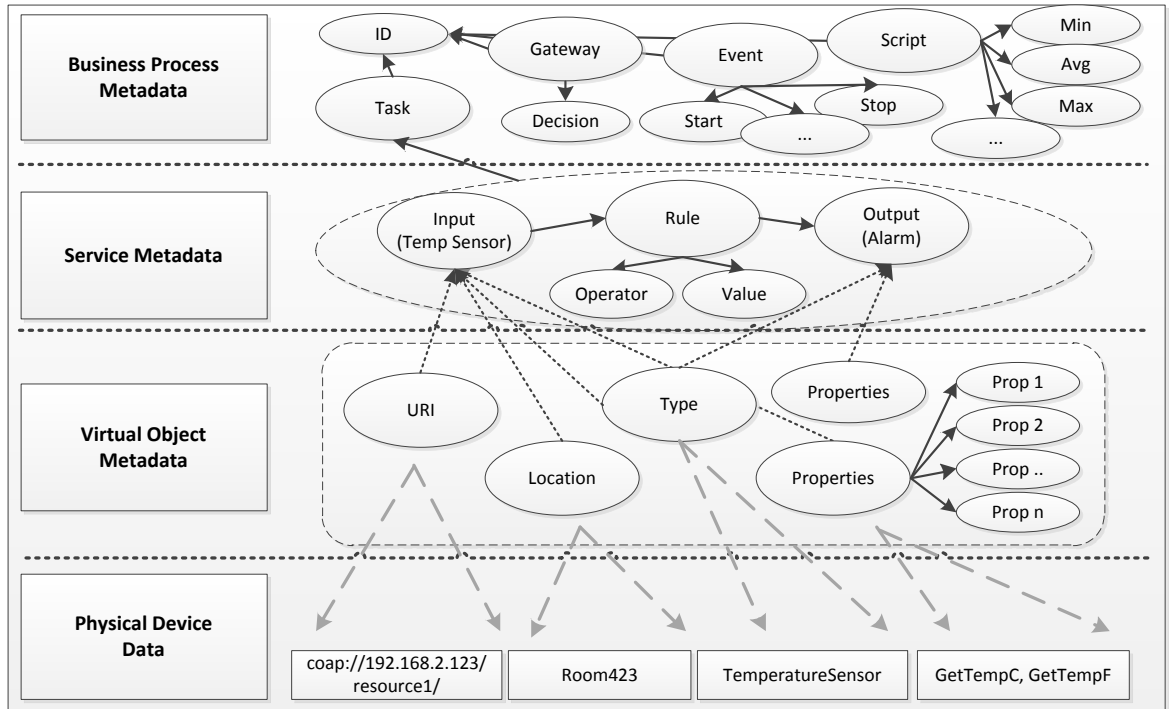
The service objects from the Service Composition Manager are used by the Business Process Modeling Editor of the proposed system to represent tasks in a process. The BPM Manager provides implementation of other BPM notations such as events, decisions and scripts to enable the user to create a visual flow for the system/process in question. This modeling is performed according to the requirement analysis of the system/process being developed. The resultant BPM is deployed using the BPM Deployment Manager and functionality of the system is tested. If any changes have to be made to the model flow, the Service Composition Manager and business process modeling manager can be used for the purpose. If new requirements arise for the system, the whole process of service composition and business process model development must be followed from start.



### 3.5. Data representations

Figure 9 shows the data production, representation and utilization scheme for the proposed system. At each layer the data is presented as a single instance illustrating the metadata used at that layer. The first layer consists of the physical devices which are the IoT resources responsible for the production of data. These IoT resources are represented in the system as virtual objects. The metadata for a virtual object is shown in the second layer of the figure. In order to register an IoT resource with the system, the owner must provide the information corresponding to the metadata in the Virtual Object Layer. This data includes the URI of the resource which describes the protocol and the complete network address of the resource so it can be accessed from anywhere. Other metadata include the location of the resource so it can be associated with other resources during the design of the business process model. The Type metadata classifies the resource as an input or output resource and it also helps in visual representation the resource in the system. This enables the users to interact with the VO in a more intuitive manner. Finally, the Properties metadata represents the actual functionality of the resource which can be called as a remote function. This consists of a list of methods provided by the owner of the resource.

As explained previously, the Service Composition Layer utilizes the VO definitions to create service objects (SO). The service metadata consist of an input SO, a rule which includes a range values according to the data type of the input VO along with an operation to describe a restriction on the operation, and finally an output VO whose operation is based on the evaluation of the condition for input VO. All these data elements combine to create a service object. In a unit SO, the input or output SO is replaced by a generic VO representing standard input or output. Service objects are represented as Tasks in the BPM.



**Figure 9: Metadata representation at each layer of the system**

The BPM meta-data consists of Task (representation of SOs from the service layer) and other BPM notation such as events, gateways (decisions) and scripts etc. These visual notations are presented to the user in the form of an editor and the user interacts with them via simple clicks and drag-n-drop techniques to create process model according to the desired functionality of their system. Each element is separately identifiable with a GUID and hence multiple instances of each notation can be utilized in a model.

## 4. BPM based DIY IoT System 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 component operation at each layer. The following sub-sections present the design details of each layer.

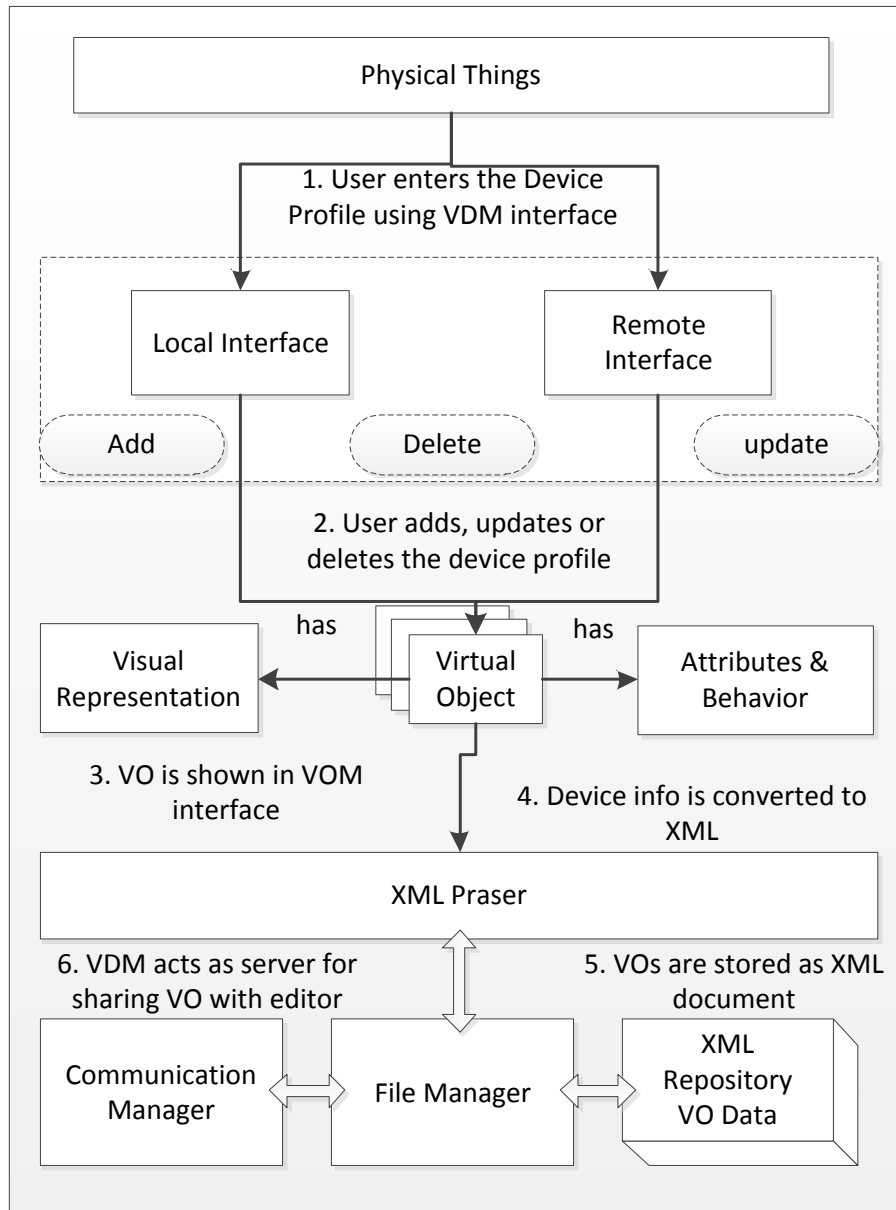
### 4.1. Virtual Object Layer

The Virtual Object Manager (VOM) is the main component at the Virtual Object Layer (VOL). 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.

Figure 10 shows the startup and operational configuration model for the Virtual Object Manager (VOM). The local and remote interfaces are used by the users to enter information related to any CoAP enabled physical thing about which they have the required information. These interfaces can also be used to add new VOs, Delete and update the previously registered VOs. Once a VO is created for a physical thing, the visual representation, the attributes and the behavior of the physical thing are encapsulated in this virtual object.

The visual representations for the VOs already registered can be viewed as a list in the main interface from where the user can view the information associated with a VO, update and delete any information for the VO. The XML Parser is used to convert the VO into XML schema already defined for the VO representation. This XML version of VO is stored in the XML repository using the File Manager component. In order to send or transfer the VO information to a remote requesting component, the Communication Manager component through the File

Manager reads all the VOs and sends it to the requesting component according to a predefined information exchange scheme.



**Figure 10: Virtual Object Manager startup and operational configuration**

The static structure of VOM is shown in Figure 11. 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.

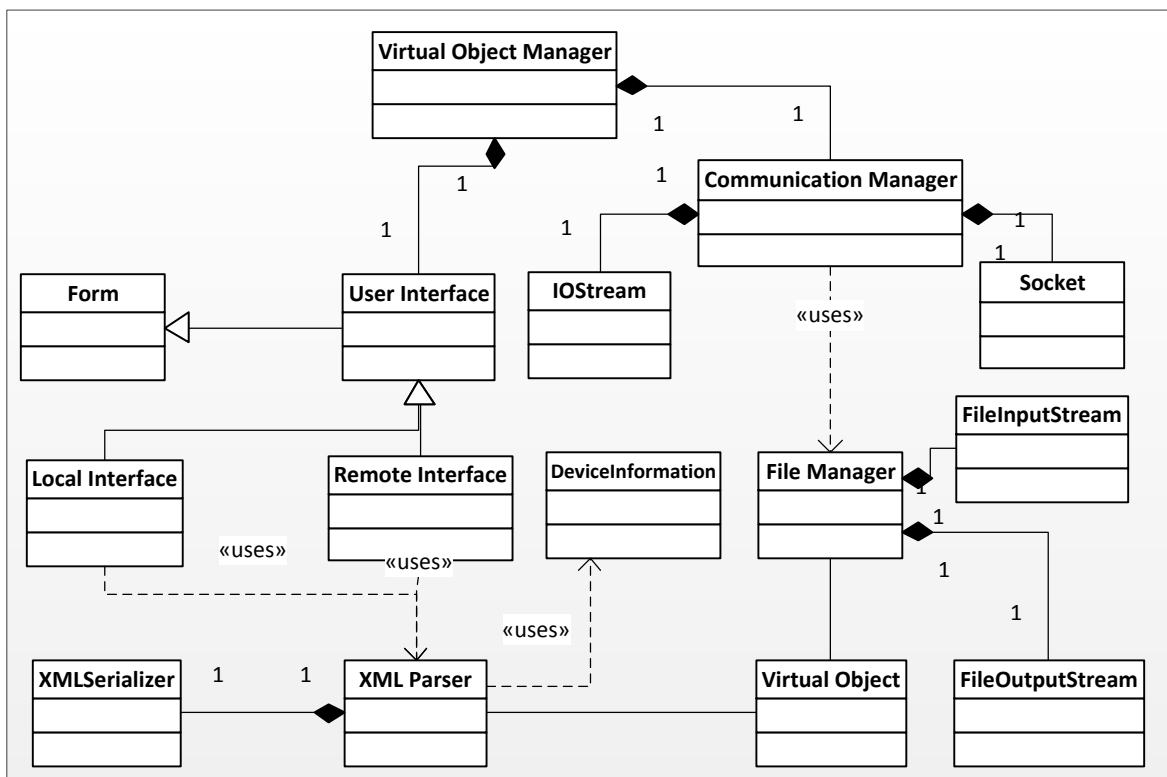
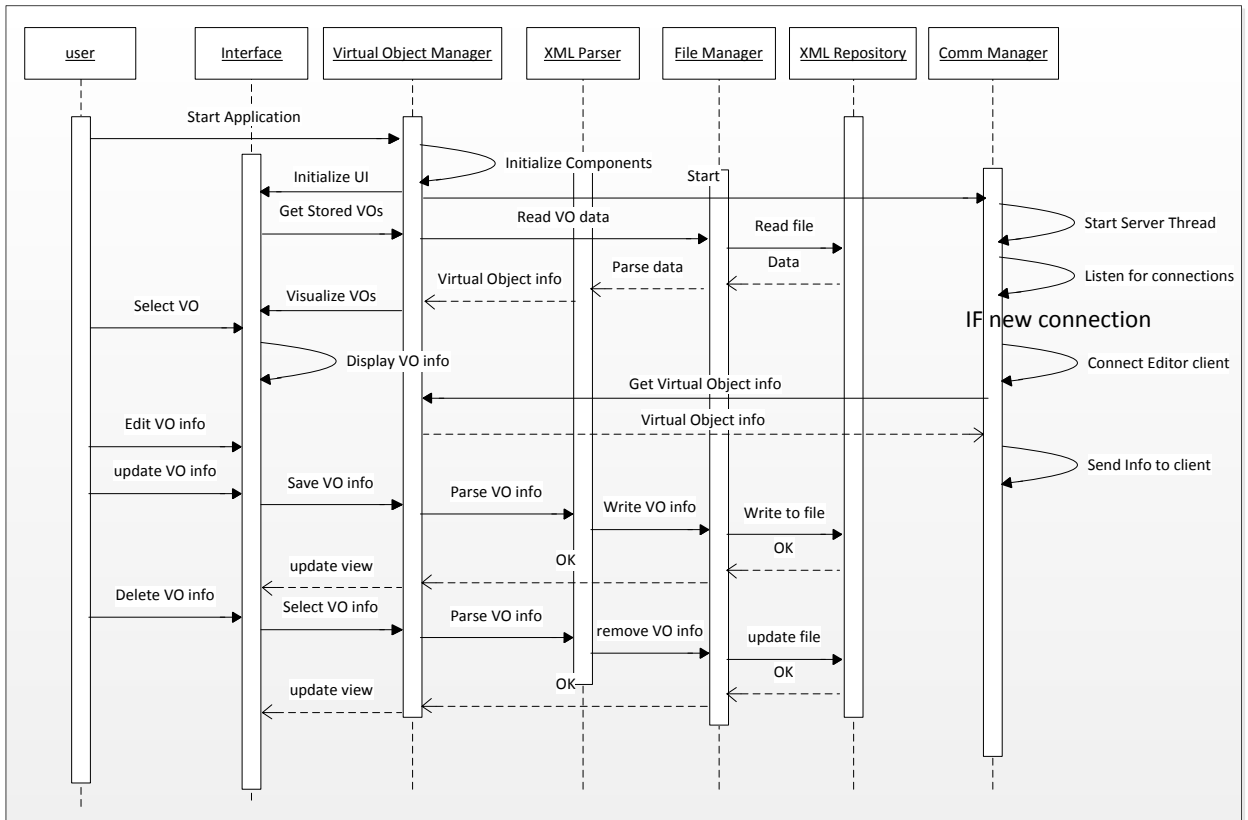


Figure 11: Static structure for Virtual Object Manager

The Figure 12 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 12: 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

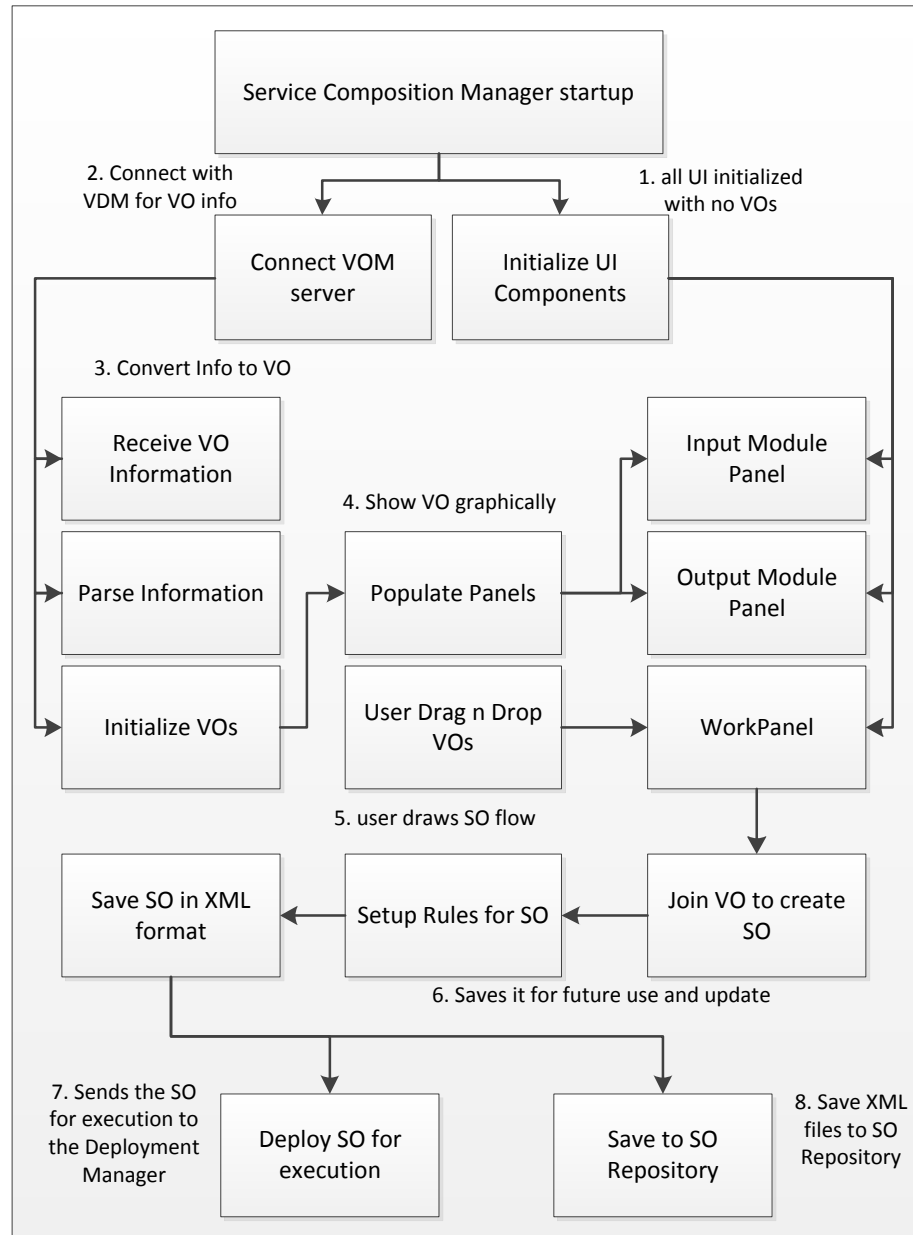
and the File Manager writes it to the XML Repository. The Delete Operation also works in the same fashion. The Communication Manager acts as a server thread which listens for incoming connections from the remote client (SCM). Once a connection request is received, the Communication Manager requests the VOM for VO information which is sent to the client.

## 4.2. Service Composition Layer

Service Composition Manager (SCM) is the main component at the service composition layer. All the other components at the service composition layer are implemented as part of the SCM. The SCM is a DIY graphical designer that is used to compose service objects (SO) from the virtual objects (VO) and the associated information that is received from the Virtual Object Layer. The SCM startup configuration is provided in Figure 13. At startup, the SCM initializes all the user interface components including the Input Module Panel, the Output Module Panel and the Work Panel. Based on the user's choice, a client component sends a connection request to the communication manager at the Virtual Object Layer. If the connection is granted, the SCM receives a list of virtual object information. The information includes resource name, URI, attributes and functionalities which can be remotely executed and information about the visual representation of the device. The information is sent as XML strings and upon reception by the SCM, it is parsed to initialize VOs. The virtual objects created are used to populate the input and output panels so that the user can visualize them and ultimately interact with them to compose them into service objects. Once all the available VOs have been instantiated, the user can interact with the SCM interface to compose Service Object (SO).

As SCM is designed to be a DIY service composer interface, the users perform simple drag-n-drop, click and double-click operations using a mouse pointer to compose a design on the

WorkPanel, which acts as the main drawing canvas. The service composition process includes joining the input and output modules and setting up rules for their interaction and operation.



**Figure 13: Service composition Manager basic configuration**

An example of this process can be visualized as the temperature control service where the temperature sensor is the input VO and the heater is the output VO. The join between the two



VOs will then specify the operational condition such as if the reading from the temperature sensor is less than 15 degree centigrade, the heater should be turned on to keep the spaces properly heated. Once the service composition is completed, the visual SO is converted into XML data and stored at the SO repository. The SOs can also be deployed and executed for testing through the service deployment manager at the Service Composition Layer.

Figure 14 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 output 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.

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.

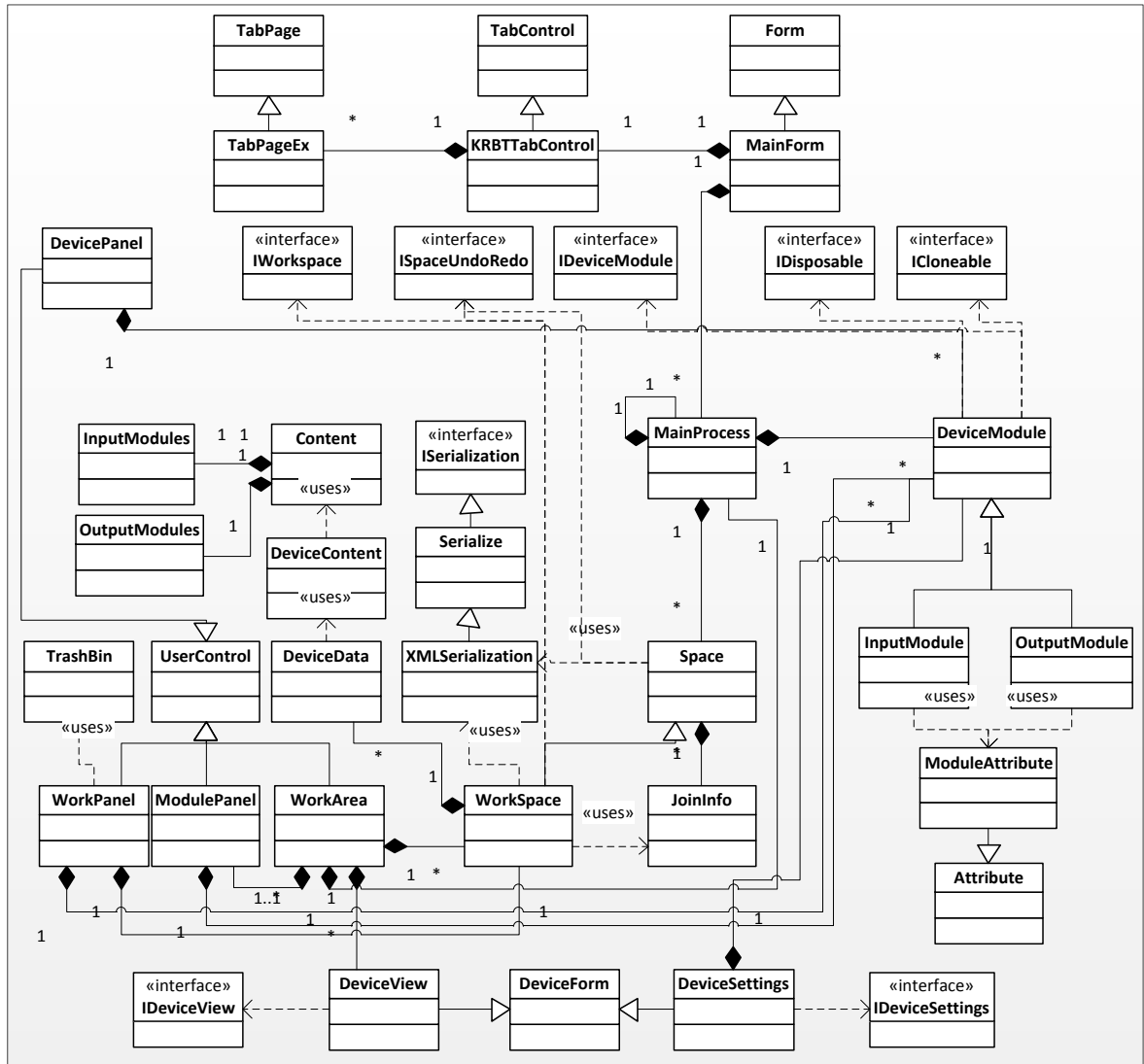


Figure 14: Service Composition Manager static structure

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 MainPage 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.

Figure 15 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. 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.

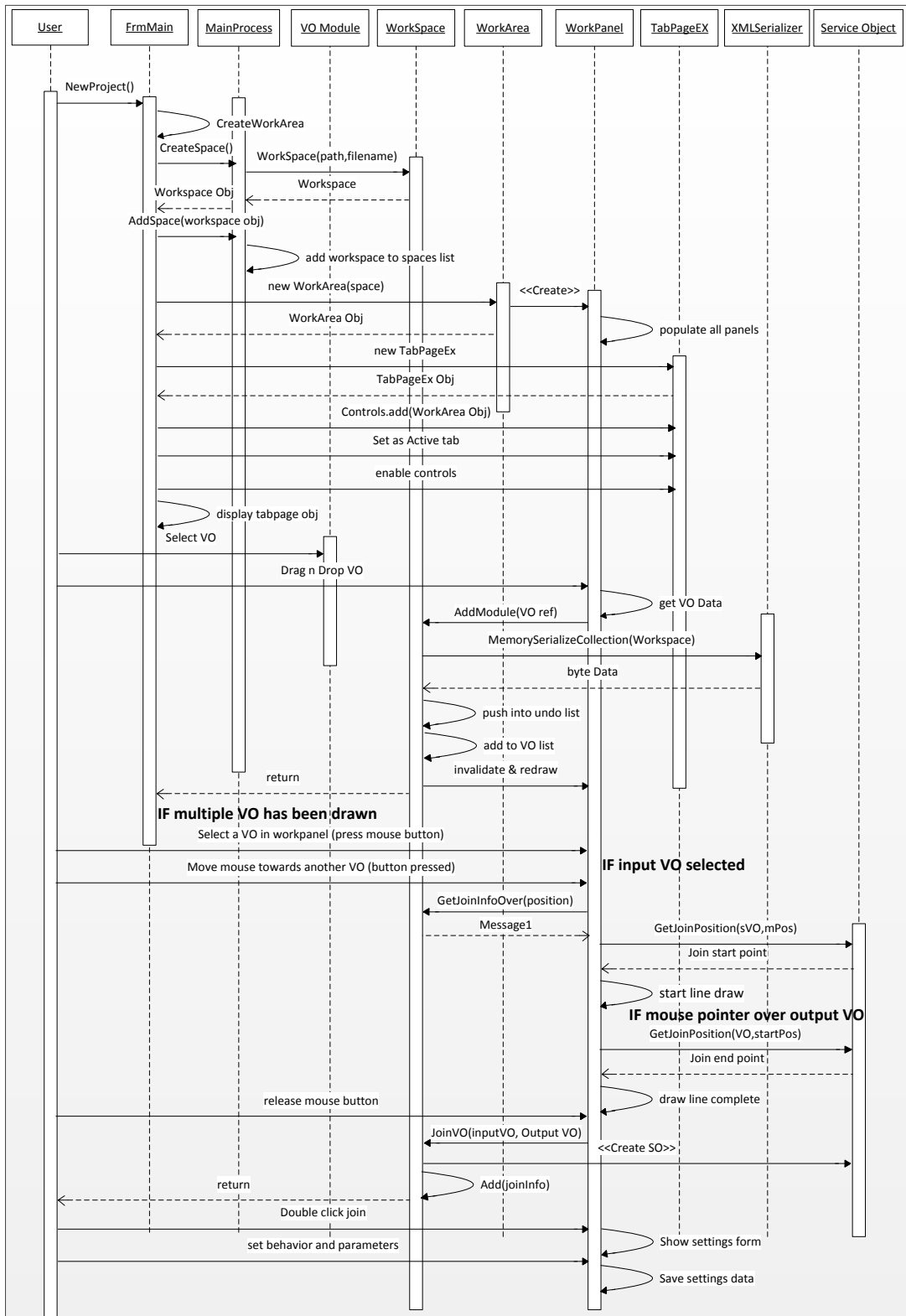


Figure 15 : VO to SO mapping sequence at SCL

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. 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.

### **4.3. Business Process Layer**

BPM Editor is the main component of the Business Process Layer (BPL) which acquires the XML representations of the Service Objects from the SO repository at the Service Composition Layer and represents them as Business Process Modeling Notation (BPMN) for the user. BPMN

is a standardized set of notations used for requirements analysis in software development and for describing processes in business setups. The user utilizes the standardized notations with an intuitive drag-n-drop approach to design IoT processes or applications according to their own needs. Figure 16 shows the startup and operational configuration for the BPM.

The BPM Editor initializes all the user interface components and the communication components at the startup. In the figure, the communication interface is represented by the second step if the SO repository is located at a remote location otherwise a simple IO operation is performed to retrieve the XML files representing the service objects. The user interfaces of the BPM Editor include a BPMN Panel which displays the general notations for creating a business process flow. These general BPM notations include Task notation, Gateway notation, Script and event notations. The service objects retrieved from the repository are represented as BPMN tasks while the other notations provide supporting logic for the creation of BPM flows. These BPMNs are XAML based classes which can be dragged and dropped onto the main canvas by the user. The BPMN Panel is basically populated when the parser module associated with the BPM Editor receives and parses the service objects from SCM repository. The parser module parses the xml files, retrieves the input and output components of the service object along with the operational rules if any, and initializes the BPM notations according to the tasks represented by the service objects.

BPM Editor initializes all the user interface components and the communication components at the startup. In the figure, the communication interface is represented by the second step if the SO repository is located at a remote location otherwise a simple IO operation is performed to retrieve the XML files representing the service objects. The user interfaces of the BPM Editor include a BPMN Panel which displays the general notations for creating a business process flow.





receives and parses the service objects from SCM repository. The parser module parses the xml files, retrieves the input and output components of the service object along with the operational rules if any, and initializes the BPM notations according to the tasks represented by the service objects.

The user then composes a BPM by using drag-n-drop and simple actions such as mouse-clicks and etc. The graphical BPM created at this stage basically represents the operational logic of the IoT process or IoT application. The BPM Editor UI provides editing functionalities such as copy, paste and delete etc. to provide an easy editing environment to the user. Once the BPM composition process is completed by the user, the graphical BPM is converted into an XML representation for storage and later on for loading into the BPM Editor for further updates and changes if the user wishes so.

Figure 17 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 modeling 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. For this purpose, the XmlParser class is utilized to parse the acquired service objects and instantiate the ServiceObject class with the data extracted from each parsed SO representation. The newly created ServiceObject class instances are stored in a list maintained at the AvailableSOs class. The class is implemented as a singleton class and is used by DesignerCanvas and the XmlParser classes.

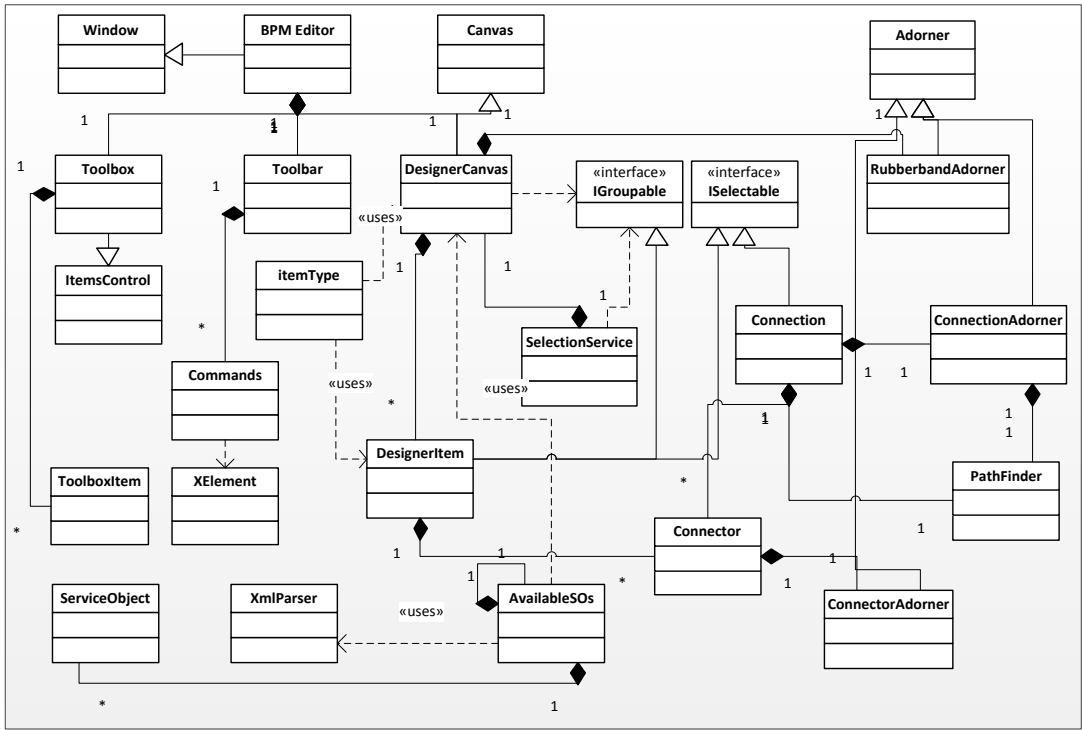
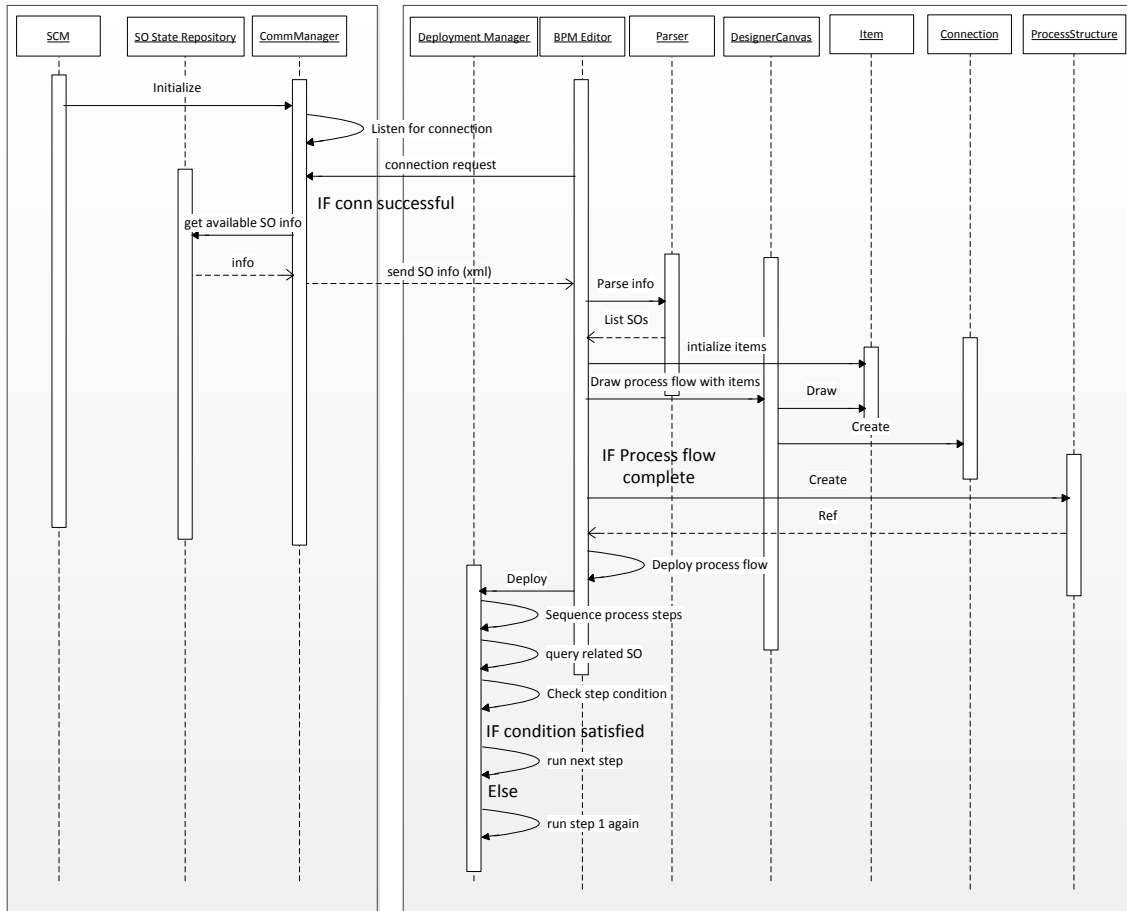


Figure 17: Static structure for BPM Editor

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. Figure 18 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

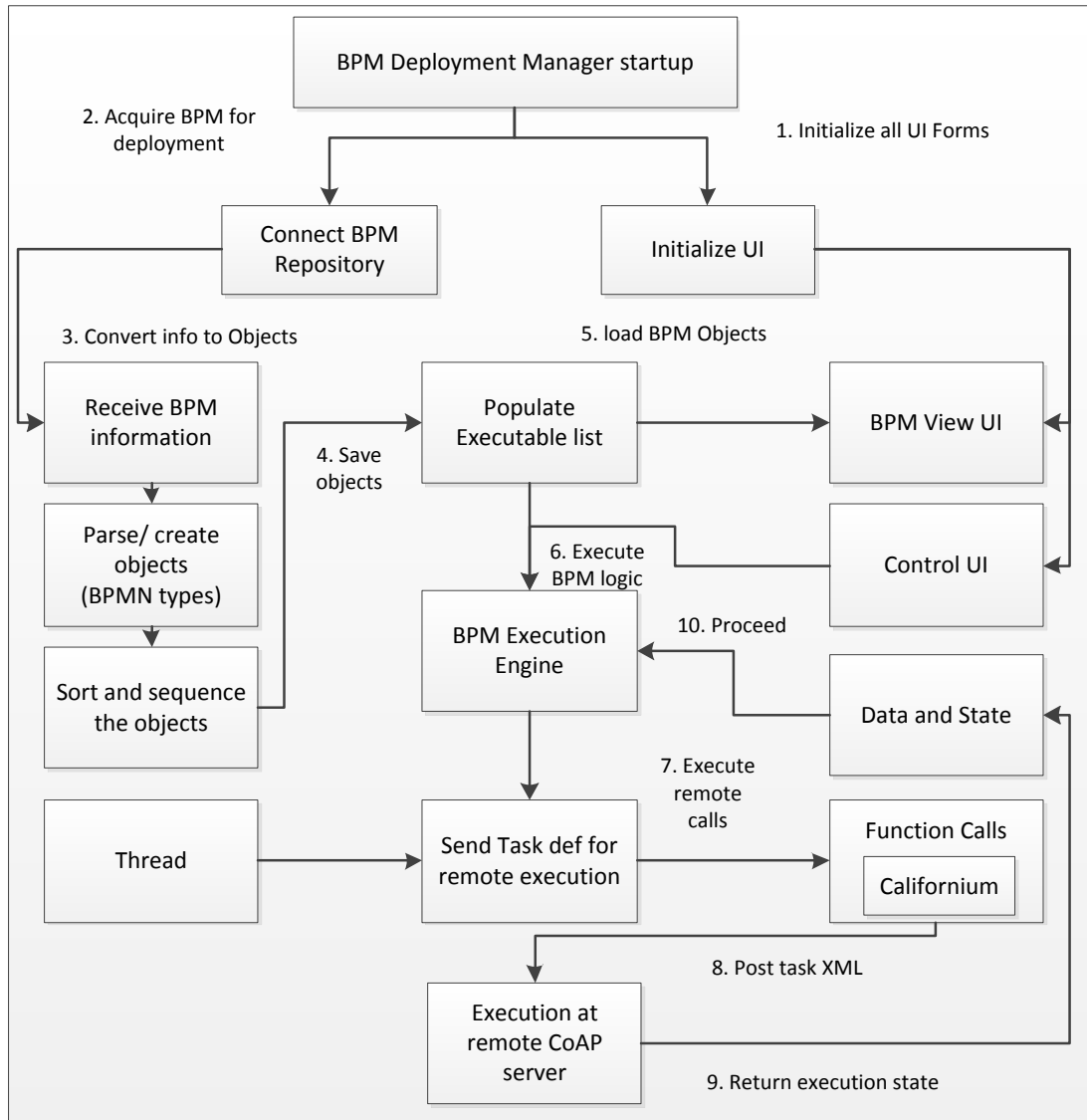
Service Composition Layer are acquired in the form of XML information objects and sent to the Business Process Layer.



**Figure 18: SO to process mapping and execution**

The acquired SOs are then parsed through an XML parser class and 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 modeling 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 Manager at the Business Process Layer.



**Figure 19: BPM Deployment Manager startup and operational configuration**

The BPM Deployment Manager is responsible for the deployment and execution of the BPM models created by the users through the BPM Editor. The BPM Deployment Manager presents a simple user interface which consists of a viewer for the users to view the components of the

BPM that is being deployed. The viewer panel does not visualize the components of the BPM as visual notations rather it lists them in a sequential order of the textual description of these components as specified by the graphical model. This operation is performed by parsing the connections among various BPM notations as part of the graphical model along with the location of each graphical item in the model to create a sequence for the execution of individual tasks.

Once the task execution sequence is created, the list of the tasks as part of the BPM is shown to the user in the view panel. The user can then choose to deploy the model. For the execution of the deployed BPM, the execution engine is responsible to send the XML representation of each task to the concerned remote IoT resource. The execution state or any data is returned to the execution engine which further decides how to proceed with the execution of the BPM. The BPMN Gateways provides branching and execution logic for the process and the BPMN Scripts provide functions such as data processing or network communications which are too costly for the remote IoT resources and thus are executed by the execution engine.

In order to provide a detailed and concise picture of the whole system's design Figure – shows the overall collaboration among various layers with the BPL at the center of the process. The CoAP enabled IoT resources means the physical devices with some sort of CoAP services and functionality which can be invoked remotely via a CoAP client. For these IoT resources, the users create virtual objects by providing the necessary information to the system via the Virtual Object Manager interface. The virtual objects created are utilized by the Service Composition Manager to enable the users to compose Service Objects. The SCM implements simple and intuitive operations such as drag-n-drop and mouse-clicks to provide DIY service object composition environment. A Service Object composed by the user consists of three main components i.e. Input resource, Output resource and the operational rule. The input resource is normally represented by the VO of a sensing device such as thermistor or gas detector etc., the

output resource is represented by a VO of an actuating device such a buzzer or LED and the operational condition is graphically represented by a join between the two resources.

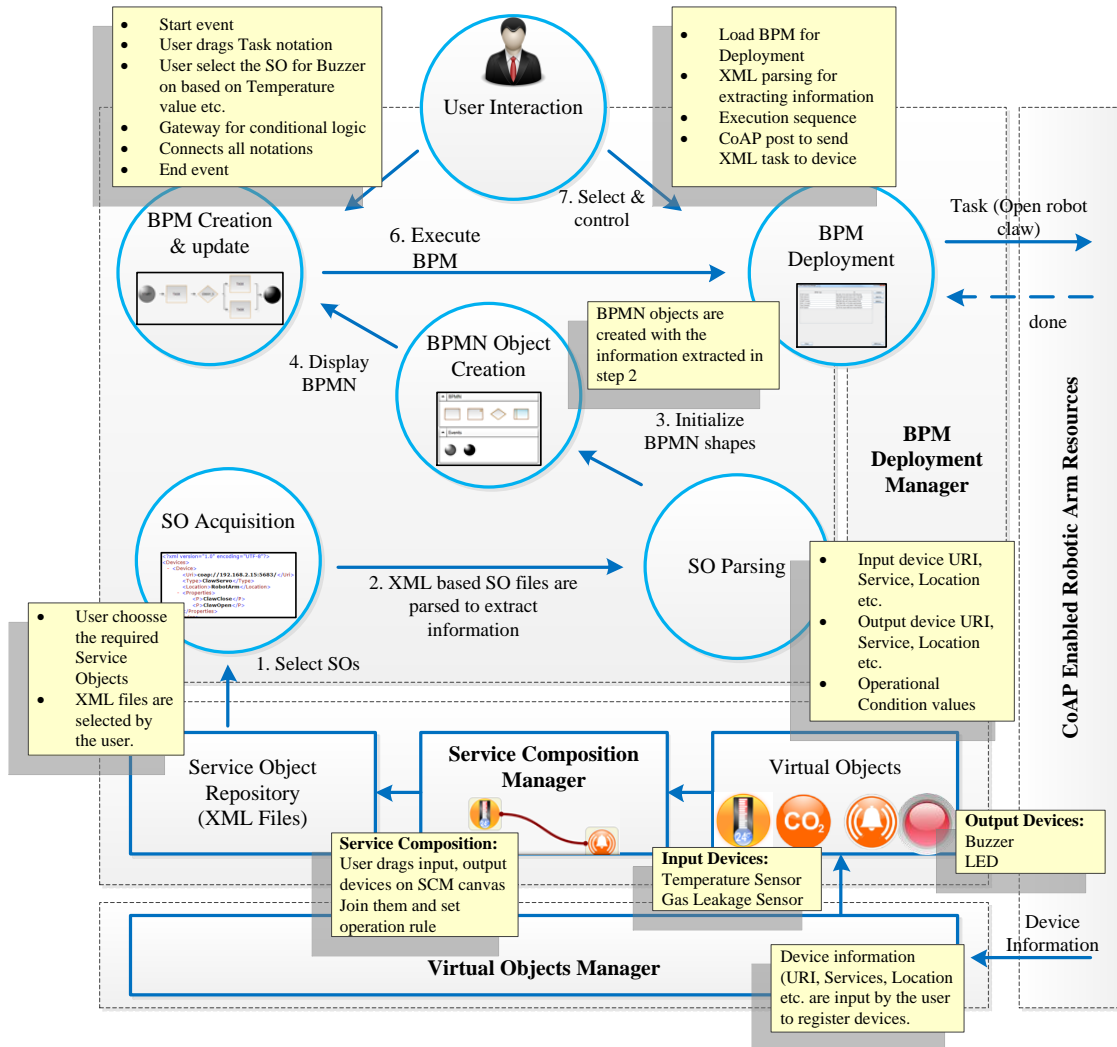


Figure 20: BPM based DIY IoT application development system collaboration from the BPL perspective

The join basically set the trigger condition for the actuating resource based on a certain range value for the input resource. For this purpose simple logical operators are used as part of the service composition process. The graphically composed Service Objects are editable by the user as an XML version of each SO is saved to the Service Object Repository at the Service

Composition Layer. The same repository is used by the Business Process Layer to acquire the definitions of Service Object in order to represent them as Business Process Modeling Notations.

For the conversion of Service Objects into Business Process Modeling Notations (BPMN), the BPM Editor first acquires the relevant Service Objects from the SO Repository at the Service Composition Layer. This has been represented as the first step in the collaboration diagram. The SO are in XML format and the BPM Editor uses the internal parser module to extract the necessary information about the associated resources. This information is used to initialize the BPMN objects which are clone-able graphical representations so that users can utilize them to create their BPM based IoT application model. BPM Editor provides a drag-n-drop based DIY modeling environment for the users to develop process models for their IoT applications and it also enables them to easily edit and change their models. The graphical models are also saved as XML files for this purpose to enable the users to share and update the models using the BPM Editor.

Once a model is completed and the user wants to deploy the IoT application represented by the BPM, the optimized XML version of the model is loaded into the BPM Deployment Manager. The BPM Deployment Manager is separate module at the Business Process Layer which is capable of communicating with the remote IoT devices. The model is parsed and converted into SO based tasks along with the application logic provided by the BPM notations. A sequence of execution is generated based on the graphical composition of the model as created by the user. The execution engine at the BPM Deployment Manager then uses the sequence to send XML based task definitions to the remote devices via CoAP Post calls. The remote devices executes the tasks by executing any relevant CoAP services and the response is sent back to the BPM Deployment Manger where the application logic is evaluated and further execution steps of the process are carried out.

# 5.BPM Based DIY IoT System

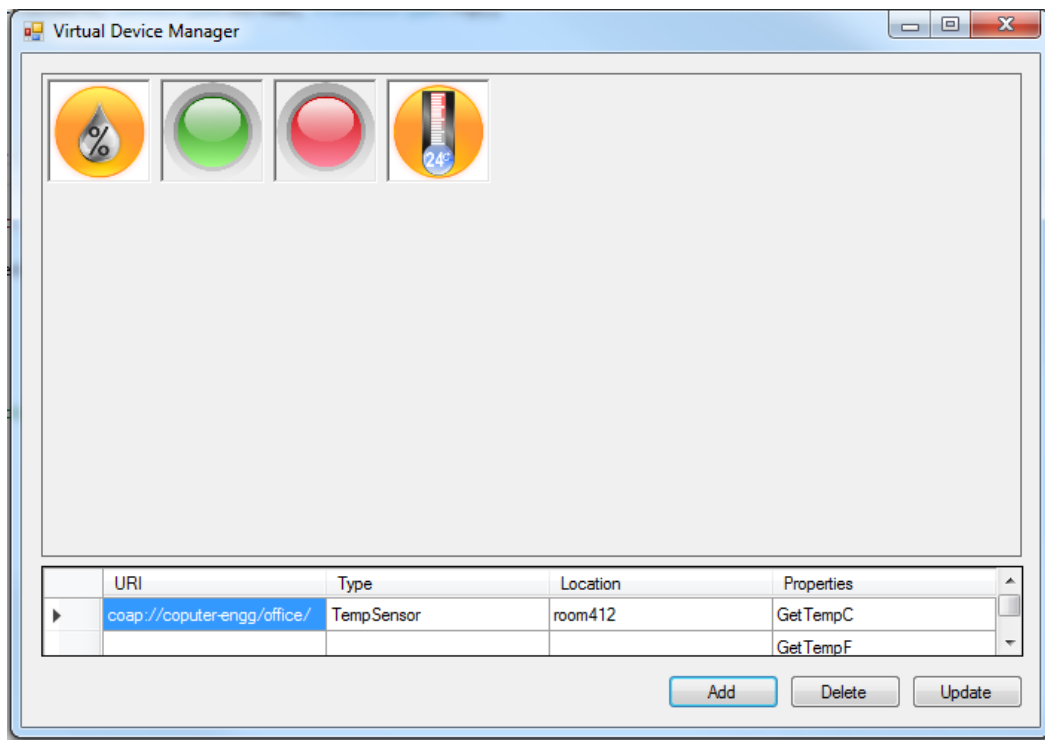
## Implementation

This chapter presents details of the prototypes implementation of the proposed architecture. Each subsection presents the purpose of the respective component, identifying its functionality and provides a description of the implementation tools and technologies used for the development of the specific component as part of the prototype development.

### 5.1. Virtual Object Manager

Figure 21 shows the screen shot for the Virtual Object Manager (VOM) module at the VOL. It also shows the XML representation for the virtual objects' data to be stored. This module enables the users to encapsulate the behavior of their IoT resources. VOM binds the IoT resource's data with a visual representation so that the virtual object can be interacted with and manipulated in a more intuitive way. For this purpose two different approaches have been utilized. The first approach is the manual one where the user provides the details regarding their IoT resources in the form of complete URI, location, type and services etc. Through this approach any device which implements the supported protocols can be added as a resource to the system. The second approach is the one where the Virtual Objects Manager is provided only the URI of the remote device and through an implemented service of CoAP protocol, the module automatically extracts the necessary information from the device. The information would then be utilized by the VOM to create the corresponding virtual objects for the remote device. This approach, however, can only be utilized with specific devices which are equipped with the system specific service implementation for the purpose.





**Figure 21: Virtual Object Manager Interface**

Figure 22 presents the XML representation for virtual objects in the form of device nodes. For saving the space only two nodes have been expanded in the figure which shows the stored information for a gas sensor and an LED device. The URI specifies the protocol and address through which the device can be uniquely identified. The Properties tag specifies the available services which can be executed via the specific resource. A resource can have more than properties (executable functions) which is specified by the sub-tags <P> in the Properties tag. Both URI and an instance of the Properties tag can be utilized to provide a uniquely addressable function of the remote resource.

The Location tag is used for specifying the location of the remote IoT resource. This tag and more information regarding the owner or allowed users etc. can be considered for future studies related to the security of the system.

```

<?xml version="1.0" encoding="UTF-8"?>
- <Devices>
  - <Device>
    <Uri>coap://192.168.2.15/</Uri>
    <Type>GasSensor</Type>
    <Location>Room423</Location>
    - <Properties>
      <P>GasReading</P>
    </Properties>
  </Device>
  + <Device>
  + <Device>
  + <Device>
  + <Device>
  - <Device>
    <Uri>coap://192.168.2.15/</Uri>
    <Type>RedLED</Type>
    <Location>room412</Location>
    - <Properties>
      <P>On</P>
      <P>Off</P>
      <P>Blink</P>
    </Properties>
  </Device>

```

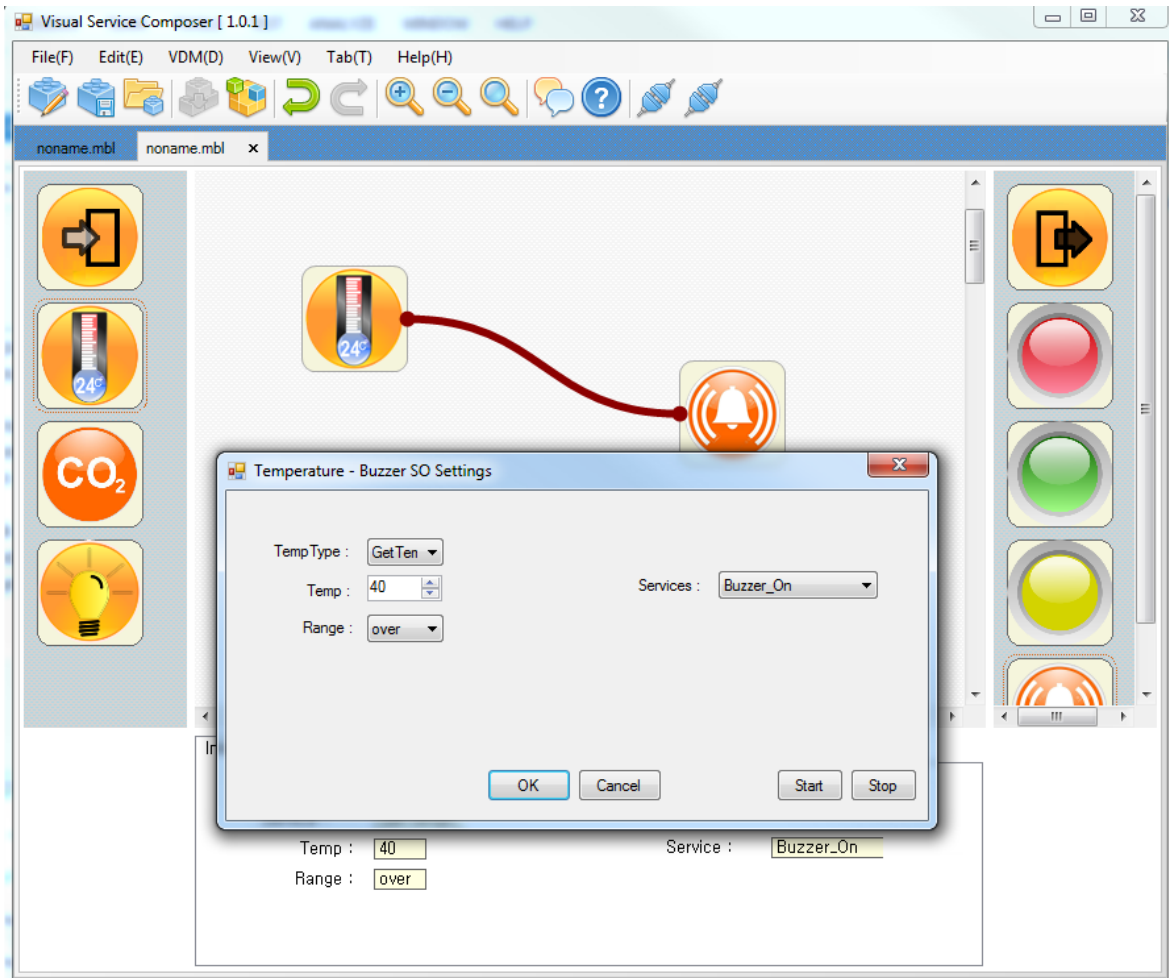
Figure 22: XML representations for virtual object storage at VOL

## 5.2. Service Composition Manager

Service Composition Manager (SCM) is the main module at the Service Composition Layer. A snap shot of the SCM interface is shown in Figure 23. The main objective of this module is to allow the user to easily visualize, interact with and manipulate the virtual objects created by VOM. The SCM is developed in C-Sharp environment as a Windows application.

For this purpose the sensor VOs and actuator VOs are separately represented as input and output modules. These modules can be directly dragged and dropped on a canvas through the basic Windows OS mouse events. The VO modules on the canvas can then be connected with simple joining lines which represents connection between the input and output VOs. Finally, the user can set the rules of operations for the joined virtual objects. For this purpose simple and intuitive approach has been implemented. The user can double-click each VO to display the

settings form for the VO or the Join between two VOs can be double-clicked by the user to display the properties settings form for the complete service object. There the user can specify the values of attributes, set ranges for the conditional operations and choose conditional operators for the evaluation of conditional logic.



**Figure 23: Service composition manager interface**

The SCM interface provides user-centric approach for the development of service objects based on the virtual representation of IoT resources. The interface is implemented with standard toolstrip for efficient editing and composition of service objects to enable user efficiency and better DIY environment. Menus and shortcuts have been implemented for providing the user

with standardized editing and composition functions such as cut, copy, paste, detailed viewing of the graphical models and commenting the models for easy recognition of implemented functionalities. Following is the main list of operation implemented as the toolbar.

- New Project: creates a new service composition project when clicked by the user.
- Save: Saves the created graphical model for SO in the form of XML document. The file is saved at a default location if not mentioned otherwise by the user.
- Open: this button display the open file dialog and the user can select pre-existing SO files to open in the editor.
- Undo: Cancels an immediate action performed by the user.
- Redo: Performs again an immediate cancelled action by the user.
- Zoom In: Enlarges the contents of the work area by a predefined factor every time the button is pressed. The zoom ability is limited to a predefined level.
- Zoom Out: the opposite of zoom in and reduces the size of the contents of the work area.
- Restore: Levels out the zooming effects of zoom in or out and restore to size of the contents in the work area.
- Comments: Toggles the show comments functionality on or off. Comments are some additional text to explain a VO in the service composition model.
- Help: opens the help window for the SCM.

The joining of input and output VOs in the SCM creates a service object (SO). These SOs are stored as XML documents which separately represent each input and output VO as part of a service object. The connections between these VOs are also represented in the XML document in the form of a join node which specifies the source and sink entities for the connection. Hence the VOs can be stored, opened and updated according to the user requirements.

```

<?xml version="1.0" encoding="UTF-8"?>
<Workspace xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema
instance">
- <Devices>
- <DeviceModule range_T="over" service="GetTempC" temp="20" xsi:type="Temperature">
  <ID>84c9b637-d104-4bf1-b324-1e5b1c68d89c</ID>
  <CoAPServices>GetTempC,GetTempF,</CoAPServices>
  <Uri>coap://192.168.2.15:5683/</Uri>
  <Location>room412</Location>
  - <Position>
    <X>299</X>
    <Y>147</Y>
  </Position>
</DeviceModule>
- <DeviceModule service="RedLED_Blink" xsi:type="RedLED" time_L="1">
  <ID>847fadf8-1990-45d4-9900-6bd916aff7d7</ID>
  <CoAPServices>RedLED_On,RedLED_Off,RedLED_Blink,</CoAPServices>
  <Uri>coap://192.168.2.15:5683/</Uri>
  <Location>room412</Location>
  - <Position>
    <X>605</X>
    <Y>209</Y>
  </Position>
</DeviceModule>
+ <DeviceModule service="GetGasReading" xsi:type="GasSensor" range_G="over" detectedValue="150">
+ <DeviceModule service="Buzzer_On" xsi:type="Buzzer">
+ <DeviceModule service="GetGasReading" xsi:type="GasSensor" range_G="under" detectedValue="150">
+ <DeviceModule service="Buzzer_Off" xsi:type="Buzzer">
</Devices>
- <Joins>
  <JoinInfo OutputType="847fadf8-1990-45d4-9900-6bd916aff7d7" InputType="84c9b637-d104-4bf1-b324-
1e5b1c68d89c"/>
  <JoinInfo OutputType="af8fd0f8-ab2a-4d44-a4f9-969935c943d5" InputType="641cc9a1-1d79-4514-923e-
5db7a3b8f714"/>
  <JoinInfo OutputType="af8fd0f8-ab2a-4d44-a4f9-699935c943d4" InputType="641cc9a1-1d79-4514-923e-
4eb7a3b8f713"/>
</Joins>
<FilePath>C:\Users\sohail\Documents\SCDesigner</FilePath>
<FileName>tempCRedLedBlink.mbl</FileName>
</Workspace>

```

Figure 24: XML representation of service objects at SCL

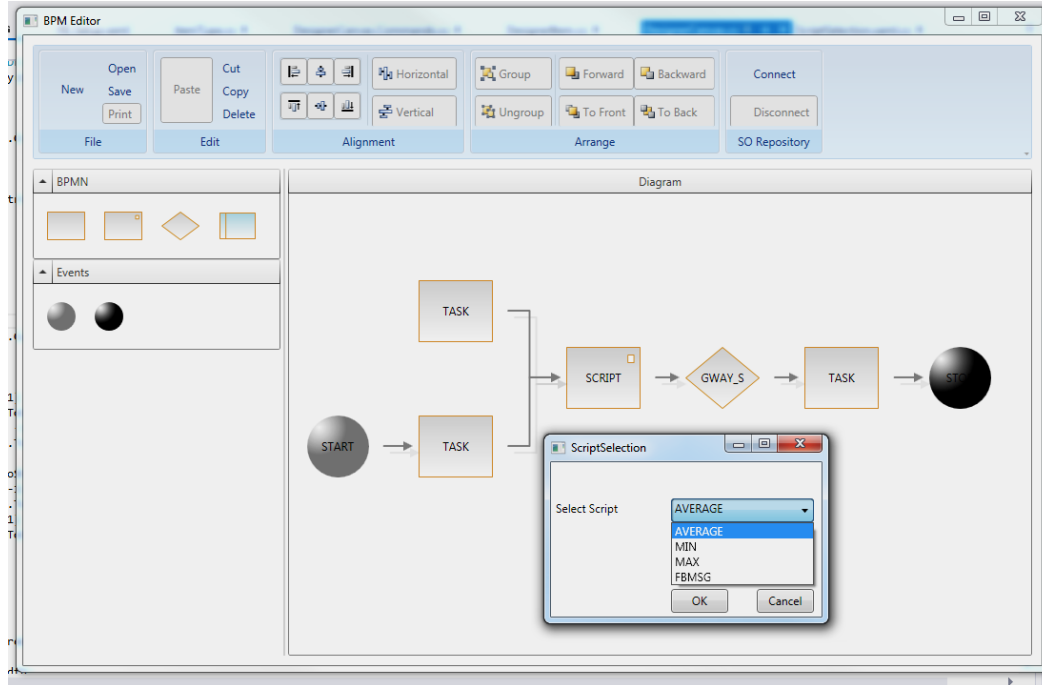
Figure 24 shows a sample of the XML documents representing Service Objects (SO). Each SO is represented by the JoinInfo tag where unique identifiers specifies the input and output device associated with the specific service object. The same identifiers are used in the DeviceModule tags as shown in the figure. The DeviceModule tags encapsulate the information about each resource as part of the saved service objects. This information include the device type, complete URI to access the remote IoT resource, the service name selected by the user at SCM to be executed along with the operational conditions as part of the SO for the specific device and the location of the remote resource. The list of names encapsulated in the tag named CoAPServices represents all the services supported by the specific device. This list is included in

the service object definitions for enabling the SCM to de-serialize the XML files and graphically render it with complete information if the user wishes to update the SO later. The location information would further be utilized for security and user rights allocation in the future studies.

### 5.3. BPM Editor

Business Process Model Editor is responsible for the representation of the Service Objects composed at the SCM as Business Process Modeling Notations (BPMN). The main interface for BPM Editor is shown in Figure 25. This component along with the Deployment Manager constitutes the Business Process Layer of the proposed architecture. The aim of providing a BPMN based representation of the Service Objects is to provide a DIY interface for anyone with the basic knowledge of the notations to create and deploy their IoT applications. It also eliminates the requirement of any programming skills because the user just has to create a graphical model and it is directly deployed as an IoT application.

The main interface is divided into three main areas. The first is the BPMN Panel on the left side. This panel groups the various implemented notations in the form of a shape palette. Each shape in the BPMN panel is an instance of the XAML based class which is derived from a common class termed as `ToolboxItem`. This class provides the cloning attributes to each shape derived from it and thus enabling the shapes to be dragged and dropped by the users. The main BPMN notations which have been implemented as part of this prototype system include the Task, Script, Gateway and the swimlane notation. In the event section of the BPMN panel, Start and Stop events have been implemented. These event notations specify the start and finish of a process represented by the graphical BPM.



**Figure 25: BPM Editor interface**

The top area of the interface provides an application level toolbar containing editing functionalities necessary for shapes and model composition. The toolbar is implemented in a standardized way to resemble the toolbars provided by well-known and common applications such as Microsoft Word and Microsoft PowerPoint. This feature enables the users to easily recognize various editing functionalities through general knowledge and helps in providing a DIY environment for model development. Following is a list of the major groups of functionalities implemented as part of the application toolbar provided by BPM Editor.

- File group: Provides common commands related to new project creation and file management such as open file, save file etc. The file open command utilizes the XML parser to retrieve information to graphically render the BPMN of the canvas while the save command does the opposite.

- **Edit Group:** This group provides editing functionalities such as cut, copy and paste etc. for the user to efficiently draw the graphical model.
- **Alignment:** This group provides commands for adjustment in the alignment of various shapes or group of shapes. In this scenario, the alignment commands are used to adjust the location of various BPMN items in reference to other items.
- **Arrange:** This group while not implemented completely, provides the commands such as grouping of various BPMN item into a single shape. A grouped object is created in order to provide easy manipulation and alignment of more than one shape (BPMN items).

For a new BPM project to be created by the user, the user must import the necessary Service Objects that will be part of the specific BPM. For this purpose, the BPM Editor provides a simple interface where the user can connect to the SO repository at the Service Composition Layer and select the XML files for the necessary SOs. The SOs are read by the FileManager and parsed by the XMLParser to extract the SO components and attributes. A list of the basic descriptions for the service objects is created which is associated with the BPMN task shape when the user double-clicks it in the editor canvas area. Service objects are represented as Tasks and the user can create their model via the same drag-n-drop approach. The sequence of operation is created by connecting the notations via arrow objects and the same arrow objects are utilized to capture information regarding the inputs and outputs of notations in the model. As mentioned earlier, the conditional logic of the process is implemented using the BPMN gateway notations while processor intensive tasks and remote communication tasks which are not suitable to be executed on the remote IoT resources are represented by the Script notations. the Script notation has been provided with a list of scripts from which the user can choose to manipulate or process the data.



```

<?xml version="1.0" encoding="UTF-8"?>
- <Root>
  + <DesignerItem>
    - <DesignerItem>
      <Left>169</Left>
      <ID>e357a8d5-ee2c-4a39-ab54-05500ff81226</ID>
      <ItemType>TASK</ItemType>
      <InDevType>GasSensor</InDevType>
      <OutDevType>RedLED</OutDevType>
      <InputUri>coap://192.168.2.15:5683/</InputUri>
      <OutputUri>coap://192.168.2.15:5683/</OutputUri>
      <InputService>GetGasReading</InputService>
      <OutputService>RedLED_On</OutputService>
      <SoLocation>room412</SoLocation>
      <Condition_Op>over</Condition_Op>
      <Condition_val>150</Condition_val>
    </DesignerItem>
    - <DesignerItem>
      <Left>312</Left>
      <ID>cdb92531-21cd-47c1-8acf-444d81092677</ID>
      <ItemType>GWAY_S</ItemType>
      <InDevType/>
      <OutDevType/>
      <InputUri/>
      <OutputUri/>
      <InputService/>
      <OutputService/>
      <SoLocation/>
      <Condition_Op>BOOL</Condition_Op>
      <Condition_val>1</Condition_val>
    </DesignerItem>
  + <DesignerItem>
  + <DesignerItem>
  + <DesignerItem>
  - <Connection>
    <SourceID>f9a69aa8-9843-44e4-81b1-df5d93ba1bca</SourceID>
    <SinkID>e357a8d5-ee2c-4a39-ab54-05500ff81226</SinkID>
    <SourceConnectorName>Right</SourceConnectorName>
    <SinkConnectorName>Left</SinkConnectorName>
    <SourceArrowSymbol>None</SourceArrowSymbol>
    <SinkArrowSymbol>Arrow</SinkArrowSymbol>
    <zIndex>2</zIndex>
  </Connection>
  + <Connection>
  + <Connection>
  + <Connection>
  + <Connection>
  + <Connection>
</Root>

```

Figure 26: XML representation of a stored BPM model

The BPM model created by the users via the BPM Editor is stored as an XML file. This file is the direct serialization of graphical notation and the associated information such as location on

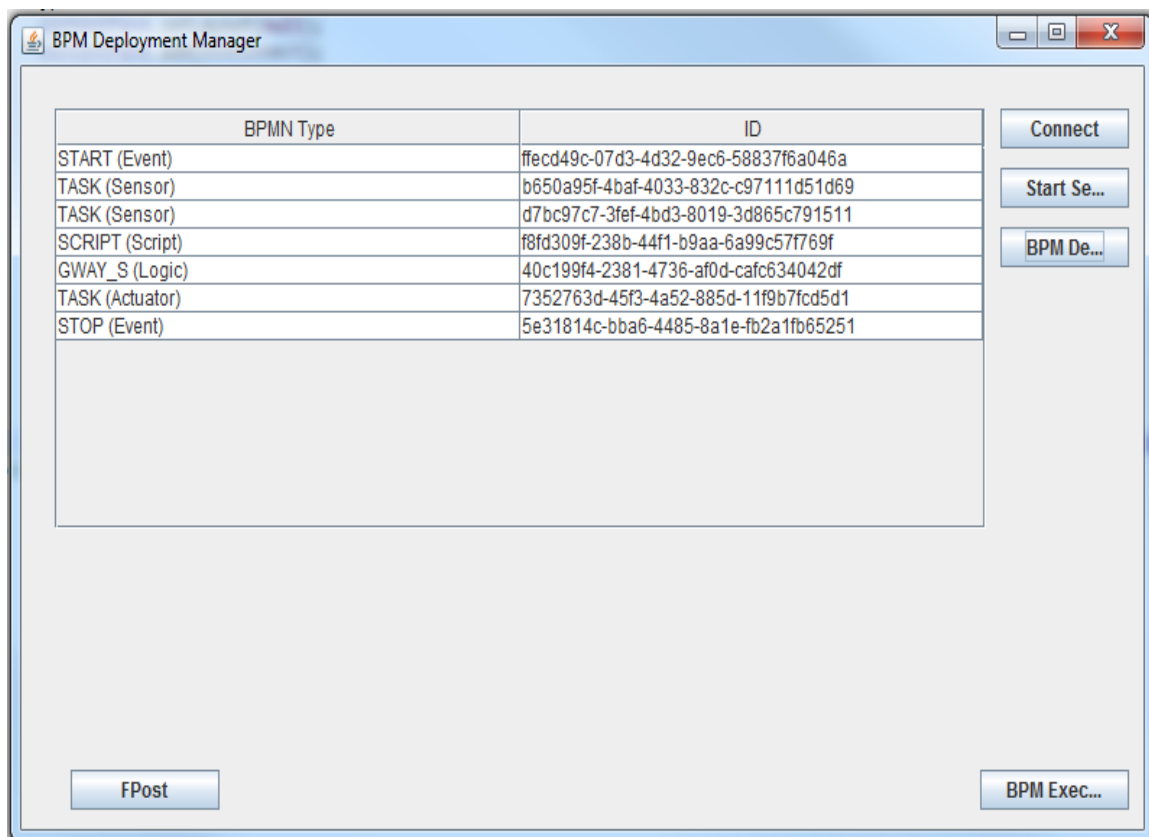
the canvas, identifiers etc. If the user chooses to reload a previously created BPM into the Editor for updates or changes, the file contains all the information to enable the BPM Editor to load and re-render the same graphical model as created by the user.

Although the XML file mentioned above is very important from the perspective of editing and updating the graphical models, it contains too much of unnecessary information from the perspective of BPM deployment and execution. For this purpose, every time a graphical BPM is stored by the user, another optimized version of the XML file is created with the sole purpose to be utilized by the BPM Deployment Manager. This XML files does not contain any information regarding the graphical rendering of the BPM and only provides information necessary for the deployment and execution of the process represented by the BPM. The XML sample is shown in Figure 26 representing tasks and other notations as DesignerItem objects. A DesignerItem tag in the figure completely represents the information encapsulated by a single BPM notation. In the figure first task represents a Task notation which encapsulates the complete information regarding a service object. The information include the names of the input, output devices associated with the SO, the complete URIs of the services for both the devices and the operational conditions for the execution of the SO. The file also includes connection objects to keep track of the source and sink items in the model and hence helps in identifying the correct sequence and execution order of the process.

## 5.4. BPM Deployment Manager

BPM Deployment Manager is the second component of the Business Process Layer. It provides the execution engine for the BPM created via the BPM Editor. Figure 27 shows the main interface for the BPM Deployment Manager. The component itself is developed using java so that I can be interoperable with several technologies. For this specific prototype as the

physical devices are CoAP based IoT resources based on Intel Edison platform, BPM Deployment Manager utilizes the Californium framework to communicate with the remote IoT resources in order to execute the process represented via the BPM. The interface is intentionally kept very simple with a viewer for the user to visualize the steps of the deployed BPM and a few controls to enable the user to load BPM from the repository and to control the execution of the BPM.



**Figure 27: Deployment manager interface with a loaded BPM**

As mentioned in the previous section, that a process model representing the interaction among various IoT resources is created by the user via the BPM Editor and stored as an XML document. These XML documents can be loaded in the Deployment Manager to be executed. The BPM in its XML format is loaded into the BPM Deployment Manager. The file is first

parsed to extract all the executable entities as represented by the BPM notations. Based on the connection between the notations, the executable entities are sorted and sequenced so that the final execution of the process is in synch with the original graphical BPM created by the user. The execution steps are then displayed in the viewer part of the interface as shown in **Figure 27**.

```

Manager (1) [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (May 17, 2016, 6:33:05 PM)
<InDevType>Temperature</InDevType>
<OutDevType>RedLED</OutDevType>
<InputUri>coap://192.168.2.15:5683/</InputUri>
<OutputUri>coap://192.168.2.15:5683/</OutputUri>
<InputService>GetTempC</InputService>
<OutputService>RedLED_Blink</OutputService>
<SoLocation>room413</SoLocation>
<Condition_Op>over</Condition_Op>
<Condition_val>1</Condition_val>
</DesignerItem>
coap://192.168.2.15:5683/
GetTempC
Element = DesignerItem
printing Node: <DesignerItem>
<Left>452</Left>
<ID>fbad380f-f41b-462c-86dd-1855189f6787</ID>
<ItemType>TASK</ItemType>
<InDevType>Temperature</InDevType>
<OutDevType>RedLED</OutDevType>
<InputUri>coap://192.168.2.15:5683/</InputUri>
<OutputUri>coap://192.168.2.15:5683/</OutputUri>
<InputService>GetTempC</InputService>
<OutputService>RedLED_Off</OutputService>
<SoLocation>room412</SoLocation>
<Condition_Op>over</Condition_Op>
<Condition_val>1</Condition_val>
</DesignerItem>
coap://192.168.2.15:5683/
GetTempC
Element = DesignerItem

ordering the execution sequence
Ordering completed..
time taken = 52
Execution Engine started..
contacting coap resource.. coap://192.168.2.15:5683/PostTaskXml
May 17, 2016 6:33:35 PM org.eclipse.californium.core.network.config.NetworkConfig createStandardWithFile
INFO: Loading standard properties from file Californium.properties
May 17, 2016 6:33:35 PM org.eclipse.californium.core.network.CoapEndpoint start
INFO: Starting endpoint at 0.0.0.0/0.0.0.0
May 17, 2016 6:33:35 PM org.eclipse.californium.core.network.EndpointManager createDefaultEndpoint
INFO: Created implicit default endpoint 0.0.0.0/0.0.0.0:64114
4.04
{}

Service execution results = False
Single condition gateway executing..
Boolean Gateway executing.. inputval = 0.0
contacting coap resource.. coap://192.168.2.15:5683/PostTaskXml
2.00
{}

Service execution results = True

```

**Figure 28: Deployment manager execution results**

The entities are stored in a list which is sorted in accordance with the flow of the process model. The list is then iterated and each entity is executed based on its attributes and behavior.

For task related to remote IoT resources, the XML representation of the complete service object is sent to the corresponding CoAP server. This transfer is done through a default CoAP post service which is implemented as part of each CoAP server.

Once the CoAP server receives the XML representation of a Service Object, the CoAP services, local or remote, are executed using the Californium framework. The response based on the complete execution of the Service Object is then sent back to the Deployment Manager, where it is utilized to evaluation the conditional gateways or provided as inputs to the other BPM notations directly connected with the specific task notation. Scripts implemented as part of the Deployment Manager are executed by the manager itself while the data is provided by other entities such as remote IoT resources. Figure 28 shows the execution of a business process model via the Deployment Manager. The XML description of each service object has been displayed separately for illustration and the responses received from the remote CoAP server has been shown before utilized to further execute the process.

**Table 1: System configuration for performance analysis**

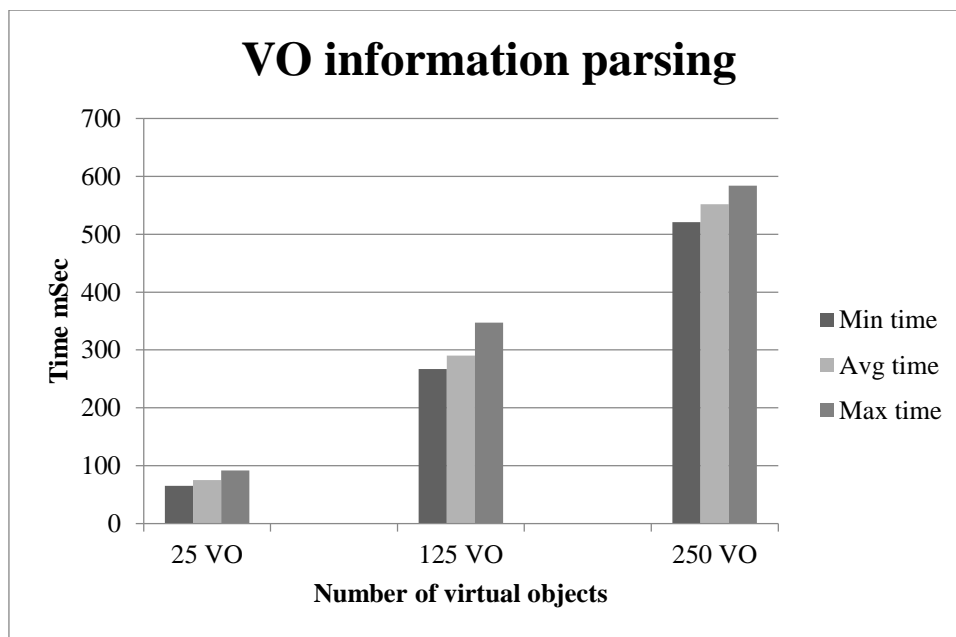
<b>System</b>	<b>Specifications</b>
CPU	Intel® Xeon® CPU E3-1230 V2 @ 3.30GHz
RAM	8 GB
Graphics	NVIDIA GeForce 9600 GT
Operating System	Window 7 Professional
Execution Environment	Visual Studio 2013 Community with .Net Framework 4.5.2

## 5.5. Performance Analysis

A basic information retrieval and parsing based performance analysis has been performed at the three levels of the implemented system. These levels include the Service Composition Manager, The BPM Editor and the BPM Deployment Manager. Same system configuration has

been utilized for performance analysis of all the three layers and the information is presented in Table 1.

The SCM acquires the VO information from the VOM and parses that information to instantiate the corresponding visual and interactive virtual representations of the devices represented by the VOs. This process has been analyzed for performance and the results have been displayed in Figure 29.

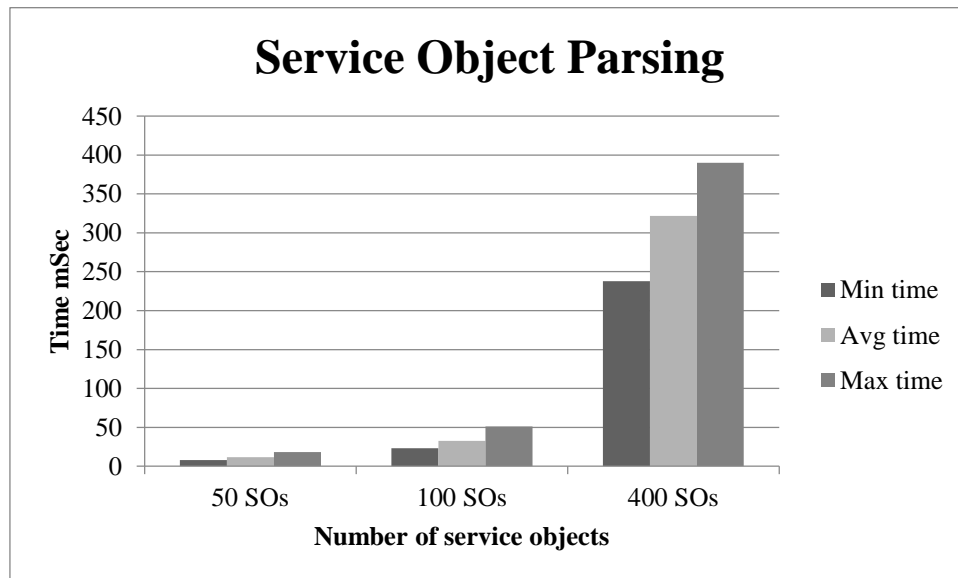


**Figure 29: Performance analysis graph at SCM**

For this analysis, three sets of 25, 125 and 250 VO information was provided to the SCM and each set of VO information was allowed to be parsed by the SCM ten times at randomly selected system resource utilization levels. The graph in Figure 29 presents the minimum, average and maximum time in milliseconds taken by the SCM to parse and instantiate the corresponding visual representations of the virtual objects.

For 25 VO information set, the minimum time taken in the ten iterations was recorded to be 65 milliseconds, averaging at the 75.3 milliseconds and the maximum delay was recorded to be

92 milliseconds. For the 125 VO information set, the minimum time taken in the ten iterations was recorded to be 267 milliseconds, averaging at the 290.5 milliseconds and the maximum delay was recorded to be 347 milliseconds. For the 250 VO information set, the minimum time taken in the ten iterations was recorded to be 521 milliseconds, averaging at the 551.9 milliseconds and the maximum delay was recorded to be 584 milliseconds. The BPM Editor acquires the Service Object information from the SCM and parses that information to instantiate the corresponding visual and interactive BPMN based representation of each SO. This process has been analyzed for performance and the results have been displayed in Figure 30.



**Figure 30: Performance analysis graph for BPM Editor**

For this analysis, three sets of 50, 100 and 400 SO information was provided to the BPM Editor and each set of SO information was allowed to be parsed by the BPM Editor ten times at randomly selected system resource utilization levels. The graph in Figure 30 presents the minimum, average and maximum time in milliseconds taken by the BPM Editor to parse and instantiate the corresponding visual business process modeling notations for the service objects.

For 50 SO information set, the minimum time taken in the ten iterations was recorded to be 8 milliseconds, averaging at the 11.4 milliseconds and the maximum delay was recorded to be 18 milliseconds. For the 100 SO information set, the minimum time taken in the ten iterations was recorded to be 23 milliseconds, averaging at the 32.5 milliseconds and the maximum delay was recorded to be 51 milliseconds. For the 400 SO information set, the minimum time taken in the ten iterations was recorded to be 238 milliseconds, averaging at the 321.7 milliseconds and the maximum delay was recorded to be 390 milliseconds.

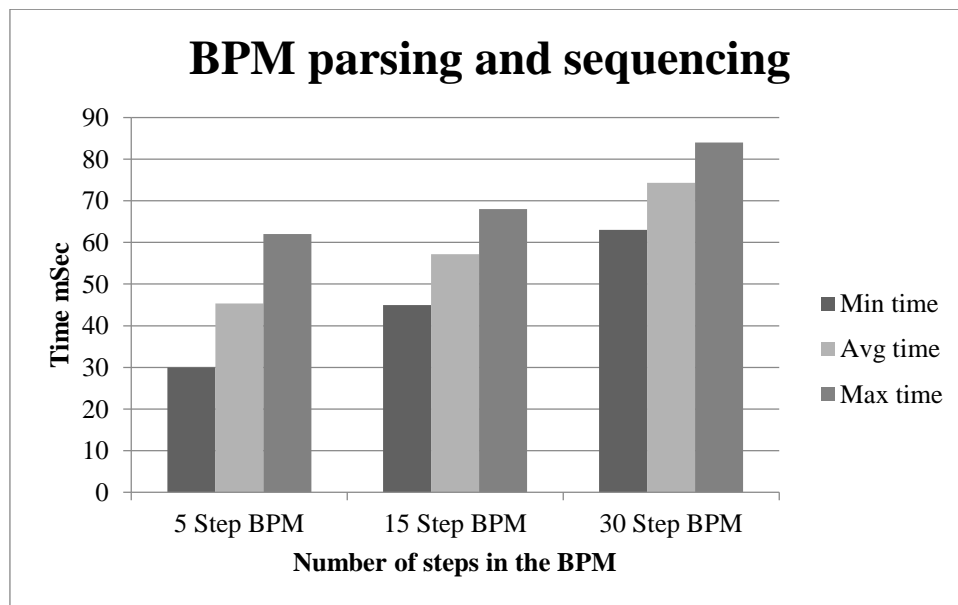


Figure 31: Performance analysis graph for BPM Deployment Manager

The BPM Deployment Manager acquires the process object from the BPM Editor in order to execute it. It first parses that information to create relevant executable objects and then sequence them according to the order set in the graphical Business Process Model created by the user. This process has been analyzed for performance and the results have been displayed in Figure 31.



For this analysis, three different BPM with 5, 15 and 30 steps process were provided to the BPM Deployment Manager and each BPM was allowed to be parsed by the BPM Deployment Manager ten times at randomly selected system resource utilization levels. The graph in Figure 31 presents the minimum, average and maximum time in milliseconds taken by the BPM Deployment Manager to parse the BPM into executable objects and adjust the sequence of execution accordingly.

For the 5 step BPM, the minimum time taken in the ten iterations was recorded to be 30 milliseconds, averaging at the 45.4 milliseconds and the maximum delay was recorded to be 62 milliseconds. For the 15 step BPM, the minimum time taken in the ten iterations was recorded to be 45 milliseconds, averaging at the 57.2 milliseconds and the maximum delay was recorded to be 68 milliseconds. For the 30 step BPM, the minimum time taken in the ten iterations was recorded to be 63 milliseconds, averaging at the 74.4 milliseconds and the maximum delay was recorded to be 84 milliseconds.

## 6. Usability Study for Robotic Arm Use-case

### 6.1. IoT in Industrial Robotics

The most prominent areas of interest for IoT in the recent few years include the ‘smart industry’, which under the term of Industry 4.0 can be defined as the development of intelligent production system and connected production sites. [44]. Smart manufacturing is another term given to the application of IoT and related technologies for full integration and collaboration of manufacturing systems for real-time response to meet the changing demands and requirements in a factory and the customer’s needs.[45][46] . Apart from smart control, the utilization of IoT technologies in production logistics enables a multi-level dynamic adaptability attribute to the system [47].

IoT technologies have also evolved according to the needs of the manufacturing industry. Industrial IoT (IIoT) is the motivation and drive for the industrial up-gradation. It enables the continuous acquisition of information, processing it in the cloud and seamlessly adjusting the manufacturing parameters via a closed loop system[48]. The same technologies can also be used to monitor the state or condition of the machines in the manufacturing environment [49]. Energy is also one of the main concerns in industrial environments and according to [50] and [51] the utilization of IoT based energy management platforms in the industrial and manufacturing environments can drastically improve the energy consumptions by providing better communication among the industrial entities. These are the reasons that in 2015, manufacturing sector has the top position in terms of the number of installed IoT devices, which is 307 million according to a survey by Gartner [52].

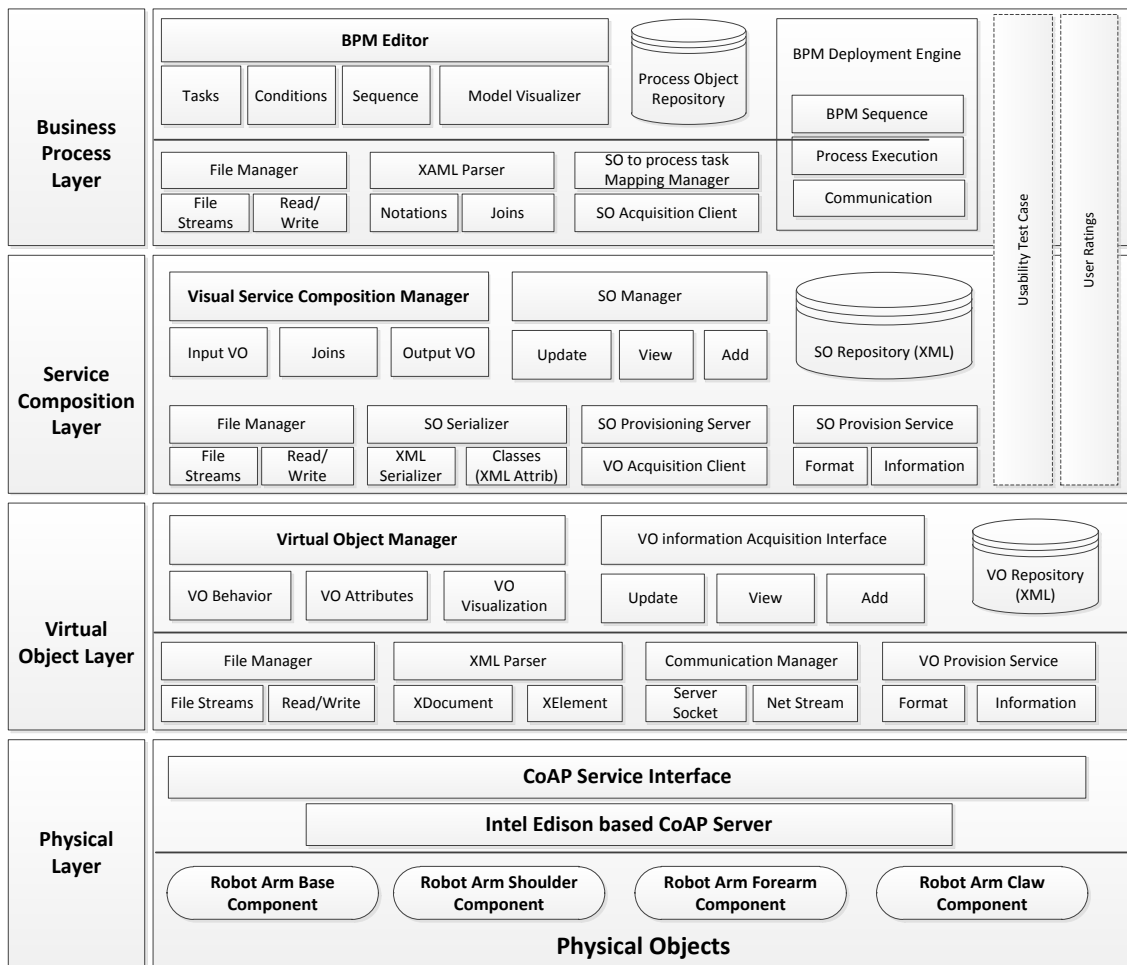
It is therefore evident that such implementation could highly benefit the industrial automation, energy consumption and most of all it will enhance the adaptability of machinery/production entities to changing conditions and requirements. However, the human interaction strategy with such implementations for programming and customization is still under consideration especially from the perspective of common users with lower skill level. This chapter presents a use-case for the usability analysis of the proposed architecture from the perspective of an automated system's customization and control. A prototype robotic arm has been developed for the use-case. Each component of the prototype can be interacted with a remote IoT resource and the system can be utilized by any common user to control and customize the operation of the robotic arm. An experiment has also been described in this chapter to analyze the user feedback regarding the system usability.

## **6.2. Implementation architecture for robotic arm use-case**

The first use-case for the evaluation of the presented system has been chosen from the robotics domain. The field of robotics has recently developed an invigorated interest from the perspective of social and industrial robotics especially from the perspective of IoT enablement. We have selected this use-case because we think that the presented system can be utilized in this domain very effectively by letting the common users with no special programming skills to customize the operations of their personal or industrial robots for innovative and unforeseen usage scenarios.

Figure 32 presents the previously described architecture from the perspective of the use-case by highlighting the prototype robot component implemented as CoAP resources for remote interactions. The architecture also illustrates the addition of new components for the evaluation

and usability study of the system from the perspective of the use-case. The following paragraphs briefly describe the architecture from the perspective of the robot arm control use-case and provide the details of the experimental setup for usability analysis and user ratings.



**Figure 32: Robotic arm use-case implementation in the proposed architecture and modifications for usability study**

For the purpose of the robot prototype demonstration, the robot arm components e.g. the robot base, shoulder, forearm and claw are implemented as CoAP devices. The physical layer of the architecture represents these components as separate physical devices. The details of implementation of these components are presented in a separate section in this document. The

information of the implemented robot components is provided to the Virtual Object Manager which converts the information into virtual objects.

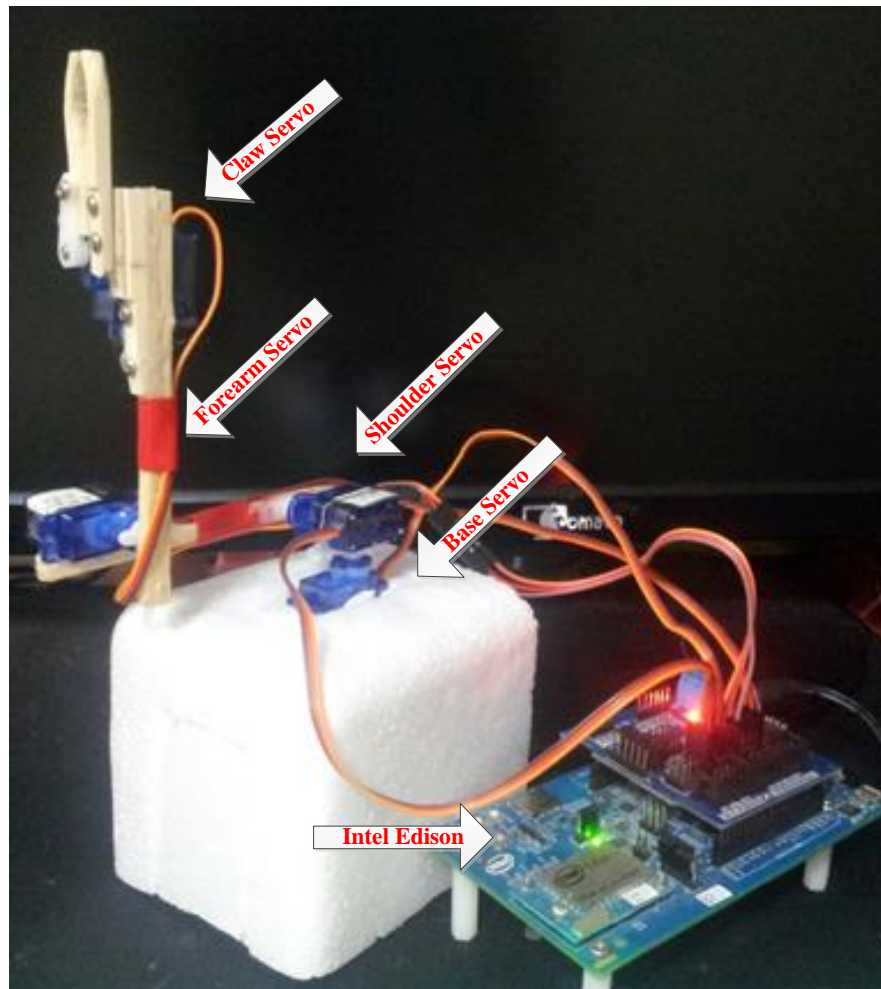
The Service Composition Layer (SCL) utilizes the virtual objects for the robot components from the virtual object repository at the Virtual Object Layer. The virtual objects are visualized using icons which can be interacted with like any Windows based control. The robot VOs are mostly output devices as they perform only actuation tasks. The scenario of operation can be enhanced further with the addition of sensing VOs for a more diverse service composition. In this prototype implementation of the use-case, the user can only compose service object based on the available services as exposed by the robot components. Based on these conditions, the user can create Unit SOs for the robot components by linking them with the generic input module.

The Business Process Layer acquires the service object definitions from the SO repository at SCL, parses the XML representation of the service objects to extract information and then represent the services as BPMN task notations. The user can then utilize these notations to visually create a business process model for the operation of the prototype robotic arm. The user at the Business Process Layer can only control those parts of the robotic arm for which the service objects have been created at the Service Composition Layer. Once the BPM is created, it can be deployed via the BPM deployment engine and the robotic arm will perform the operations accordingly.

### **6.3. Robotic Arm prototype implementation**

This section provides the implementation details for the robotic arm based use-case for the DIY IoT composition architecture. The section will only describe the details of the prototype robotic arm development and the associated components for usability analysis.





Figure 33 presents the finalized form of the prototype robotic arm. The prototype has been developed itself as a DIY project utilizing minimal resources in terms of material as well as cost. The main components of the prototype are as given below.



**Figure 33: Intel Edison based finalized robotic arm prototype**

**Robotic Arm Base component:** Base is the part of the prototype robotic arm which attaches the rest of the arm to a fixed base. The base contains a servo motor which is permanently connected with a stable base on one end and allows the horizontal rotation of the upper parts of the arm from 0 degrees to 180 degrees.

Table 2: Device implementation summary for robotic arm use-case

Component	Base	Shoulder	Forearm	Claw
Visual Representation				
Servo Model	SG-90 miniature servo	SG-90 miniature servo	SG-90 miniature servo	SG-90 miniature servo
Rotation Axis	Horizontal	Vertical	Vertical	Horizontal/ Vertical
CoAP Server	Intel Edison with Californium Framework	Intel Edison with Californium Framework	Intel Edison with Californium Framework	Intel Edison with Californium Framework
CoAP Services	RotateBase0	RotateShoulder0	RotateForeArm0	ClawOpen
	RotateBase90	RotateShoulder90	RotateForeArm90	ClawClose
	RotateBase180	RotateShoulder180	RotateForeArm180	

**Robotic Arm Shoulder component:** The next joint of the prototype robotic arm is termed as the Shoulder. The shoulder is basically a servo motor attached to the base and on the other side is attached an arm. The shoulder servo mimics the shoulder joint and allows the movement of the arm vertically from 0 to 180 degrees.

**Robotic Arm Forearm component:** Forearm is the next joint created by attaching a servo at the free end of the arm attached with the shoulder servo. The purpose of the forearm is to enable the robotic arm to extend via vertical rotation movement and to hold the claw of the robotic arm. This joint also can move from 0 to 180 degrees.

**Robotic Arm Claw component:** The last component of the prototype robotic arm is the Claw. The claw is created by attaching a servo at the free end of the forearm. It consists of a static

finger and a moveable finger. The moveable finger is directly attached to the servo motor of the claw and it mimics the opening and closing of the claw.

Table 2 provides the details related to prototype implementation of the robotic arm for the use-case. The four parts of the robotic arms are equipped with SG-90 miniature servo. SG-90 is a cost effective component for prototype development and it offers enough torque for the operation of our robotic arm. The rotation axis of the components is based on the orientation of the component hence the base rotates horizontally while shoulder and forearm rotates vertically.

Each component is implemented as a CoAP resource as part of CoAP server which is hosted by an Intel Edison board with Java based Californium framework. The Californium framework aids in the development of CoAP server and provides interfaces for CoAP based communication between the server and client. Each component of the robotic arm prototype exposes a number of CoAP services. For example, the base component exposes three CoAP services such as the “RotateBase0”, indicating the Base Rotation to 0 Degrees and “RotateBase180” indicating Base Rotation to 180 Degrees. These service names are used by the Virtual Object Manager along with other information provided by the user to create complete URI for each CoAP resource.

## 6.4. Usability study for Robotic Arm use-case

As the proposed system is focused on providing a DIY development environment for enabling mass involvement in IoT applications development, its usability from the perspective of end-user with different levels of programming skills can ultimately judge its performance. The robotic arm use-case has been developed to test the usability of the proposed architecture in the IoT enabled industrial robotics domain. The study presented in this chapter is a two pronged study. The first part captures the performance of participants from the DIY perspective. This study allows a participant to interact with the system (SCM and BPM Editor) without any prior



training and captures the time in seconds to complete the task. After that the participant is trained briefly, she is allowed to interact with the system in the same manner capturing any difference in the performance (time in seconds to complete the task again). The test scenario for the robotic arm use-case is illustrated in the Figure 34. The XML snippet termed as Virtual Objects in the figure shows the virtual representation of physical IoT devices being sent from VOM to SCM while the second snippet shows the definition of an individual Service Object (part of the BPM created by user for the control of the robotic arm) being sent to CoAP enabled robotic arm for execution. The response is sent back to the Deployment Manager where further execution of the BPM based process is decided.

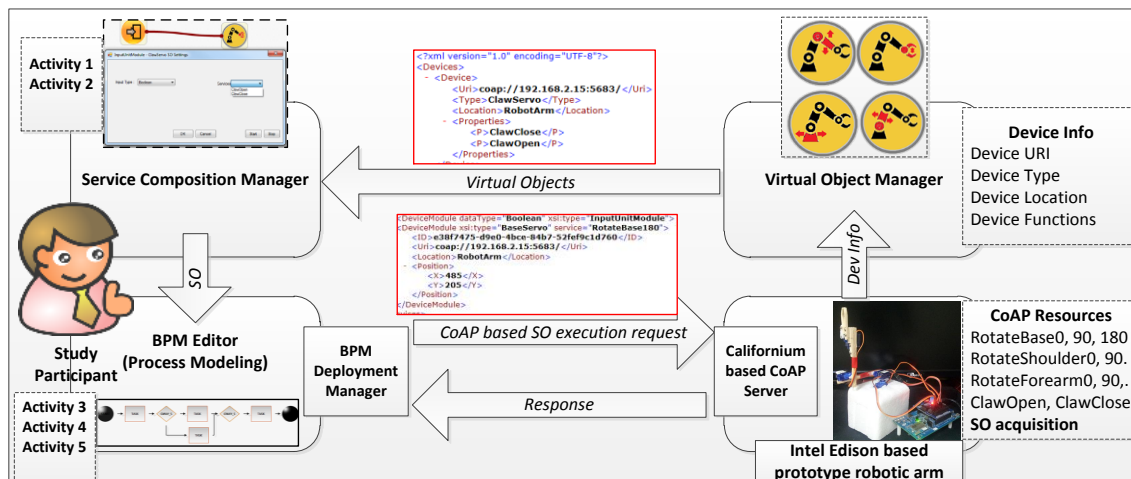


Figure 34: Test scenario for usability assessment of robotic arm use-case

The second part of the study utilizes the System Usability Scale (SUS) [53] for the users to assess the system usability from user's perspective. The SUS questionnaire provides a structured rating system for judging the usability of the system according to the standardized questions. Table 3 provides the list of questions for SUS upon which the participant is asked to rate the system usability. SUS is a 10 item Likert scale through which the users subjectively rate their experience with the system on a scale of 1 (Strongly Disagree) to 5 (Strongly Agree).

**Table 3: System Usability Scale items**

No.	SUS Item
Q1.	I think that I would like to use this system frequently
Q2.	I found the system unnecessarily complex
Q3.	I thought the system was easy to use
Q4.	I think that I would need the support of a technical person to be able to use this system
Q5.	I found the various functions in this system were well integrated
Q6.	I thought there was too much inconsistency in this system
Q7.	I would imagine that most people would learn to use this system very quickly
Q8.	I found the system very cumbersome to use
Q9.	I felt very confident using the system
Q10.	I needed to learn a lot of things before I could get going with this system

$$S = \{\sum(Q_i - 1) + \sum(5 - Q_j)\} * 2.5 \dots\dots\dots Eq. 1$$

Where  $i= 1, 3, 5, 7, 9$  and  $j= 2, 4, 6, 8, 10$ .

Equation 1 is used to calculate the SUS based usability score based on the participants' responses. The resulting scores represent the overall usability of the system (ranging from 0-100). A higher score represents better usability in the given context. According to the pilot studies made by Bangor et al.[54], the usability score above 70 indicates above average usability for the system in question. The same results were verified by Bangor et al. [55] by utilizing more than 900 surveys based on an augmented SUS with adjective ratings scale. In each survey the participants rated a system based on SUS and then provided the adjective ratings for the system's usability. The adjective ratings based usability scale obtained from the study is presented in Table 4 for reference. The Service Composition Layer and Business Process Layer

for this use-case implementation have been slightly changed so that it can be utilized for usability experiments. The flows of the usability experiment at Service Composition Layer and Business Process Layer are given in the following sub-sections.

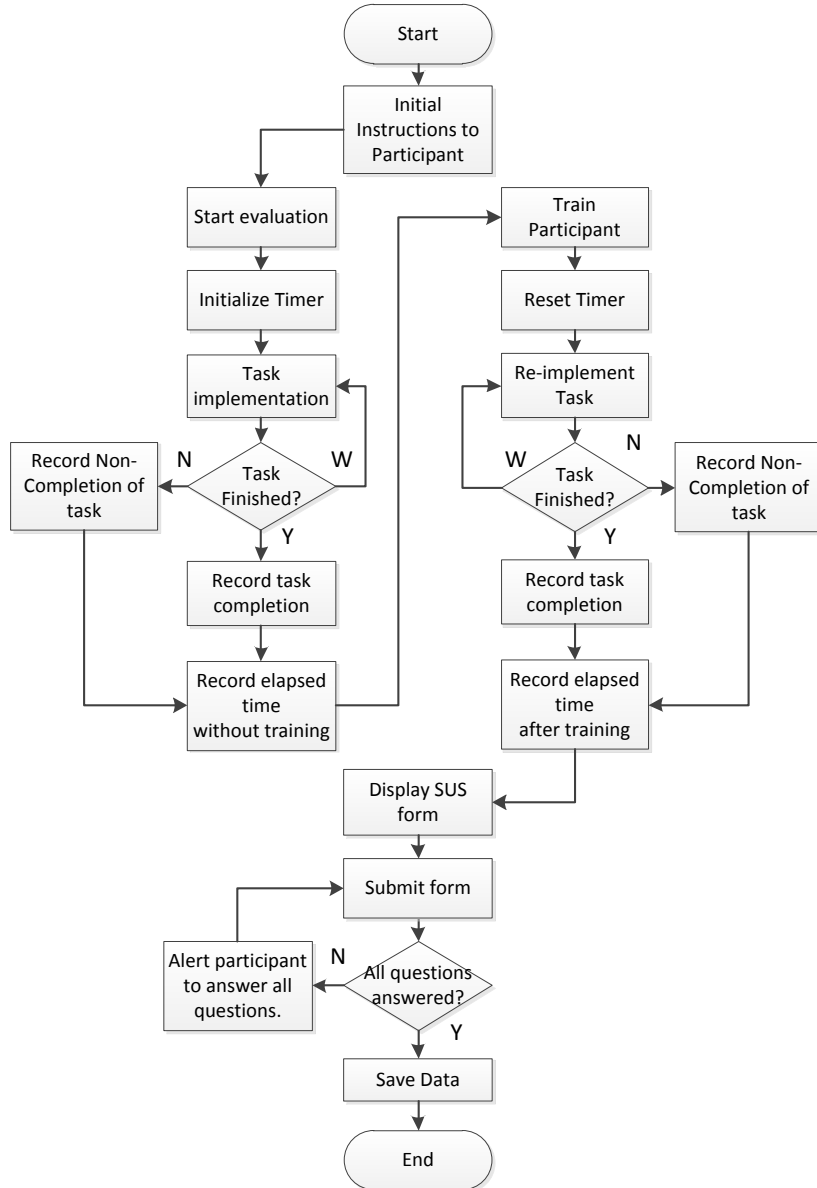
**Table 4: Descriptive Statistics of SUS Scores for Adjective Ratings [55]**

<b>Adjective</b>	<b>Count (Survey)</b>	<b>Mean SUS Score</b>	<b>Standard Deviation</b>
Worst Imaginable	4	12.5	13.1
Awful	22	20.3	11.3
Poor	72	35.7	12.6
Ok	211	50.9	13.8
Good	345	71.4	11.6
Excellent	289	85.5	10.4
Best Imaginable	16	90.9	13.4

### **6.4.1.1. SCM usability assessment for Robotic Arm**

#### **use-case**

Figure 35 illustrates the various steps of the usability analysis experiment designed for the Service Composition Manager (SCM) in the robotic arm use-case. The participant is given a brief description of a service composition task with a visual cue related to the task that the participant is asked to implement using the SCM without any prior training. The participant is allowed to interact with the service composition manager in order to complete the task. A supervisor checks the progress of the participant. The time taken in seconds by the participant in the first step is recorded along with the information if the task was successfully completed or not.



**Figure 35: Flow of SCM usability assessment experiment for robotic arm use-case.**

In the next step of the experiment, the participant is given a brief session of training on how to use the SCM in order to compose a service. Once the training session is completed, the participant is asked to implement the same task again. Once again the elapsed time in seconds and the completion state of the task are recorded by the system. As soon as the interaction session of the participant is completed, the System Usability Scale based ratings form is

presented to the participant by the Service Composition Manager in order to rate the system based on his/her interaction experience. The participant also provides age and gender information but no other personal information is recorded to keep the data anonymous. The participant is asked to rate his/her level of computer programming skills level on a scale of 1 to 5. This information is not part of the SUS ratings but it is used for further usability analysis of the SCM from a DIY perspective.

**Table 5: SCM usability assessment experimental setup based on robotic arm use-case**

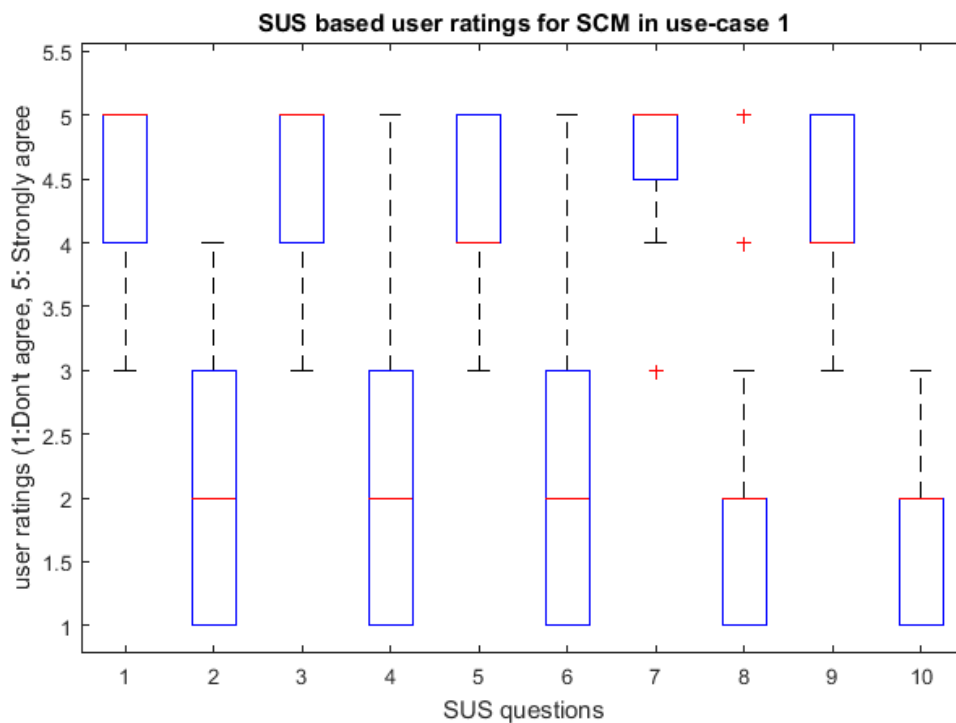
<b>Test use-case</b>		Prototype Robotic Arm	
<b>Number of Participants</b>		20	
		Programmers	Non-programmers
		9	11
<b>Test environment</b>		Service Composition Manager (SCM)	
Activity 1	SCM Task without training	Creation of a Service Object for controlling the robot claw. (open the robot claw)	
	Data recorded	Time to complete the task.	
		Successful/unsuccessful completion of the task	
Brief training session on how to use the SCM			
Activity 2	SCM task after training	Creation of a Service Object for controlling the robot claw. (open the robot claw)	
	Data recorded	Time to complete the task.	
		Successful/unsuccessful completion of the task	
Activity 3	Usability rating	Participant provides personal information and SUS questionnaire based system usability ratings.	
	Data recorded	SUS based participant's responses for 10 questions.	

For the usability analysis of the robotic arm use-case, a total of 20 participants took part in the experimental setup explained below. The usability analysis of the proposed system was done at Jeju National University, so most of the participants of the experiments were international students and native Korean students from various departments along with some participants external to the university. The participants mostly belonged to the age groups between 21 to 35

years with few outliers. The participants included 15 male and 5 female participants. The experimental setup is summarized in Table 5.

### 6.4.1.2. SCM usability score based on SUS

All the participants in the experiment provided SUS based responses to the 10 questions presented in Table 3. Figure 36 presents a question-wise summary boxplot of the responses from all of the participants.



**Figure 36: Results of SCM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Robotic arm use-case)**

The figure shows that for the odd number of SUS questions, the participants mostly agreed by giving the ratings between 4 and 5. The median of responses for Q1, Q3, and Q7 lies at 5 (Strongly agree) while for Q5 and Q9, the median response given by participants lies at 4

showing strong agreement with the statements. For the even numbered questions with negative remarks about the system, the median responses by the participants for Q2, Q4, Q6, Q8 and Q10 lies at 2 (Disagree) respectively. The '+' signs in the boxplot indicate any outlier values in the responses data set recorded by 20 participants based on System Usability Scale.

All the responses from the 20 participants were utilized to calculate the usability score for the Service Composition Manager based on the robotic arm use-case. The average usability score for the SCM is 80 which shows very positive response from all the participants and indicates that the system is highly useable from the perspective of its utilization for service composition task in the industrial or domestic robotics domains.

#### **6.4.1.3. SCM usability from DIY perspective**

The analysis presented in this section is intended to analyze the usability of Service Composition Manager from the perspective of a DIY platform. For this purpose, statistical analysis of the time data recorded during the usability experiment for various task performed by the participants has been conducted. In order for the SCM to be a DIY platform for service composition, it should be equally useable to people with less or no programming skills as it is people with programming skills.

Based on the participant's subjective ratings of their programming skills, two groups of participants were defined. Participants with programming skills above 3 were included in the 'Programmers' (9) group while the rest were grouped as the 'Non-programmers' (11) and assuming that the data samples are following a normal distribution.

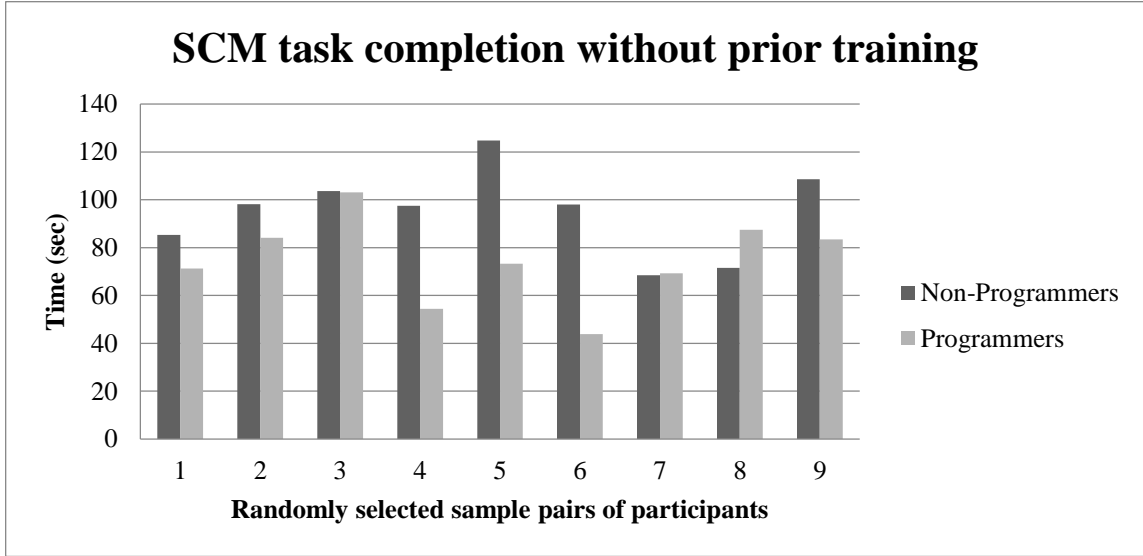


Figure 37: Sample comparison of the time taken by both groups to complete SCM task without any prior training

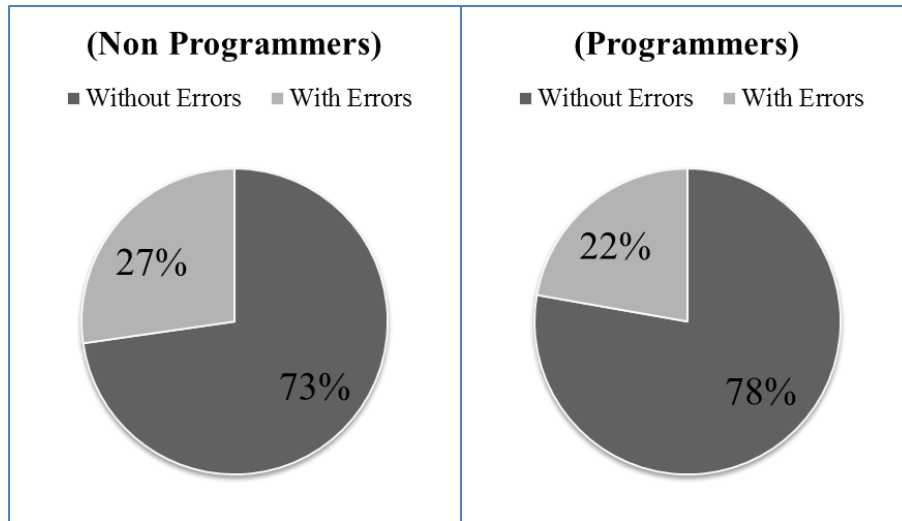


Figure 38: Comparison of percentage for successful task completion by Programmers vs Non-Programmers

Two-Sample t-Test analysis was performed for the two groups based on the time taken by each participant to complete the first service composition task without any training given to the participants. Figure 37 provides a comparison graph of the sample data recorded for both groups.



The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which clearly shows a numerical difference in the mean time taken by both groups to complete the activity 1 at SCM. As there was no prior training, participants from both groups made errors while completing the activity 1. Figure 38 provides a comparison of the percentage of each group completing the task with or without errors. The figure shows that members of both groups were affected almost the same due to the lack of knowledge about how to use the SCM interface.

Table 6 shows the outcome of the t-Test analysis based on the null hypothesis (H0) that the mean time taken by programmers to complete the task without training is not significantly different than the mean time taken by the non-programmers to complete the same task without any prior training. This is as represented by the Hypothesized Mean Difference (0) in the table.

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

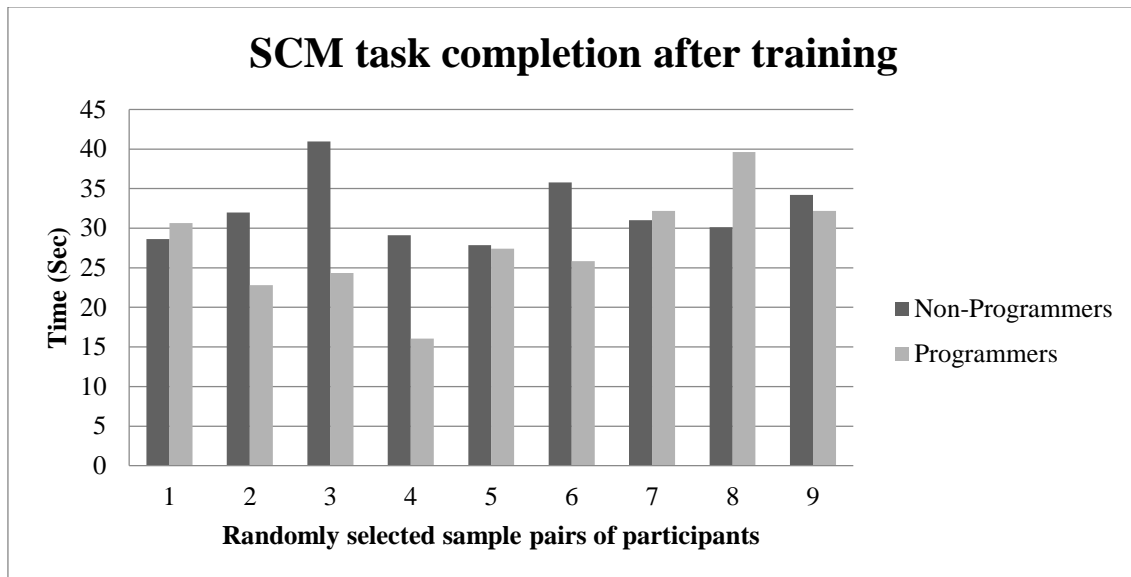
**Table 6: Two-Sample t-Test for unequal variances (Activity 1 before training)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	93.40436364	74.46166667
Variance	317.1363765	317.474346
Observations	11	9
Hypothesized Mean Difference	0	
df	17	
t Stat	2.365887786	
P(T<=t) two-tail	0.03013123	
t Critical two-tail	2.109815578	

As the data samples are not paired, a t-Test with the assumption of unequal variances for both data samples was conducted. For the null hypothesis to be rejected, t Stat should be greater

than t Critical two-tail. This is true for the case as described in Table 6,  $2.36 > 2.11$ , hence the null hypothesis is rejected which means that there was a significant difference in the time taken by programmers to complete the service composition task without any prior training ( $M=74.46$ ) and time taken by non-programmers to complete the task without any prior training ( $M=93.40$ );  $t(17)=2.11, p = 0.030$ .

The difference shown by the t-test presents very slight difference between the two samples and the difference can be attributed to the fact that programmers had the additional edge of computer usage for programming purposes and an understanding of the drag-n-drop type interfaces previously while the non-programmers used slightly more time to understand and explored the interface as there was no prior training given to them.



**Figure 39: Sample comparison of the time taken by both groups to complete SCM task after the training session**

To prove the fact, a second Two-Sample t-Test was performed for the two groups based on the time taken by each participant to complete the same service composition task after a brief session of training given to the participants. A sample of data used for this analysis is given in

Figure 39. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which clearly shows a numerical difference in the mean time taken by both groups to complete the activity 2 at SCM.

Table 7 shows the outcome of the t-Test based on the null hypothesis (H0) that the mean time taken by programmers to complete the task after the training is not significantly different than the mean time taken by the non-programmers to complete the same task after the training session as represented by the Hypothesized Mean Difference (0) in the table. Due to the difference of variance for both data samples (21.7 and 45.6) t-Test assuming unequal variances was conducted.

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

**Table 7: Two-Sample t-Test for unequal variances (Activity 1 after training)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	33.18409	27.89778
Variance	21.70197	45.62954
Observations	11	9
Hypothesized Mean Difference	0	
df	14	
t Stat	1.99195	
P(T<=t) two-tail	0.066252	
t Critical two-tail	2.144787	

As shown in Table 7, t Stat is not greater than t Critical two-tail ( $1.99 < 2.145$ ), hence the null hypothesis is accepted. The test specifies that there is no statistically significant difference in the time taken by programmers to complete the service composition task after the training

session (M=27.88) and time taken by non-programmers to complete the task after the training session (M=33.18);  $t(14)=1.99$ ,  $p = 0.066$ . The above analysis shows that the proposed service composition manager interface is suitable for the DIY usage and is equally helpful to people with programming skills and people with less or no programming skills.

The above analysis indicates that the proposed Service Composition Manager interface is suitable for the DIY usage after a minimal training which can be provided in the form of a manual or video tutorial. It is also deduced that the interface is equally helpful to people with programming skills and people with less or no programming skills. It is important to note that both the groups achieved 100 % task completion without any errors once they were trained on how to use the SCM.

## **6.4.2. BPM Editor usability assessment for Robotic Arm use-case**

Figure 40 illustrates the various steps of the usability analysis experiment designed for the Business Process Modeling Editor (BPM Editor) at the Business Process Layer of the proposed system. The participant is given a brief description of a simple process based on the previously composed services with a visual cue related to the final model of the process that the participant is asked to implement using the BPM Editor.

The participant is allowed to interact with the editor in order to model the process by simple drag-n-drop and mouse click operation on the graphical BPM notations. A supervisor checks the progress of the participant. The time taken in seconds by the participant in the first step is recorded by the application along with the information if the task was successfully completed or not. In the next step of the experiment, the participant is given a brief session of training on how to use the BPM Editor in order to visually create a business process model. Once the training

session is completed, the participant is asked to create the same business process model again using the BPM editor. Again the elapsed time in seconds and the completion of the task are recorded by the system.

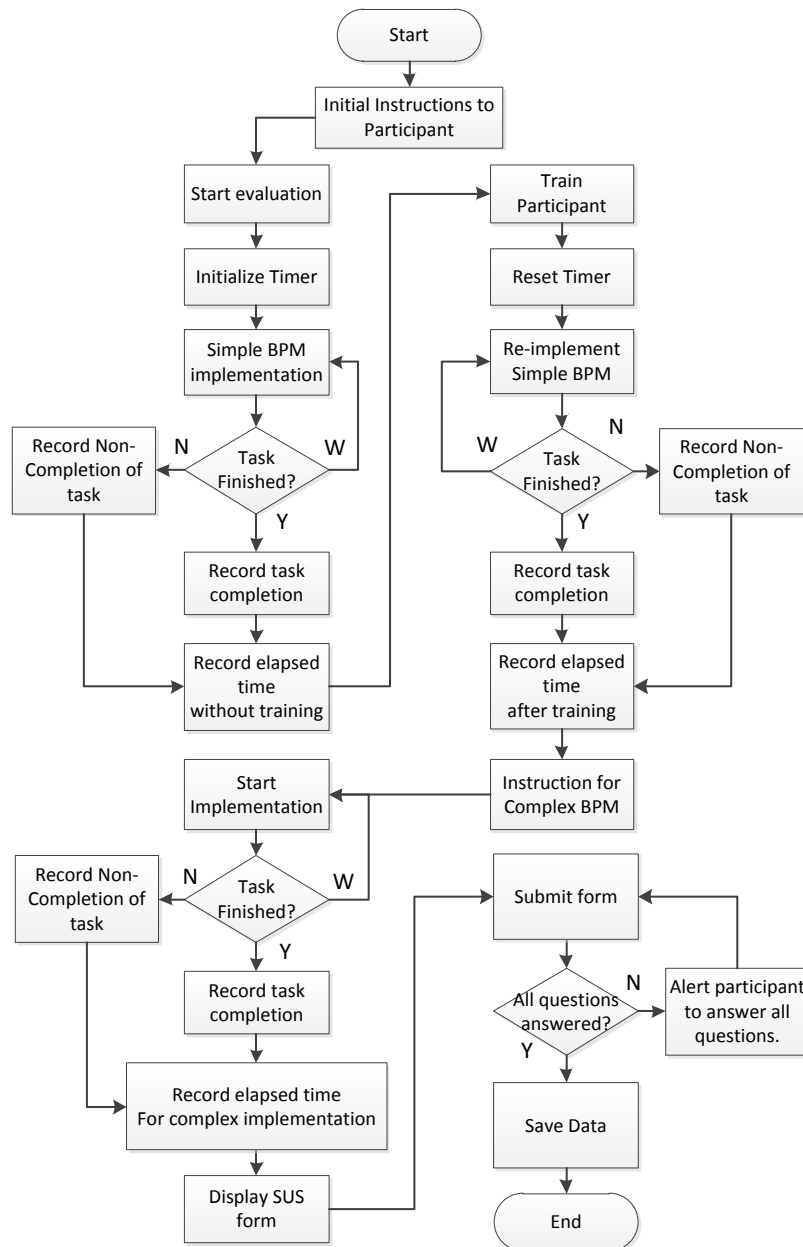


Figure 40: Flow of BPM Editor usability assessment experiment for robotic arm use-case

As Business Process Models can become complex for larger processes, an extra step has been added to this experiment to let the participant assess the system based on a complex process modeling task. In this step, the participant is given the description of a more complex process model to be visually created using the BPM editor. The participant tries to visually create the corresponding process model using the graphical notations. Again the time taken by the participant to complete the task and the state of errors made by the participant is recorded by the system.

**Table 8: BPM Editor usability assessment experimental setup based on robotic arm use-case**

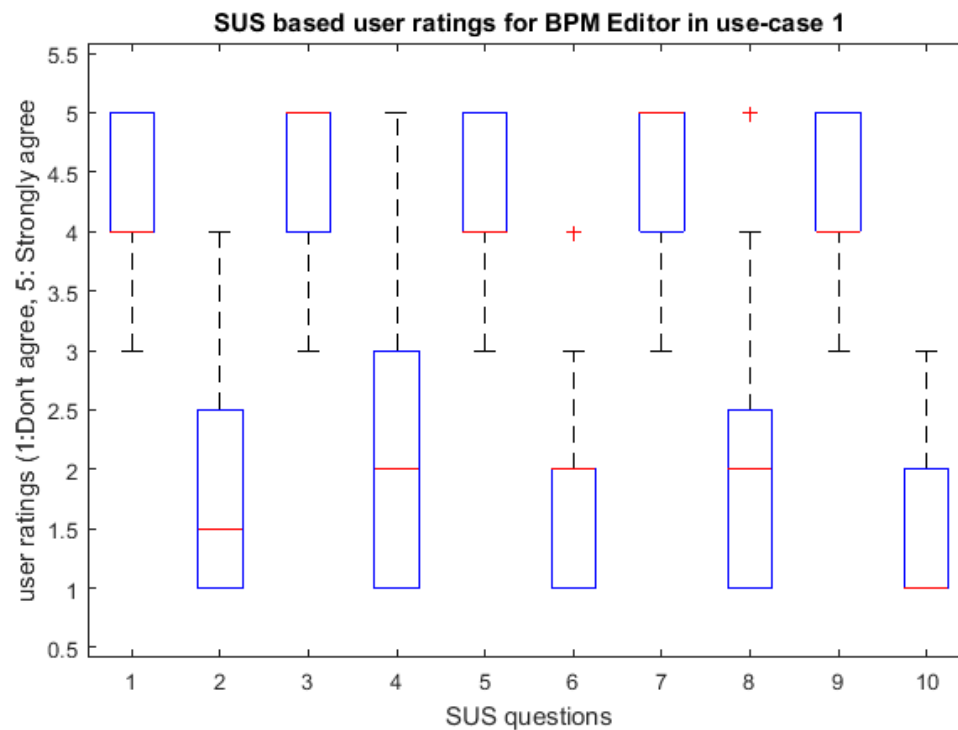
<b>Test use-case</b>		Prototype Robotic Arm	
<b>Number of Participants</b>		20	
		Programmers	Non-programmers
		9	11
<b>Test environment</b>		Business Process Modeling Editor (BPM Editor)	
Activity 1	Simple BPM task without training	Three step BPM creation task to for the control of prototype robot arm motion.	
	Data recorded	Time to complete the task.	
		Successful/unsuccessful completion of the task	
Brief training session on how to use BPM Editor			
Activity 2	Simple BPM task after training	Three step BPM creation task to for the control of prototype robot arm motion.	
	Data recorded	Time to complete the task.	
		Successful/unsuccessful completion of the task	
Activity 3	Complex BPM task	6 Step BPM task utilizing gateway notations and script notations to create a complex process model for robot arm control.	
	Data recorded	Time to complete the task.	
		Successful/unsuccessful completion of the task	
Activity 4	Usability rating	Participant provides personal information and SUS questionnaire based system usability ratings.	
	Data recorded	SUS based participant's responses for 10 questions.	

Once the interaction session of the participant is completed, a System Usability Scale based assessment form is presented to the participant by the system in order to rate the system based on his/her interaction experience. The participant also provides age and gender information but no

other personal information is recorded to keep the data anonymous. The participant is asked to rate his/her level of computer programming skills level on a scale of 1 to 5. This information is not part of the SUS questionnaire but used for further usability analysis of the system from a DIY perspective. The experimental setup is summarized in Table 8.

### 6.4.2.1. BPM Editor usability score based on SUS

All the participants in the experiment provided SUS based responses to the 10 questions presented in Table 3. Figure 41 presents a question-wise summary boxplot of the responses from all of the participants.



**Figure 41: Results of BPM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Robotic arm use-case)**

The figure shows that for the odd number of SUS questions, the participants mostly agreed by giving the ratings between 4 and 5. The median of responses for Q3 and Q7 lies at 5 (Strongly agree) while for Q1, Q5 and Q9, the median response given by participants lies at 4 showing strong agreement with the statements. For the even numbered questions with negative remarks about the system, the median responses by the participants for Q4, Q6 and Q8 lies at 2 (Disagree) while for Q2 and Q10 the median response by all the participants is 1.5 and 1 respectively. This shows a strong disagreement of the study participants with the negative remarks regarding the usability of the BPM Editor. The '+' signs in the boxplot indicate any outlier values in the responses data set recorded by 20 participants based on System Usability Scale.

All the responses from the 20 participants were utilized to calculate the usability score for the Business Process Modeling Editor based on the robotic arm use-case. The average usability score for the BPM Editor is 81.62 which shows very positive response from all the participants and indicates that the BPM based IoT application development for the industrial or domestic robotics domain is highly useable based on user ratings.

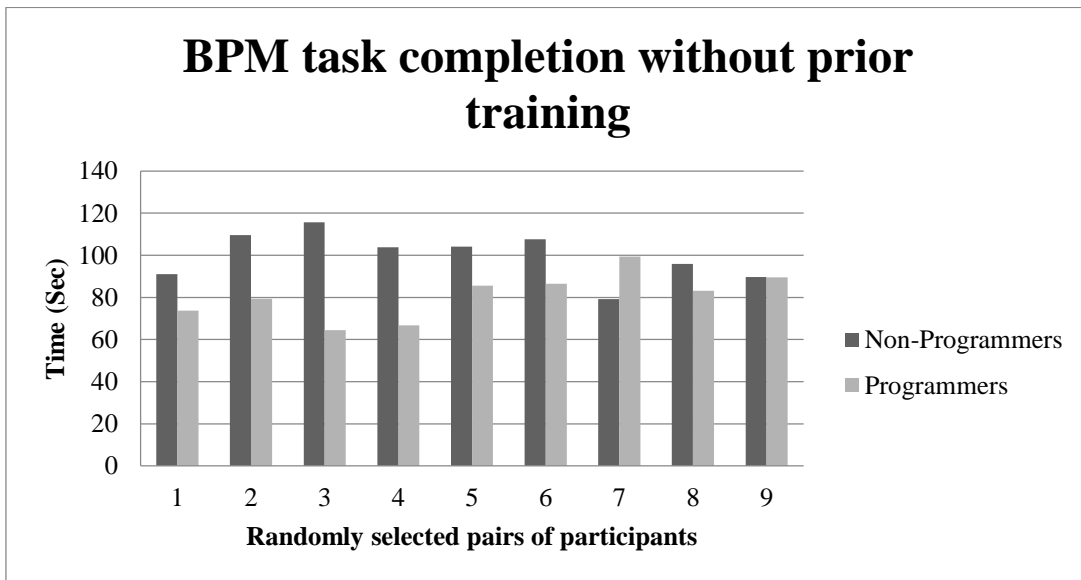
#### **6.4.2.2. BPM Editor usability from DIY perspective**

The analysis presented in this section is intended to analyze the usability of the Business Process Modeling Editor from the perspective of a DIY platform. As in the previous section, statistical analysis of the time data recorded during the usability experiment for various task performed by the participants has been conducted. In order for the BPM Editor to be a DIY platform for IoT application process modeling, it should be equally useable to people with less or no programming skills as it can be to people with programming skills.

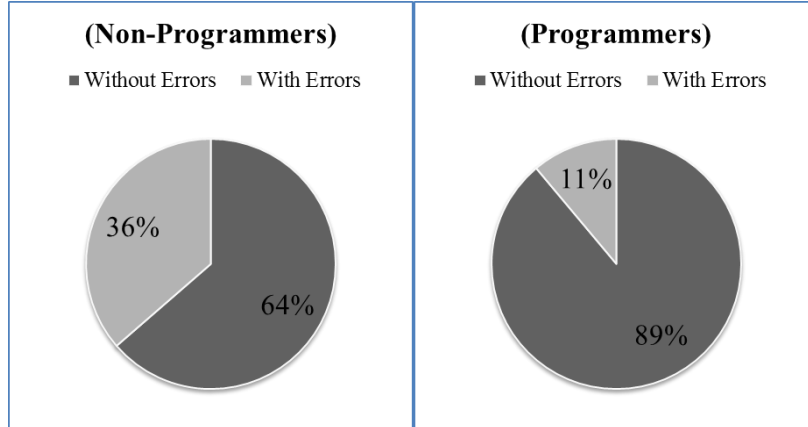


This study also utilizes the same groups of ‘Programmers’ (9) and ‘Non-programmers’ (11) participants and assuming that the data samples follow a normal distribution. The groups have been created based on the participants’ subjective ratings of their own programming skills.

Two-Sample t-Test analysis was performed for the two groups based on the time taken by each participant to complete the first BPM task without any training given to the participants. Figure 42 provides a comparison graph of the sample data recorded for both groups. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 1 using the BPM Editor.



**Figure 42: Sample comparison of the time taken by both groups to complete BPM task without any prior training (Robotic arm use-case)**



**Figure 43: Comparison of percentage successful completion of simple BPM task by Programmers vs Non-Programmers without any prior training (Robotic arm use-case)**

As there was no prior training, participants from both groups made errors while completing the activity 1. Figure 43 provides a comparison of the percentage of each group completing the task with or without errors.

For this part of the usability analysis experiment for the robotic arm use-case, the participants were asked to create a simple BPM for controlling the prototype robotic arm without any prior training on how to use the proposed BPM Editor. Two-Sample t-Test analysis was performed for the two groups (Non-programmers, Programmers) based on the time taken by each group member to complete the simple BPM task without any training given to the participants.

Table 9 shows the outcome of the t-Test analysis based on the null hypothesis (H0) that the mean time taken by programmers to complete the task without training is not significantly different than the mean time taken by the non-programmers to complete the same task without any prior training. This is represented by the Hypothesized Mean Difference (0) in the table. A t-Test assuming unequal variances was conducted due to the difference in the variance for both samples of data.

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

**Table 9: Two-Sample t-Test for unequal variances (Simple BPM task without prior training, robotic arm use-case)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	97.06555	80.908
Variance	170.6756	125.5395
Observations	11	9
Hypothesized Mean Difference	0	
df	18	
t Stat	2.976622	
P(T<=t) two-tail	0.008086	
t Critical two-tail	2.100922	

Based on the outcome of the t-Test analysis, the null hypothesis is rejected which means that there is a significant difference in the time taken by programmers to complete the BPM task without any prior training (M=80.91) and time taken by non-programmers to complete the task without any prior training (M=97.06);  $t(18)=2.98$ ,  $p = 0.008$ .

The difference shown by the t-test presents a slight difference between the two samples and the difference can be attributed to the fact that programmers had the additional edge of computer usage for programming purposes and an understanding of the drag-n-drop type interfaces previously while the non-programmers used slightly more time to understand and explore the interface as there was no prior training given to them.

To prove the fact, a second Two-Sample t-Test was performed for the two groups based on the time taken by each group member to complete the same BPM task after a brief session of training given to the participants. Figure 44 presents the comparison graph of sample data recorded for both the groups. The graph shows the comparison of time in seconds, taken by

randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 2 using the BPM Editor.

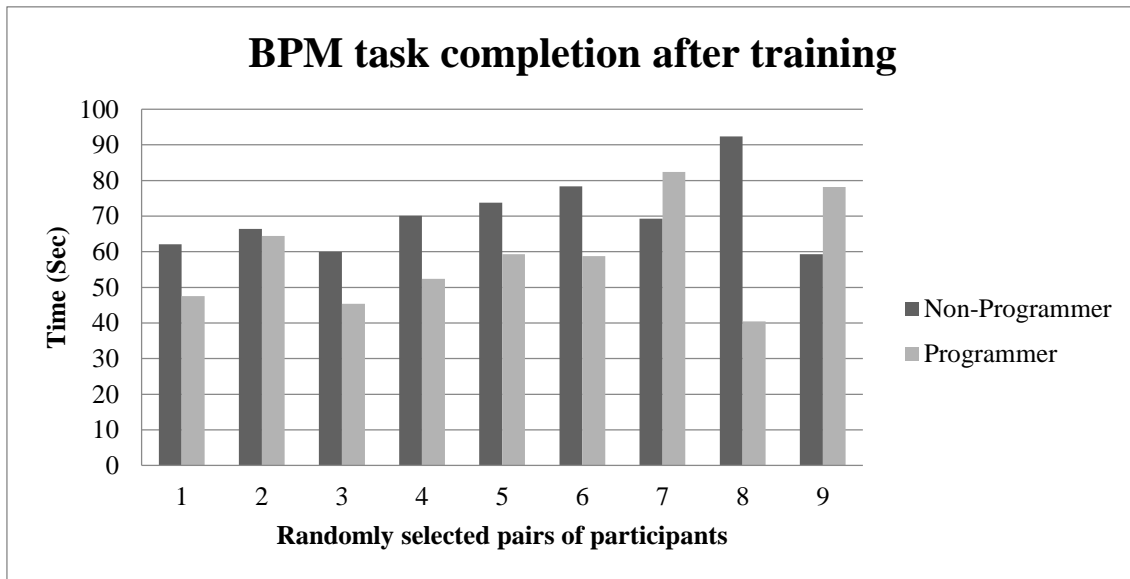


Figure 44: Sample comparison of the time taken by both groups to complete BPM task after training (Robotic arm use-case)

Table 10 shows the outcome of the t-Test based on the null hypothesis ( $H_0$ ) that the mean time taken by programmers to complete the simple BPM task after the training is not significantly different than the mean time taken by the non-programmers to complete the same task after the training session. This is represented by the Hypothesized Mean Difference (0) in the table. As the data samples have different variance, t-Test assuming unequal variances was conducted.

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

Based on the outcome of the t-Test analysis, the null hypothesis is accepted as t Stat is less than t Critical ( $1.58 < 2.11$ ). The result shows that there is no significant difference in the time taken by programmers to complete the Simple BPM task after training on how to use the BPM Editor ( $M=58.74$ ) and time taken by non-programmers to complete the task after training ( $M=68.60$ );  $t(17)=1.58$ ,  $p = 0.132$ .

**Table 10: Two-Sample t-Test for unequal variances (Simple BPM task after training)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	68.60327	58.74956
Variance	175.8813	206.4545
Observations	11	9
Hypothesized Mean Difference	0	
df	17	
t Stat	1.579304	
P(T<=t) two-tail	0.132692	
t Critical two-tail	2.109816	

To further prove the impact of DIY nature of the BPM Editor, both groups of participants were asked to complete a more complex BPM task. Figure 45 shows a comparison graph of the sample data collected from both groups. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group.

A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 3 using the BPM Editor. The time for programmers and non-programmers to complete the task has been analyzed using a Two-Sample t-Test. The null hypothesis ( $H_0$ ) states that the mean time taken by programmers to complete the complex BPM task is not significantly different than the mean time taken by the

non-programmers to complete the same task. Table 11 presents the outcome of the t-Test. The null hypothesis is stated as the Hypothesized Mean Difference (0) in the table.

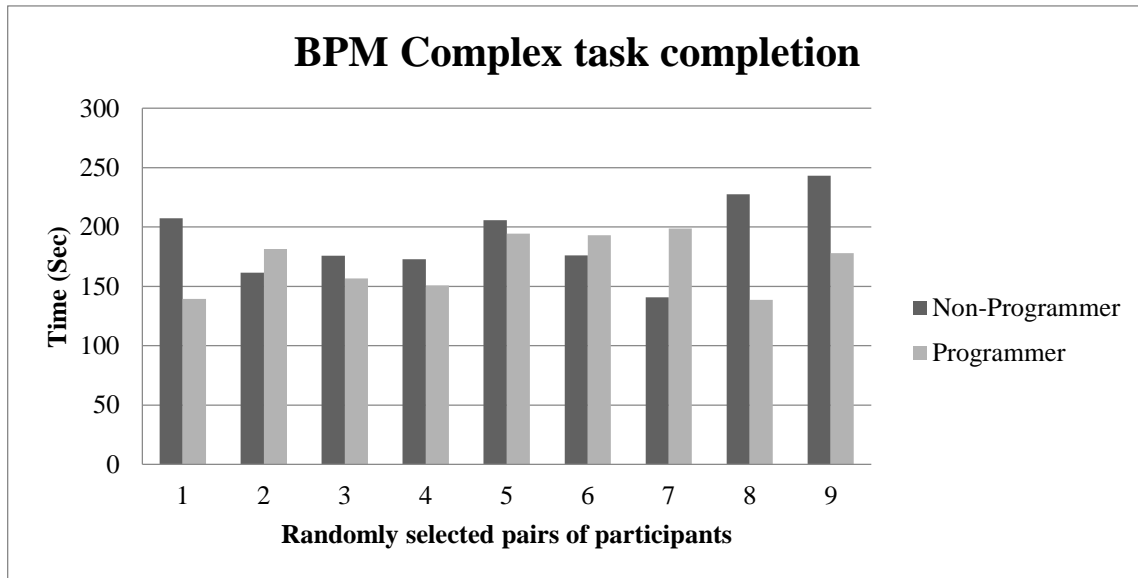
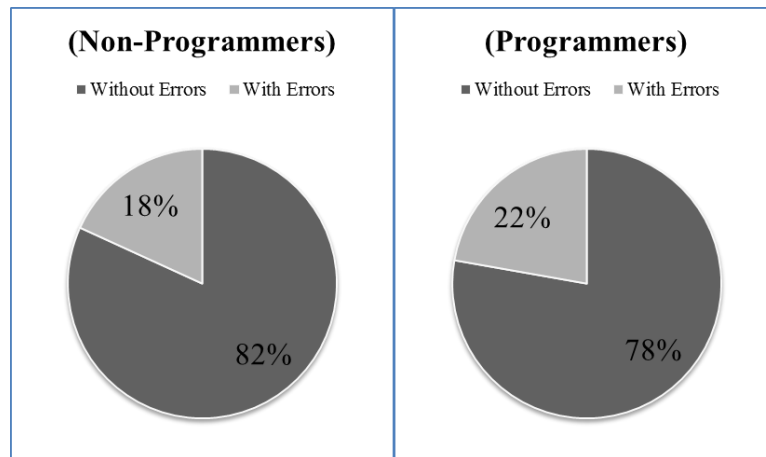


Figure 45: Sample comparison of the time taken by both groups to complete BPM complex task (Robotic arm use-case)

Table 11: Two-Sample t-Test for unequal variances (Complex BPM task)

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	183.5797	170.1048
Variance	1125.859	577.5301
Observations	11	9
Hypothesized Mean Difference	0	
df	18	
t Stat	1.044222	
P(T<=t) two-tail	0.310205	
t Critical two-tail	2.100922	



**Figure 46: Comparison of percentage successful completion of BPM complex task by Programmers vs Non-Programmers (after training)**

According to the outcome of the t-Test analysis, the null hypothesis is accepted. It means that there is no statistically significant difference found in the time taken by programmers to complete the complex BPM task ( $M=170.10$ ) and the time taken by non-programmers to complete the same task ( $M=183.58$ );  $t(18)=1.04$ ,  $p = 0.310$ . Hence, it can be deduced from the analysis that the BPM Editor can provide significant help to non-programmers to perform as efficiently as the programmers and provides a better DIY environment for programming IoT related applications and processes in the industrial or domestic robotics domain. The task completion percentage comparison as shown in Figure 46 also supports the claim that the behavior of the BPM Editor is almost same for both programmers and non-programmers.

# 7. Usability Study for Smart-Space use-case

## 7.1. IoT enabled Smart-Spaces

The recent development in technology in the form of Internet of Things (IoT) has on one hand revitalized innovations in industry and on the other hand it has generated a new interest towards smart spaces research and development [56]. The concept of smart spaces has been associated with a number of physical spaces such as home, office, malls, schools and smart cities. These physical spaces are utilized by people in different ways and based on the complexities involved, the people's experience can degrade [57]. Technologies such as Internet of Things have been utilized in these spaces to improve user experience in terms of services.

The smart space services span over several domains and the domain getting the most attention is remote monitoring of occupants' activities in a smart space. The purposes of activity monitoring include health care purposes as presented in [58][59] [60] and [61]. Occupants' activities in a smart home are also monitored for the purpose of energy consumption optimization as presented by Lima et al. [62].

Smart home monitoring and appliance control is another topic vastly studied in this regard. The aim of smart home monitoring and appliance control services is to provide a remote interface which enables the user to monitor and allow smart appliances to be controlled with minimal interactions from the users [63] [64] [65]. Energy management is another type of smart services deployed in smart spaces. These services utilizes IoT and other recent technologies to monitor energy load and provide a smart consumption scheme for smart homes [66][67] and office spaces [68] [69]. The list of these services and domains of application in smart spaces it so



vast that large scale implementations of smart spaces are now being termed as smart cities [70] [71].

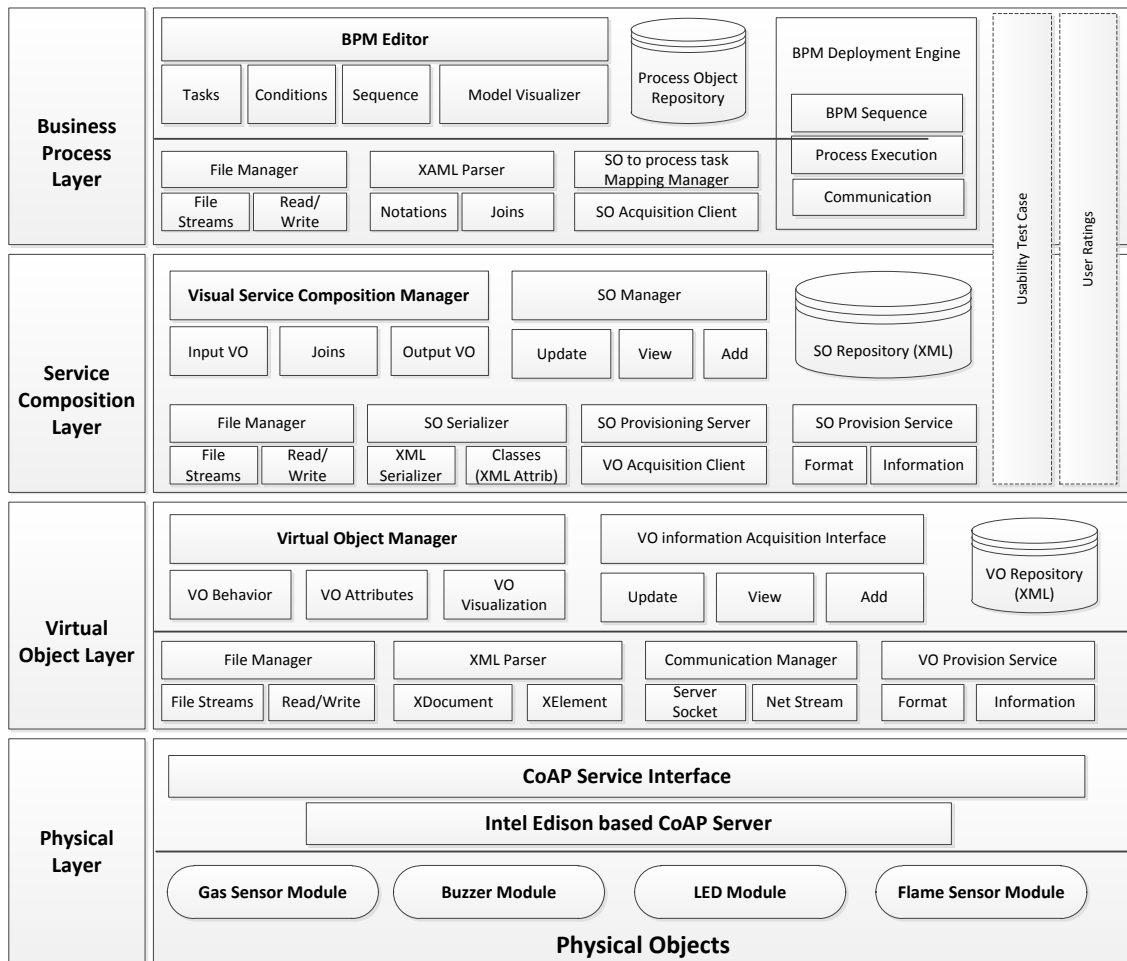
Although there are numerous effort for research and development as specified by the few examples in the previous paragraph, there are still several questions to be answered yet. One of the research challenges as specified by Helal [57] is, how intuitive and effective interaction between users and smart spaces can be implemented?. The question becomes more valid when viewed from the perspective of end-user customization of these IoT implementations in smart spaces based on the user preferences. Intuitiveness in this regard is important as the end-user is not equipped with programming skills and hence alternative techniques must be proposed and tested. Literature review indicates that not much research has been done in this regard.

It is due to these circumstances that our second use-case for the usability analysis of the proposed system has been chosen from smart space perspective. This chapter presents the use-case for the usability analysis of the proposed architecture from the perspective of end-user customization and control of a smart space. A prototype smart space based on sensing and actuation devices has been developed for the use-case. Each device in the prototype smart space has been implemented as a remote IoT resource and the proposed architecture can be utilized by any common user to control and customize the interaction with the smart space. An experiment has also been described in this chapter to analyze user feedback regarding the system usability.

## **7.2. Implementation architecture for smart-space use-case**

Figure 47 presents the previously described architecture from the perspective of the use-case by highlighting the prototype smart space components implemented as CoAP resources for

remote programming and customization according to user preferences. The architecture also illustrates the addition of new components for the evaluation and usability study of the system from the perspective of the use-case. The following paragraphs briefly describe the architecture from the perspective of the smart space programming and customization use-case and provide the details of the experimental setup for usability analysis and user ratings.



**Figure 47: Smart space use-case implementation in the proposed architecture and modifications for usability study**

For the purpose of the prototype smart space demonstration, the various sensing and actuating components e.g. the temperature sensor, gas sensor, buzzer, LED etc. are implemented as CoAP enabled IoT devices. The Physical Layer of the architecture represents these

components as separate physical devices. The detail of implementation for these components/devices is presented in a separate section in this document. The information of the implemented smart space components is provided to the Virtual Object Manager by the user and VOM converts the information into virtual objects.

The Service Composition Layer (SCL) utilizes the virtual objects for the smart space components from the virtual object repository at the Virtual Object Layer. The Virtual Objects are visualized using icons which can be interacted-with like any Windows based control. The prototype component VOs include sensors as well as actuators for portraying a smart space implementation, hence the users can create sensing as well as actuation service objects to experience the system a more diverse manner. Service composition based on these components enable the users to define their own sets of unit service objects which can be utilized to define the interaction flow for the prototype smart space. Unit SOs for the smart space components are created by linking the respective virtual objects with the generic input or output modules provided by the SCM interface.

The Business Process Layer acquires the service object definitions from the SO repository at SCL, parses the XML representation of the service objects to extract information and then represent the services as BPMN task notations. The user can then utilize these notations to visually create a business process model for the operation/behavior of the prototype smart space. The user at the Business Process Layer can only utilize those components of the smart space for which the service objects have been created at the Service Composition Layer. Once the BPM is created, it can be deployed via the BPM deployment engine and the smart space prototype will perform/ behave in accordance with the user defined model.

### 7.3. Smart-space prototype implementation

This section provides the details of prototype implementation of smart space based use-case for the DIY IoT composition architecture. The section will only describe the details of the prototype smart space development and the associated components for usability analysis.

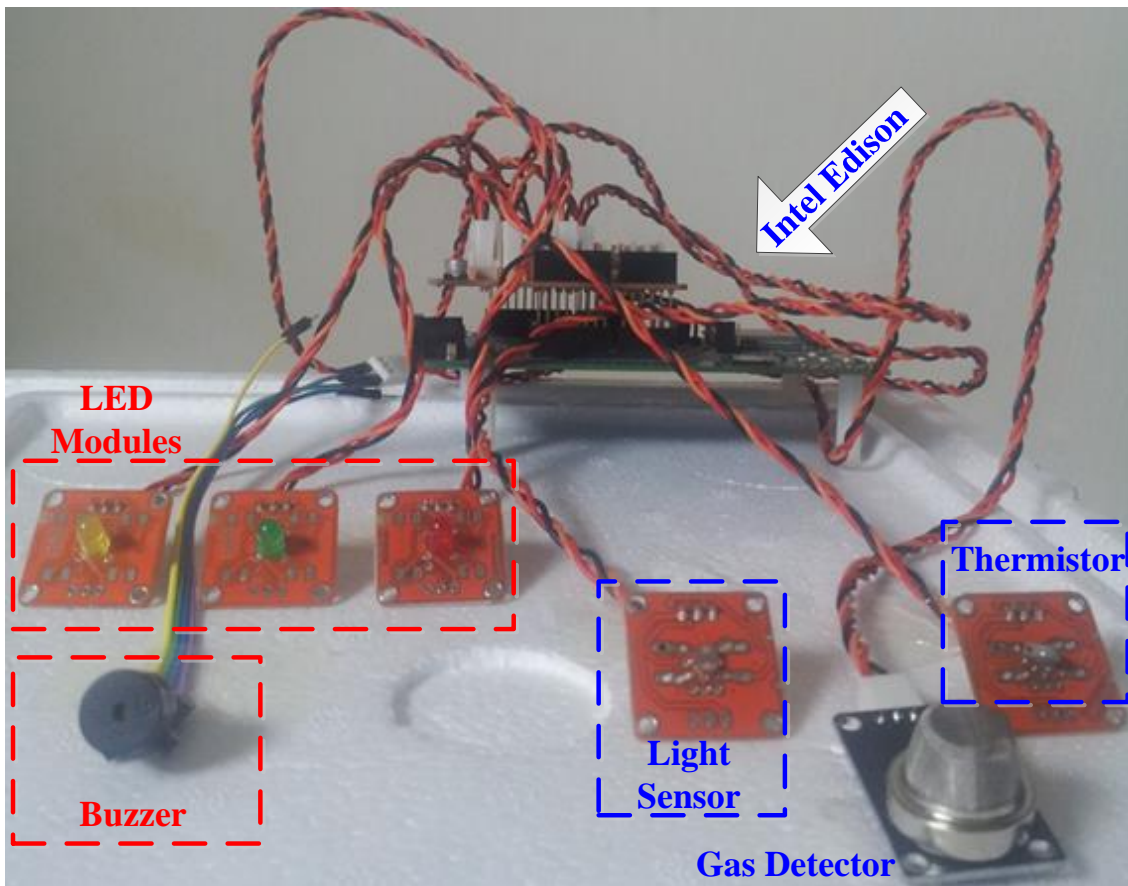





Figure 48: Intel Edison based finalized Smart-Space prototype

Figure 48 presents the finalized form of the prototype smart space. The prototype has been developed as a miniature representation of a smart space scenario where multiple sensing devices are deployed to capture the contextual data and actuating devices are deployed to modify the surroundings or interact with the people in the smart space. The prototype consists of the

following sensing and actuating resources which will be used by users to define/customize the behavior of the smart space based on the contextual situations. The following components have been utilized in the smart space prototype implementation for context acquisition and are termed as the input devices. Table 12 provides summary of the input devices used for this use case.

**Gas sensor module:** Gas sensor (MQ2) is a common and useful gas leakage detection module. The module has been developed for Arduino related implementations and can be utilized for home and industry related applications. It can detect Hydrogen gas, Liquefied Petroleum Gas, Methane gas, Carbon Monoxide and smoke or Propane. It has been implemented as a CoAP resource on the Edison platform for this use case.

**Table 12: Input device implementation summary for smart space use-case**

Input Devices	Temperature Sensor	Gas Sensor	Light Sensor
Visual Representation			
Servo Model	TinkerKit T000200 thermistor module	MQ2 gas sensor module	TinkerKit T000090 LDR module
CoAP Server	Intel Edison with Californium Framework	Intel Edison with Californium Framework	Intel Edison with Californium Framework
CoAP Services	GetTempC	GetGasReading	GetLightReading
	GetTempF		


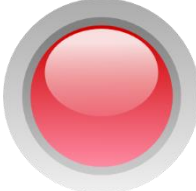
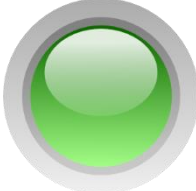
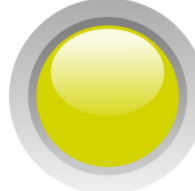
**Temperature sensor module:** TinkerKit T000200 thermistor is specifically designed to be used with the TinkerKit development toolkit for Arduino. The thermistor outputs a voltage between 0 and 5 volts as the temperature value increases. The module has been utilized to implement

Edison based CoAP services for providing temperature values in Centigrade as well as Fahrenheit scales.

**Light sensor module:** The TinkerKit T000090 Light Dependent Resistor (LDR) is another module which is part of the TinkerKit development toolkit for Arduino. The LDR outputs 5 volts when it receives no light and 0 volts when it is exposed to bright light. The module has been utilized to implement Edison based CoAP service for providing illumination value at a given instant.

Table 13 provides the details related to the output devices used in the prototype implementation of the smart space for the use-case. The four output devices include a buzzer and three LED modules with different colors. Given below is a brief description of the buzzer and LED modules.

**Table 13: Output device implementation summary for smart space use-case**

<b>Output Devices</b>	<b>Buzzer</b>	<b>Red LED</b>	<b>Green LED</b>	<b>Yellow LED</b>
<b>Visual Representation</b>				
<b>Device Model</b>	KY-012 Active Buzzer	TinkerKit T01011x LED Module	TinkerKit T01011x LED Module	TinkerKit T01011x LED Module
<b>CoAP Server</b>	Intel Edison with Californium Framework	Intel Edison with Californium Framework	Intel Edison with Californium Framework	Intel Edison with Californium Framework
<b>CoAP Services</b>	BuzzerOn	RedLED_On	GreenLED_On	YellowLED_On
	BuzzerOff	RedLED_Off	GreenLED_Off	YellowLED_Off
		RedLED_Blink	GreenLED_Blink	YellowLED_Blink

**Buzzer module:** Active buzzers are used to produce sounds in electronics. The KY-012 active buzzer has been utilized for producing some sort of alarming sound in the smart space prototype implementation. The buzzer module is implemented as a CoAP resource on the Edison platform and it exposes two CoAP services for turning the buzzer on and turning it off.

**LED module:** Light Emitting Diodes are the simplest types of output devices from the perspective of electronic prototyping. We have used three LED modules from TinkerKit toolbox for Arduino development. Each of these LEDs has been implemented as a CoAP resource which exposes three CoAP services for turning the LED on, turning it off and making the LED blink for a period of time.

The IoT resources have been implemented using Java programming and Californium framework for CoAP implementations. The Californium framework aids in the development of CoAP server and provides interfaces for CoAP based communication between the server and client. The service names for each input and output device are used by the Virtual Object Manager along with other information provided by the user to create complete URI for each CoAP resource.

## 7.4. Usability study for Smart-Space use-case

As the proposed system is focused on providing a DIY development environment for enabling mass involvement in IoT applications development, its usability from the perspective of end-user with different levels of programming skills can ultimately judge its performance. The Smart-Space use-case has been developed to test the usability of the proposed architecture in the IoT enabled Smart-Space domain. The study presented in this chapter is a two pronged study. The first part captures the performance of users from the DIY perspective. This study allows the user to interact with the system without any prior training and captures the performance data. After that the user is trained briefly and the user is allowed to interact with the system in the

same manner capturing any difference in the performance of the users. The second part of the study utilizes the System Usability Scale (SUS) [53] for the users to assess the system usability from user's perspective. SUS is a low-cost usability measurement procedure which correctly categorizes the usability of a system even with smaller user groups. Table 3 provides the list of items for SUS upon which the participant is asked to rate the system. SUS is a 10 item Likert scale through which the users subjectively rate their experience with the system on a scale of 1 (Strongly Disagree) to 5 (Strongly Agree).

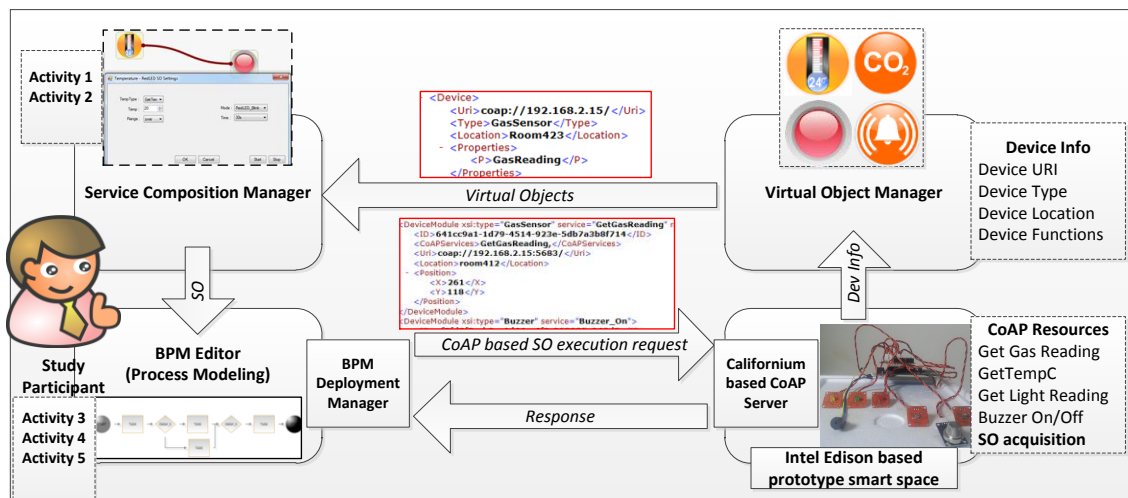


Figure 49: Test scenario for usability assessment of smart space use-case

The test scenario for the smart space use-case is illustrated in the Figure 49. The XML snippet termed as Virtual Objects in the figure shows the virtual representation of physical IoT devices being sent from VOM to SCM while the second snippet shows the definition of an individual Service Object (part of the BPM created by user as the definition of Smart Space behavior) being sent to CoAP enabled Smart Space controller for execution. The response is sent back to the Deployment Manager where further execution of the BPM based process is decided.

The Service Composition Layer and Business Process Layer for this use-case implementation have been slightly changed so that it can be utilized for usability experiments.



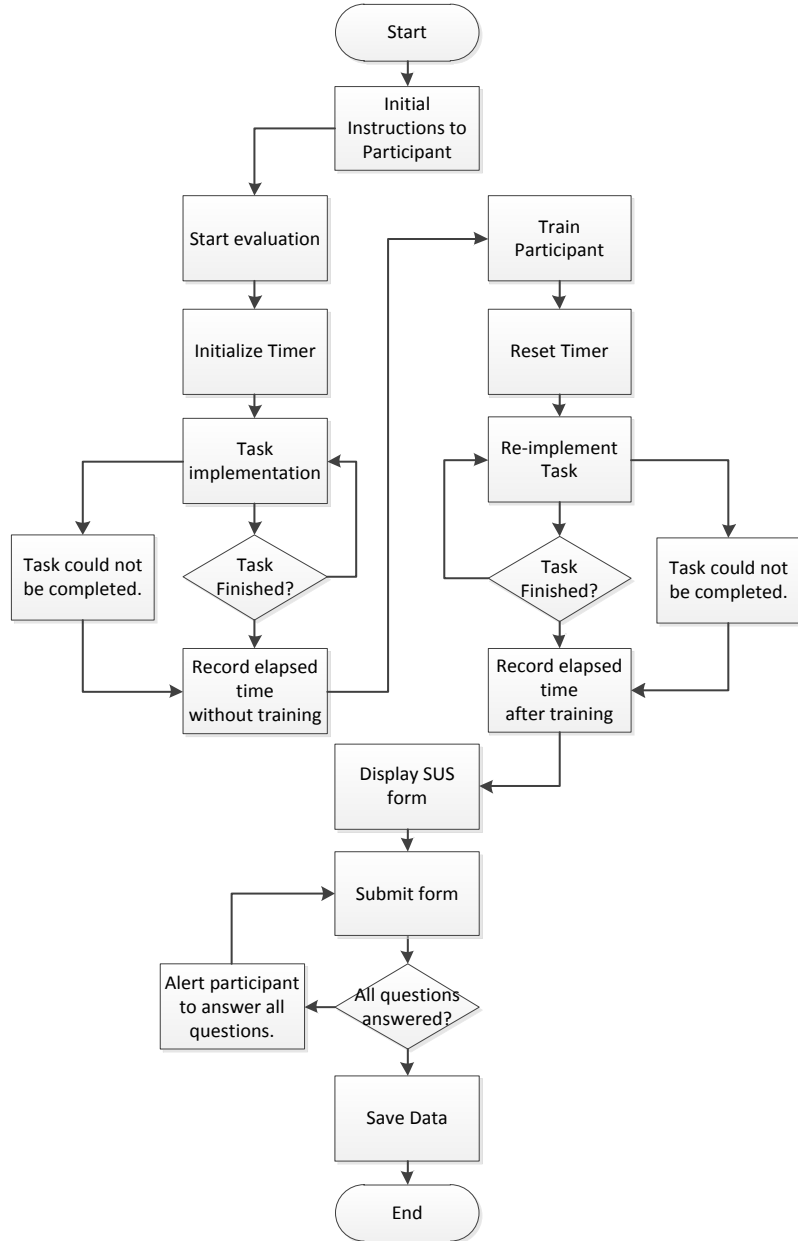
The flows of the usability experiment at the Service Composition Layer and Business Process Layer is given in the following sub-sections.

### **7.4.1. SCM usability assessment**

Figure 50 illustrates the various steps of the usability analysis experiment designed for the Service Composition Manager (SCM). The participant is given a brief description of a service composition task with a visual cue related to the task that the participant is asked to implement using the SCM.

The participant is allowed to interact with the service composition manager in order to complete the task. A supervisor checks the progress of the participant. The time taken by the participant in the first step is recorded along with the information if the task was successfully completed or not. In the next step of the experiment, the participant is given a brief session of training on how to use the SCM in order to compose a service. Once the training session is completed, the participant is asked to implement the same task again using the SCM. Again the elapsed time and the completion of the task are noted.

Once the interaction session of the participant is completed, a questionnaire based on System Usability Survey is presented to the participant by the service composition manager in order to rate the system based on his/her interaction experience. The participant also provides age and gender information but no other personal information is recorded to keep the data anonymous. The participant is asked to rate his/her level of computer knowledge on a scale of 1 to 10. This information is not part of the SUS questionnaire but used for associating the SUS results with age and computer knowledge for better assessment of the system usability.



**Figure 50: Flow of SCM usability analysis experiment for smart space use-case**

For the usability analysis of the smart space use-case, a total of 18 participants took part in the experimental setup explained below. The usability analysis of the proposed system was done at Jeju National University, so most of the participants of the experiments were international students and native Korean students from various departments along with some participants

external to the university. The participants mostly belonged to the age groups between 21 to 35 years with few outliers. The participants included 13 male and 5 female participants. The experimental setup is summarized in Table 14.

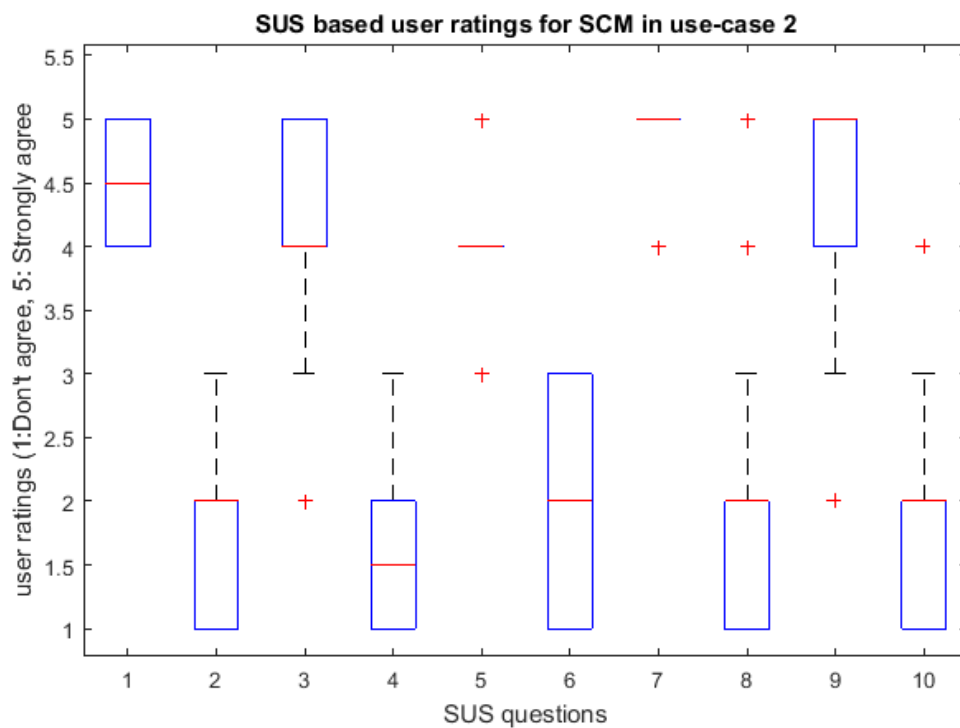
**Table 14: SCM usability assessment experimental setup based on smart space use-case**

<b>Test use-case</b>		Prototype Smart Space
<b>Number of Participants</b>		18
		Programmers
		Non-programmers
		9
		9
<b>Test environment</b>		Service Composition Manager (SCM)
Activity 1	SCM Task without training	Creation of a Service Object for controlling an LED module based on a specific range of temperature value as read from the Thermistor module.
	Data recorded	Time to complete the task. Successful/unsuccessful completion of the task
Brief training session on how to use the SCM		
Activity 2	SCM task after training	Creation of a Service Object for controlling an LED module based on a specific range of temperature value as read from the Thermistor module.
	Data recorded	Time to complete the task. Successful/unsuccessful completion of the task
Activity 3	Usability rating	Participant provides personal information and SUS questionnaire based system usability ratings.
	Data recorded	SUS based participant's responses for 10 questions.

### 7.4.1.1. SCM usability score based on SUS

All the participants in the experiment provided SUS based responses to the 10 questions presented in Table 3. Figure 51 presents a question-wise summary boxplot of the responses from all of the participants. The figure shows that for the odd number of SUS questions, the participants mostly agreed by giving the ratings between 4 and 5. The median of responses for Q7 and Q9 lies at 5 (Strongly agree) while for Q1, Q3, and Q5 the median response given by the participants lies at 4.5, 4 and 4 respectively showing strong agreement with the positive

statements regarding the usability of the SCM. For the even numbered questions with negative remarks about the system, the median responses by the participants for Q2, Q6, Q8 and Q10 lies at 2 (Disagree) while for Q4 the median response by all the participants is 1.5. This shows a strong disagreement of the study participants with the negative remarks regarding the usability of the SCM in Smart Space scenario. The '+' signs in the boxplot indicate any outlier values in the responses data set recorded by 18 participants based on System Usability Scale.

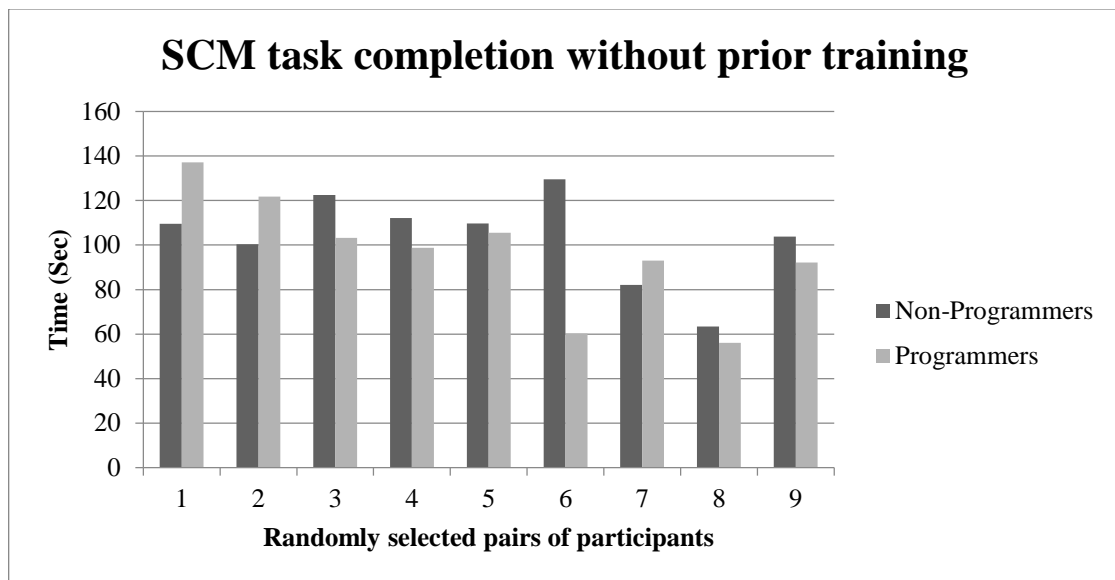


**Figure 51: Results of SCM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Smart space use-case)**

All the responses from the 18 participants were utilized to calculate the usability score for the SCM based on the smart space use-case. The average usability score for the SCM is 80.6 which shows very positive response from all the participants and indicates that the system is highly useable for end-user control and customization of smart space implementations.

### 7.4.1.2. SCM usability from DIY perspective

The analysis presented in this section is intended to check the usability of Service Composition Manager from the perspective of a DIY platform. For this purpose, statistical analysis of the time data recorded during the usability experiment for tasks performed by the participants has been conducted.

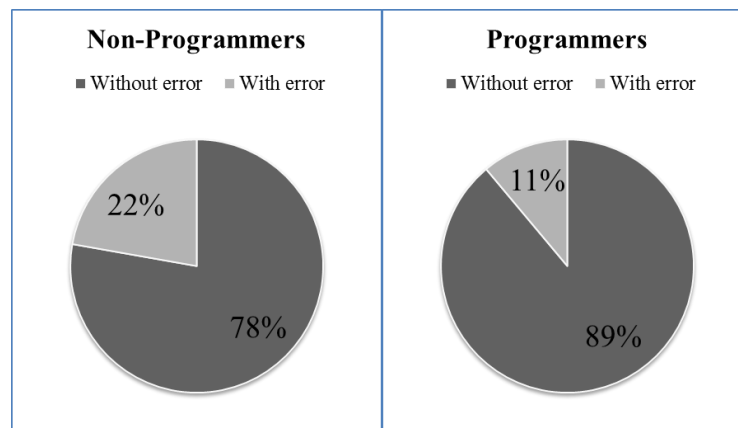


**Figure 52: Sample comparison of the time taken by both groups to complete SCM task without any prior training (smart space use-case)**

In order for the SCM to be a DIY platform for service composition, it should be equally useable to people with less or no programming skills as it is to people with programming skills. Based on the participant's subjective ratings of their programming skills, two groups of participants were defined. Participants with programming skills above 3 were included in the 'Programmers' (9) group while the rest were grouped as the 'Non-programmers' (9).

Two-Sample t-Test analysis was performed for the two groups based on the time taken by each participant to complete the first service composition task without any training given to the

participants. Figure 52 provides a comparison graph of the sample data recorded for both groups before any training was given to the participants. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 1 using the SCM in the Smart-Space scenario.



**Figure 53: Percentage for successful SCM task completion by Programmers and Non-Programmers (Smart space scenario)**

As there was no prior training, participants from both groups made errors while completing the activity 1. Figure 53 provides a comparison of the percentage of each group completing the task with or without errors. Table 15 shows the outcome of the t-Test analysis based on the null hypothesis (H0) that the mean time in seconds taken by programmers to complete the task without training is not significantly different than the mean time taken by the non-programmers to complete the same task without any prior training (Hypothesized Mean Difference =0).

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

For the null hypothesis to be rejected, t Stat should be greater than t Critical two-tail. This is not the case as described in Table 15,  $0.66 < 2.13$ , hence the null hypothesis is accepted which

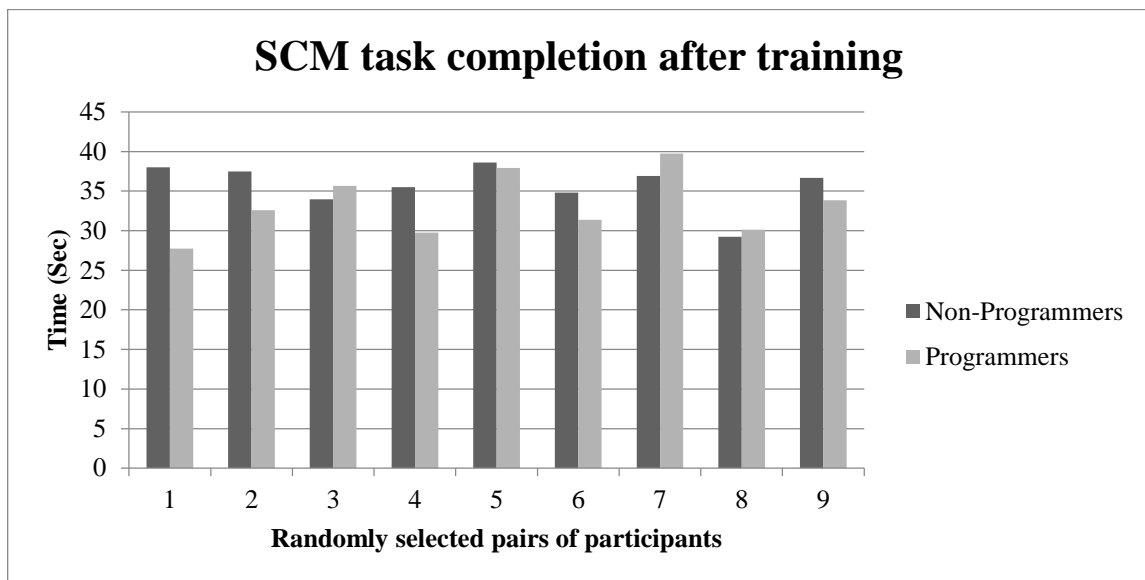
means that there was no statistically significant difference in the time taken by programmers to complete the service composition task without any prior training ( $M=102.67$ ) and the time taken by non-programmers to complete the task without any prior training ( $M=96.43$ );  $t(15)=2.13$ ,  $p = 0.66$ . This result supports the fact that in the first experimental analysis for the smart space use-case, the difference in time to complete the service composition task without any prior training for both groups was very small and shows that even without prior training anyone can utilize SCM effectively without the need to have special technical skills.

**Table 15: Two-Sample t-Test for unequal variances (Activity 1 before training)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	103.6714	96.43367
Variance	406.5848	674.2933
Observations	9	9
Hypothesized Mean Difference	0	
df	15	
t Stat	0.660447	
P(T<=t) two-tail	0.518981	
t Critical two-tail	2.13145	

The change in the results of the smart space use-case can be attributed to the fact that people are generally more informed about the common devices (temperature sensors, LEDs etc.) used for the smart space scenario. The mean time to complete the service composition task without any prior training for both groups indicates that all the participants took more time to complete the smart space based task as compared to the robotic arm control task. This is due to the fact that the smart space service composition task involved setting the conditions for input devices in order to control the behavior of output devices which was not the case in the smart space use-case.

A second Two-Sample t-Test was performed for the two groups based on the time taken by each participant to complete the same service composition task after a brief session of training given to the participants. Figure 54 provides a comparison graph of the sample data recorded for both groups after training was given to the participants. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 2 using the SCM in the Smart-Space scenario.



**Figure 54: Sample comparison of the time taken by both groups to complete SCM task after the training (smart space use-case)**

Table 16 shows the outcome of the t-Test based on the null hypothesis (H0) that the mean time taken by programmers to complete the task after the training is not significantly different



than the mean time taken by the non-programmers to complete the same task after the training session (Hypothesized Mean Difference =0).

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

As shown in Table 16, t Stat is not greater than t Critical two-tail ( $1.53 < 2.13$ ), hence the null hypothesis is accepted. The test specifies that there is no statistically significant difference in the time taken by programmers to complete the service composition task after the training session ( $M=33.20$ ) and time taken by non-programmers to complete the task after the training session ( $M=35.69$ );  $t(9)=1.52$ ,  $p = 0.147$ . The task completion rate without any errors for groups was 100 percent indicating the effectiveness of the SCM with minimal amount of training to the users.

**Table 16: Two-Sample t-Test for unequal variances (Activity 2 after training)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	35.68767	33.19644
Variance	8.14964	15.77758
Observations	9	9
Hypothesized Mean Difference	0	
df	15	
t Stat	1.527874	
P(T<=t) two-tail	0.147354	
t Critical two-tail	2.13145	

The statistical analysis shows that the mean time taken by both groups to complete the service composition task after the training has significantly reduced. The absence of any statistical difference in the time for both groups indicates that SCM is suitable for the DIY usage and that it can be utilized for DIY service composition in the smart space domain.

## 7.4.2. BPM Editor usability assessment

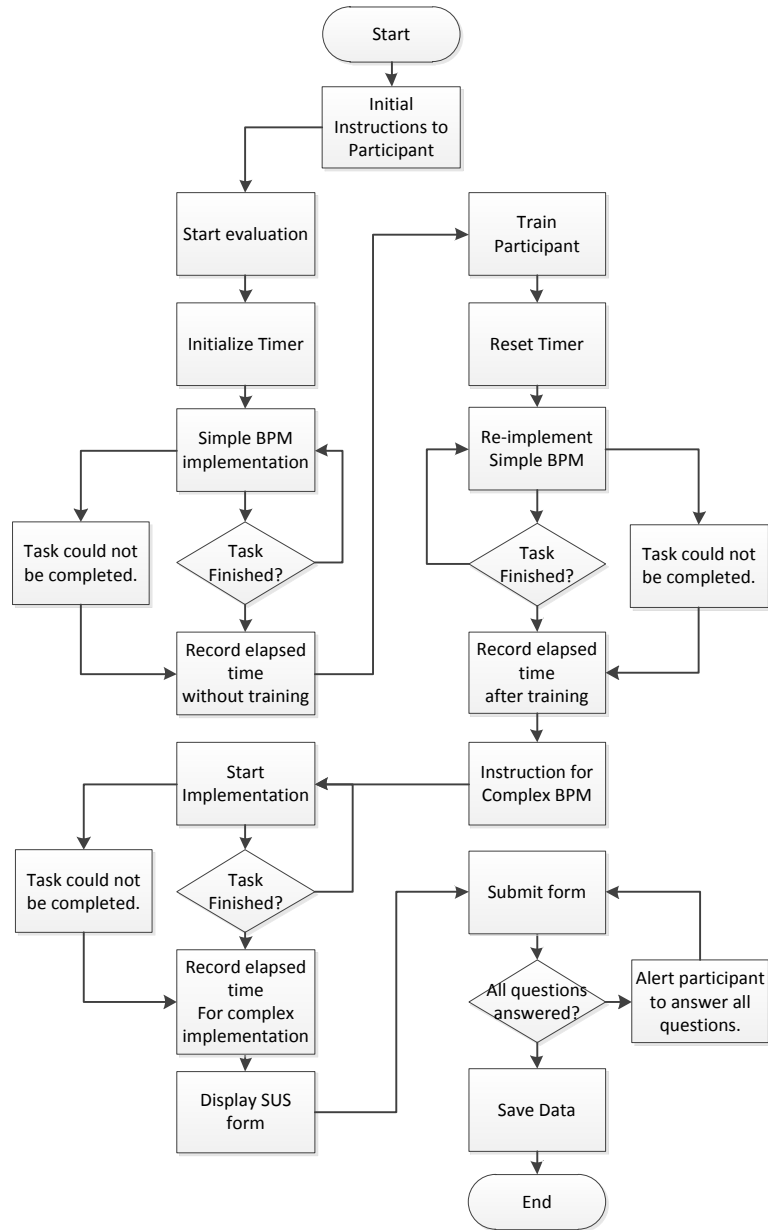


Figure 55: Flow of BPM Editor usability experiment for smart space use-case

Figure 55 illustrates the various steps of the usability analysis experiment designed for the Business Process Modeling Editor at the Business Process Layer of the proposed system. The participant is given a brief description of a simple process based on the previously composed

services with a visual cue related to the final model of the process that the participant is asked to implement using the BPM Editor. The participant is allowed to interact with the editor in order to model the process by simple drag-n-drop and mouse click operation on the graphical BPM notations. A supervisor checks the progress of the participant. The time taken by the participant in the first step is recorded by the application along with the information if the task was successfully completed or not.

**Table 17: BPM Editor usability assessment experimental setup based on smart space use-case**

<b>Test use-case</b>		Prototype Smart Space
<b>Number of Participants</b>		18
		Programmers
		Non-programmers
		9
		9
<b>Test environment</b>		Business Process Modeling Editor (BPM Editor)
Activity 1	Simple BPM task without training	Three step BPM creation task to for the control of prototype smart space behavior.
	Data recorded	Time to complete the task. Successful/unsuccessful completion of the task
Brief training session on how to use BPM Editor		
Activity 2	Simple BPM task after training	Three step BPM creation task to for the control of prototype smart space behavior.
	Data recorded	Time to complete the task. Successful/unsuccessful completion of the task
Activity 3	Complex BPM task	6 Step BPM task utilizing gateway notations and script notations to create a complex process model for smart space behavior control.
	Data recorded	Time to complete the task. Successful/unsuccessful completion of the task
Activity 4	Usability rating	Participant provides personal information and SUS questionnaire based system usability ratings.
	Data recorded	SUS based participant's responses for 10 questions.

In the next step of the experiment, the participant is given a brief session of training on how to use the BPM Editor in order to visually create a business process model. Once the training session is completed, the participant is asked to create the same business process model again

using the BPM editor. Again the elapsed time and the completion of the task are recorded by the system.

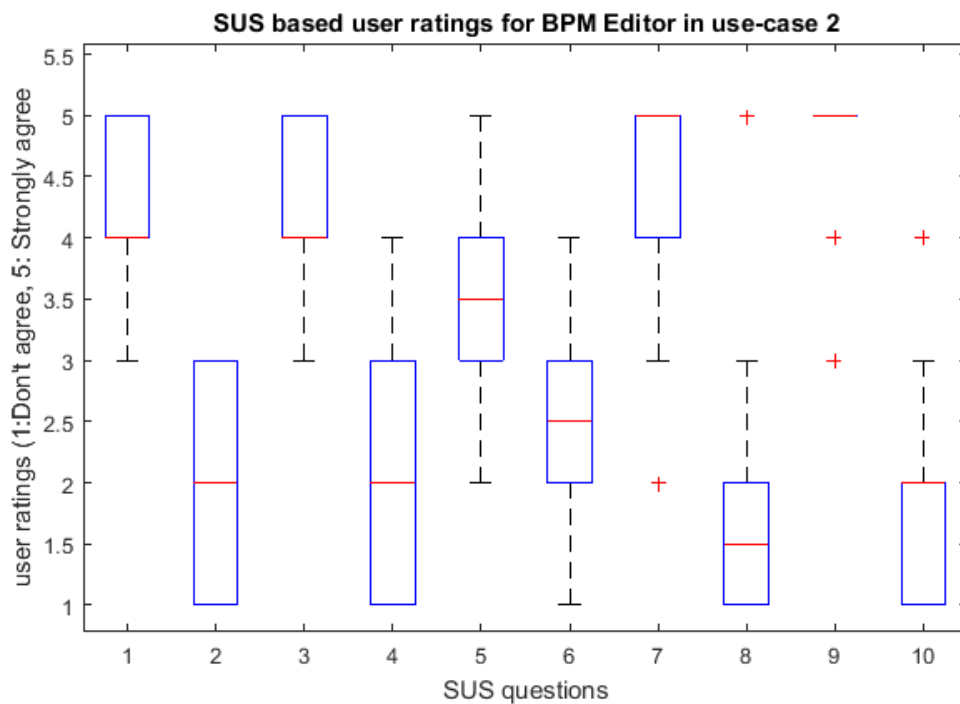
As business process models can become complex for larger processes, an extra step has been added to this experiment to let the participant assess the system based on a complex process modeling task. In this step, the participant is given the description of a more complex process model to be visually created using the BPM editor. The participant tries to visually create the corresponding process model using the graphical notations. Again the time taken by the participant to complete the task or to withdraw from the step is recorded by the system. Once the interaction session of the participant is completed, a System Usability Scale based assessment form is presented to the participant by the system in order to rate the system based on his/her interaction experience.

The participant also provides age and gender information but no other personal information is recorded to keep the data anonymous. The participant is asked to rate his/her level of computer programming skill on a scale of 1 to 5. This information is not part of the SUS questionnaire but used for further usability analysis of the system from a DIY perspective. Table 17 provides a summary of the experimental setup.

#### **7.4.2.1. BPM Editor usability score based on SUS**

All the participants in the experiment provided SUS based responses to the 10 questions presented in Table 3. Figure 56 presents a question-wise summary boxplot of the responses from all of the participants. The figure shows that the median of responses for odd numbered questions of SUS Q1, Q3, Q5, Q7 and Q9 lies at 4, 4, 3.5, 5 and 5 respectively. This shows a strong agreement of the study participants with the positive statements regarding the usability of the BPM Editor in Smart-Space scenario. For the even numbered questions with negative

remarks about the system's usability, the median responses by the participants for Q2, Q4, Q6, Q8 and Q10 lies at 2, 2, 2.5, 1.5 and 1 respectively. These median responses show a strong disagreement of the study participants with the negative remarks regarding the usability of the BPM Editor in Smart-Space scenario. The '+' signs in the boxplot indicate any outlier values in the responses data set recorded by 18 participants based on System Usability Scale.



**Figure 56: Results of BPM usability study based on SUS. Rating values range from 1: "Don't agree" to 5: "Strongly agree" (Smart space use-case)**

All the responses from the 18 participants were utilized to calculate the usability score for the Business Process Modeling Editor based on the use-case 2 scenario. The average usability score for the BPM Editor is 81.95 which shows very positive response from all the participants and indicates that the BPM based IoT application development for the smart space domain is highly useable based on user ratings.

### 7.4.2.2. BPM Editor usability from DIY perspective

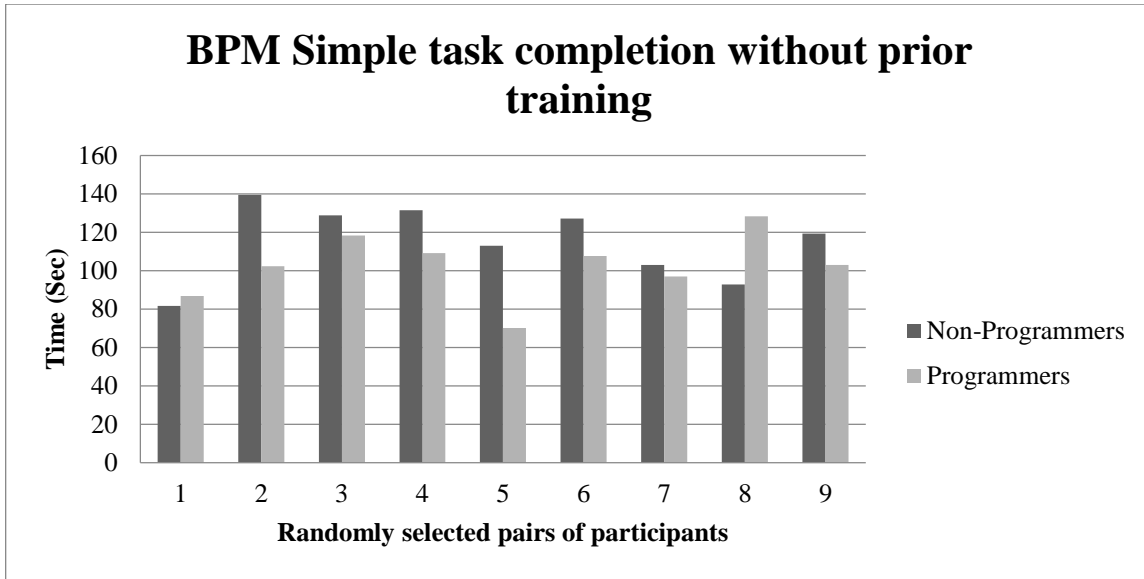
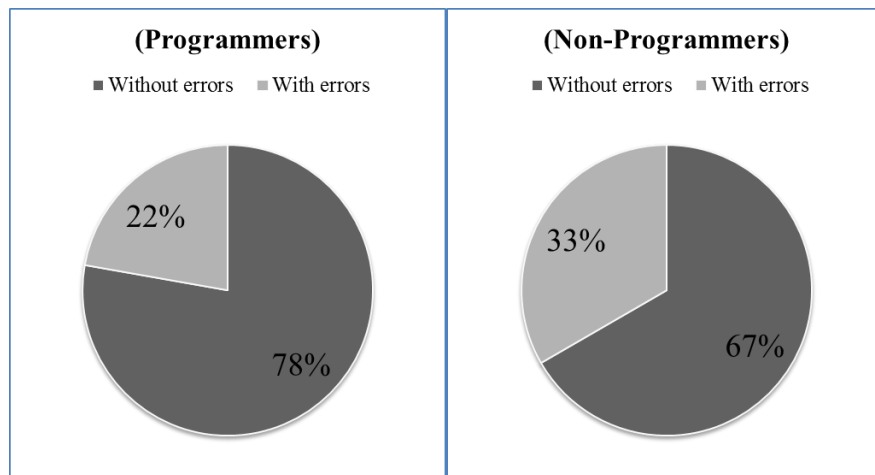


Figure 57: Sample comparison of the time taken by both groups to complete BPM simple task without any prior training (Smart space use-case)

In the second part of the usability analysis experiment for the second use-case, the participants were asked to create a simple BPM for controlling the prototype smart space without any prior training on how to use the proposed BPM Editor. Figure 57 provides a comparison graph of the sample data recorded for both groups. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the 'Non-Programmers' group with a randomly selected participant from the 'Programmers' group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 1 using the BPM Editor for the Smart-Space scenario.

As there was no prior training, participants from both groups made errors while completing the activity 1. Figure 58 provides a comparison of the percentage of each group completing the task with or without errors.



**Figure 58: Comparison of percentage successful completion of simple BPM task by Programmers vs Non-Programmers without any prior training (smart space use-case)**

A Two-Sample t-Test analysis was performed for the two groups (non-programmers, programmers) based on the time taken by each group member to complete the simple BPM task without any training given to the participants. Table 18 shows the outcome of the t-Test analysis based on the null hypothesis ( $H_0$ ) that the mean time taken by programmers to complete the task without training is not significantly different than the mean time taken by the non-programmers to complete the same task without any prior training (Hypothesized Mean Difference =0).

$$H_0: \mu_0 - \mu_1 = 0$$

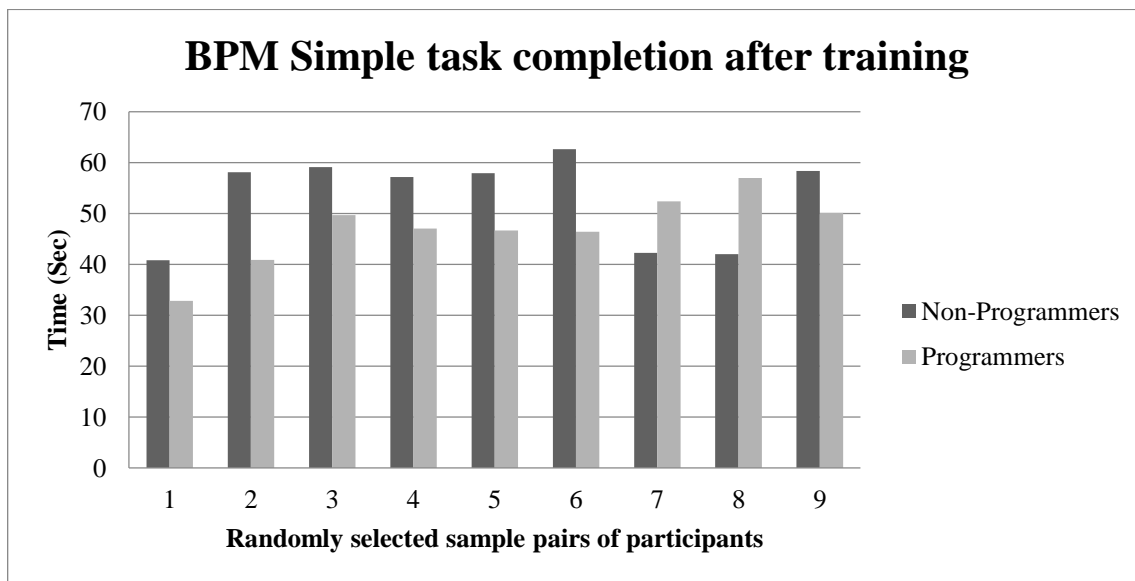
$$H_1: \mu_0 - \mu_1 \neq 0$$

Based on the outcome of the t-Test analysis, the null hypothesis is accepted which means that there is a significant difference in the time taken by programmers to complete the BPM task

without any prior training (M=102.58) and time taken by non-programmers to complete the task without any prior training (M = 115.24);  $t(16) = 1.48$ ,  $p = 0.159$ .

**Table 18: Two-Sample t-Test for unequal variances (Simple BPM task without prior training, smart space use-case)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	115.238	102.5856
Variance	373.1667	287.9604
Observations	9	9
Hypothesized Mean Difference	0	
df	16	
t Stat	1.476227	
P(T<=t) two-tail	0.159292	
t Critical two-tail	2.119905	



**Figure 59: Sample comparison of the time taken by both groups to complete BPM simple task after the training (Smart space use-case)**

The analysis indicates that although there is numerical difference between the mean time taken by both groups to complete the simple BPM task but it is not statistically significant.



Hence, it is safe to state that BPM Editor is usable equally for people with no or less programming skills as it is for programmers. This analysis backs the proposed idea that BPM Editor can be used as a DIY IoT application composer. The results also supports the fact that the BPM Editor usability experiment in the robotic arm use-case also resulted in a small difference of time taken by both the groups to complete the simple BPM task but just not small enough to prove the null hypothesis true.

To further strengthen the claim, a second Two-Sample t-Test was performed for the two groups based on the time taken by each group member to complete the same BPM task after a brief session of training given to the participants. Figure 59 provides a comparison graph of the sample data recorded for both groups to complete the simple task after the training session. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 2 using the BPM Editor for the Smart-Space scenario.

**Table 19: Two-Sample t-Test for unequal variances (Simple BPM task after training, smart space use-case)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	53.17033	47.00756
Variance	76.32227	47.95783
Observations	9	9
Hypothesized Mean Difference	0	
df	15	
t Stat	1.658429	
P(T<=t) two-tail	0.11799	
t Critical two-tail	2.13145	

Table 19 shows the outcome of the t-Test based on the null hypothesis (H<sub>0</sub>) that the mean time taken by programmers to complete the task after the training is not significantly different than the mean time taken by the non-programmers to complete the same task after the training session (Hypothesized Mean Difference =0).

$$H_0: \mu_0 - \mu_1 = 0$$

$$H_1: \mu_0 - \mu_1 \neq 0$$

According to the outcome of the t-Test analysis, the null hypothesis is accepted. It means that there is no statistically significant difference found in the time taken by programmers to complete the BPM task (M=47.01) and the time taken by non-programmers to complete the task (M=53.17) after both groups had the short training session on how to use the BPM Editor; t(15)=1.66, p = 0.117. It is thus deduced, that both groups performed almost the same after the brief training session with a significant reduction in the mean time taken to complete the Simple BPM task. It is however, notable that the percentage of successful task completion without any error by both the groups rose to 100 percent, an indication of how easily anyone can learn to use the BPM based DIY IoT composition platform.

To further test the DIY nature of the BPM Editor in smart space scenario, both the groups of participants were asked to complete a more complex BPM task. The time in seconds for programmers and non-programmers to complete the task has been presented in

Figure 60. The graph shows the comparison of time in seconds, taken by randomly selected individual participant from the ‘Non-Programmers’ group with a randomly selected participant from the ‘Programmers’ group. A total of 9 sample comparisons are shown in the graph which shows a numerical difference in the mean time taken by both groups to complete the activity 3 using the BPM Editor for the Smart-Space scenario. The percentage successful completion of the complex BPM task without any errors is shown in Figure 61. The graphs illustrate that

programmers and non-programmers regardless of their programming skills level can utilize the BPM Editor with the same success rate.

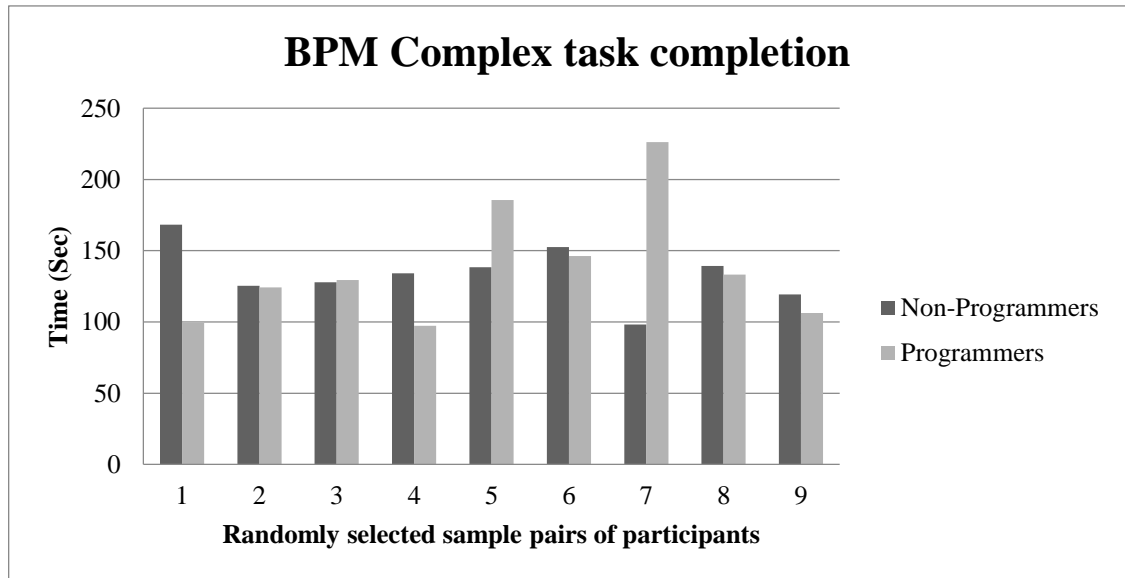


Figure 60: Sample comparison of the time taken by both groups to complete BPM complex task (smart space use-case)

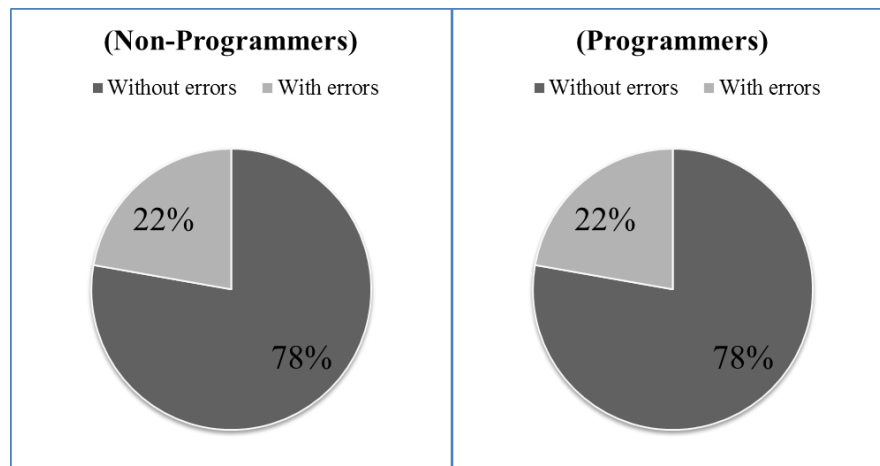


Figure 61: Comparison of percentage successful completion of BPM complex task by Programmers vs Non-Programmers (smart space use-case)

To statistically analyze the data a Two-Sample t-Test assuming unequal variances has been conducted. The null hypothesis (H0) states that the mean time taken by programmers to complete the complex BPM task is not significantly different than the mean time taken by the non-programmers to complete the same task (Hypothesized Mean Difference =0). Table 20 presents the outcome of the t-Test.

**Table 20: Two-Sample t-Test for unequal variances (BPM complex task, Smart space use-case)**

	<i>Non-Programmers</i>	<i>Programmers</i>
Mean	133.6898	138.6546
Variance	398.1995	1819.463
Observations	9	9
Hypothesized Mean Difference	0	
df	11	
t Stat	-0.31628	
P(T<=t) two-tail	0.75771	
t Critical two-tail	2.200985	

According to the outcome of the t-Test analysis, the null hypothesis is accepted. It means that there is no statistically significant difference found in the time taken by programmers to complete the complex BPM task (M=138.65) and the time taken by non-programmers to complete the same task (M = 133.58);  $t(11) = -0.316$ ,  $p = 0.758$ .

The means for each sample and the negative t Stat value indicates that in this case the programmer took more time to complete the complex BPM task than the time taken by non-programmers. Figure 61 shows that the percentage success rate without errors for both the groups while completing the BPM complex task is exactly the same showing same behavior of any participant regardless their programming skills level.

Based on the Skewness (1.34) and Kurtosis (2.46) values for the sample data, it cannot be safely said that the time data collected for both groups to complete the BPM complex task is normal enough. Hence, a non-parametric statistical analysis is conducted in the form of Mann-Whitney U. Mann-Whitney U is mostly used in situations where the data sample for analysis does not follow a normal distribution. Table 21 provides all the details of finding Mann-Whitney U for the time taken by Programmers and non-programmers to complete the BPM complex task with the null hypothesis (H0) that there is no statistically significant difference in the time for both groups.

**Table 21: Mann-Whitney-U Test for non-parametric equivalence (BPM complex task, Smart space use-case)**

Data samples		18
Number of Non-Programmers (group 1)	n1	9
Number of Programmers (group 2)	n2	9
R1		74
R2		97
U1		52
U2		29
t-Stat (from Mann-Whitney U table)		17

As the smaller U (U2 in this case) is greater than t-Stat ( $29 > 17$ ), hence the null hypothesis is accepted proving that the difference in time taken by each group to complete the BPM complex task is statistically not significant and that BPM Editor can provide a DIY programming interface to anyone regardless of their programming skills level. Hence, it can be deduced from the analysis that the BPM Editor can be utilized as an effective DIY development environment for smart space related IoT application and process modeling.

## 8. Conclusion

The vision of Internet of Things is a global network of diverse sensing and actuating devices for the provision of useful services via data acquisition, communication, data sharing and actuations. Global adaptation of the IoT vision requires mass involvement of general population and the lack of technical and programming skills on behalf of the common people drives the motivation to towards the development of Do-It-Yourself type IoT development platforms and architectures.

This study proposes the novel idea of enhanced IoT composition architecture based on DIY Business Process Modeling approach. The presented system utilizes the prevalent ideas of object virtualization and service provisioning and represents them as a DIY interface based on a Business Process Modeling (BPM). BPM has been at the core of software requirement analysis and specification processes. It fills the communication gap between the clients and developers by providing a standardized set of graphical notation called as the Business Process Modeling Notations (BPMN). BPMN is easy to learn and can be globally interpreted into the same description of a process. Although there are efforts to extend BPMN to incorporate IoT concepts but we believe that a standardized modeling language such as BPMN can provide better DIY environment for IoT application development. It is specifically important from the IoT point of view because IoT is an emerging field and mass involvement has been reported to be very necessary for the successful realization of IoT vision. The DIY approach of development in such a scenario may prove very beneficial as it eliminates the requirement of programming skills on behalf of the users or general public.

The study visualizes the presented idea in the form of a layered architecture which consists of the Physical Layer, Virtual Object 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 document. In order to demonstrate the applicability and usability of the proposed architecture in multiple domains, two use-cases have been selected from industrial robotics and smart space domain respectively. Prototype implementations based on CoAP devices and services have been developed to demonstrate the feasibility of the proposed system and to perform usability studies for each use-case. For the usability studies, the System Usability Scale (SUS) based preliminary usability studies have been carried out. Apart from the SUS based usability scoring, further statistical analysis has been conducted for both the use-case scenarios to assess the usability of the proposed architecture from a DIY perspective. For this purpose, participants' task completion time during the experiments was recorded. Data was grouped based on the reported programming skills of the participants and t-Test analysis was carried out for both the groups to determine if the system can provide a better DIY environment regardless of the programming skills of the users. The results of extensive analysis based on the two use-case scenarios indicate that the proposed system can provide better DIY programming environment to anyone in multi-domain setups.

## References

- [1] W. S., “Business process modeling improves administrative control,” *Automation*, pp. 44–50, 1967.
- [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] M. Rosen, B. Lublinsky, and K. T. Smith, *Applied SOA*. Indianapolis: Wiley Publishing, Inc. 10475 Crosspoint Boulevard Indianapolis, IN 46256, 2008.
- [4] V. Stiehl, *Process-Driven Applications with BPMN*. Springer International Publishing, 2014.
- [5] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, “The Web of Things: Interconnecting Devices with High Usability and Performance,” in *2009 International Conference on Embedded Software and Systems*, 2009, pp. 323–330.
- [6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
- [7] a P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, “Web Services for the Internet of Things through CoAP and EXI,” *Commun. Work. (ICC), 2011 IEEE Int. Conf.*, no. Xml, pp. 1–6, 2011.
- [8] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” 2014.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] N. Eddy, “Gartner: 21 Billion IoT Devices To Invade By 2020 - InformationWeek,” 2015. [Online]. Available: <http://www.informationweek.com/mobile/mobile-devices/gartner-21-billion-iot-devices-to-invade-by-2020/d/d-id/1323081>. [Accessed: 08-Jan-2016].
- [13] M. Wolf and S. McQuitty, “Understanding the do-it-yourself consumer: DIY motivations and outcomes,” *AMS Rev.*, vol. 1, no. May 2016, pp. 154–170, 2011.
- [14] A. Aluva, “Why DIY Concept Has Started Trending Among Indian Startups - Inc42 Media,” 2015. [Online]. Available: <https://inc42.com/resources/startups-adopting-do-it-yourself-models-diy-concept-trending-among-indian-startups/>. [Accessed: 24-May-2016].
- [15] D. De Roeck, K. Slegers, J. Criel, M. Godon, L. Claeys, K. Kilpi, and A. Jacobs, “I



- would DiYSE for it!,” *Proc. 7th Nord. Conf. Human-Computer Interact. Mak. Sense Through Des. - Nord. '12*, p. 170, 2012.
- [16] L. J. Bannon, “Forgetting as a feature, not a bug: the duality of memory and implications for ubiquitous computing,” *CoDesign*, vol. 2, no. 1, pp. 3–15, 2006.
- [17] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
- [18] 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.
- [19] 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.
- [20] C. Anderson, “Makers: The New Industrial Revolution,” *Compet. Rev.*, vol. 24, no. 2, pp. 147–149, Mar. 2014.
- [21] 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.
- [22] H. Michael, *Essential business process modeling*. 2005.
- [23] J. Barjis, “The importance of business process modeling in software systems design,” *Sci. Comput. Program.*, vol. 71, no. 1, pp. 73–87, 2008.
- [24] D. (AccuProcess I. . Singh, “5 Key Benefits of Business Process Modeling > Business Analyst Community & Resources | Modern Analyst.” [Online]. Available: <http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/1728/5-Key-Benefits-of-Business-Process-Modeling.aspx>. [Accessed: 04-Jun-2016].
- [25] M. Chinosi and A. Trombetta, “BPMN: An introduction to the standard,” *Comput. Stand. Interfaces*, vol. 34, no. 1, pp. 124–134, 2012.
- [26] R. Want, B. N. Schilit, and S. Jenson, “Enabling the internet of things,” *Computer (Long Beach. Calif.)*, vol. 48, no. 1, pp. 28–35, 2015.
- [27] S. Flowers, M.-G. Juan, J. Sapsed, P. Nightingale, A. Grantham, and G. Voss, “The New Inventors: How users are changing the rules of innovation,” 2008.
- [28] R. Kleinfeld, S. Steglich, L. Radziwonowicz, and C. Doukas, “Glue.Things: A Mashup Platform for Wiring the Internet of Things with the Internet of Services,” *Proc. 5th Int. Work. Web Things*, pp. 16–21, 2014.
- [29] Roberto, “Introduction to Node RED | Sensetecnic,” 2015. [Online]. Available: <http://developers.sensetecnic.com/article/introduction-to-node-red/>. [Accessed: 19-Apr-2016].
- [30] S. Heo, S. Woo, J. Im, and D. Kim, “IoT-MAP : IoT Mashup Application Platform for the Flexible IoT Ecosystem,” in *5th International Conference on the Internet of Things (IoT)*, 2015, pp. 163–170.
- [31] C. Richardson, “Overview of POJO programming,” 2006.

- [32] F. Pramudianto, C. A. Kamienski, E. Souto, F. Borelli, and L. L. Gomes, “IoTLink : An Internet of Things Prototyping Toolkit,” in *11th International Conference on Ubiquitous Intelligence & Computing*, 2014, pp. 1–9.
- [33] H. Nguyen, M. Quoc, and M. Serrano, “Super Stream Collider–Linked Stream Mashups for Everyone,” in *Semantic Web Challenge co-located with ISWC2012*, 2012.
- [34] D. Carlson, M. Mögerle, M. Pagel, S. Verma, and D. S. Rosenblum, “Ambient Flow: A visual approach for orchestrating smart devices in the internet of things,” in *5th International Conference on the Internet of Things (IoT)*, 2015.
- [35] N. Kefalakis, J. Soldatos, A. Anagnostopoulos, and P. Dimitropoulos, “A Visual Paradigm for IoT Solutions Development,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9001, pp. 169–182, 2015.
- [36] J. Coutaz and J. L. Crowley, “A First-Person Experience with End-User Development for Smart Homes,” *IEEE Pervasive Comput.*, vol. 15, no. 2, pp. 26–39, 2016.
- [37] M. M. Burnett and B. a. Myers, “Future of end-user software engineering: beyond the silos,” *Proc. Futur. Softw. Eng. - FOSE 2014*, no. April 2016, pp. 201–211, 2014.
- [38] A. Salihbegovic, T. Eterovic, E. Kaljic, and S. Ribic, “Design of a domain specific language and IDE for Internet of things applications,” *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. May, pp. 996–1001, 2015.
- [39] “openHAB.” [Online]. Available: <http://www.openhab.org/>. [Accessed: 19-Apr-2016].
- [40] J. Munoz, V. Pelechano, and C. Cetina, “Implementing a pervasive meeting room: A model driven approach,” *Int. Work. Ubiquitous Comput. (IWUC 2006)*, pp. 13–20, 2006.
- [41] B. Bertran, J. Bruneau, D. Cassou, N. Lorient, C. Consel, B. Bertran, J. Bruneau, D. Cassou, N. Lorient, and E. Balland, “DiaSuite : a Tool Suite To Develop Sense / Compute / Control Applications,” *Sci. Comput. Program.*, no. 4, 2012.
- [42] M. Kovatsch, “CoAP for the Web of Things: From Tiny Resource-constrained Devices to the Web Browser,” in *4th International Workshop on the Web of Things (WoT 2013)*, 2013.
- [43] M. Butcher, “REST Without JSON: The Future of IoT Protocols - DZone IoT,” 2015. [Online]. Available: <https://dzone.com/articles/json-http-and-the-future-of-iot-protocols>. [Accessed: 23-May-2016].
- [44] F. Wortmann and K. Fluchter, “Internet of Things,” *Bus. Inf. Syst. Eng.*, no. JANUARY, 2015.
- [45] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. Do Noh, “Smart Manufacturing : Past Research , Present Findings , and Future Directions,” *Int. J. Precis. Eng. Manuf. Technol.*, vol. 3, no. 1, pp. 111–128, 2016.
- [46] K. Cheng, “Keynote presentation—2: Smart tooling, smart machines and smart manufacturing: Working towards the Industry 4.0 and beyond,” *Autom. Comput.*, pp. 1–1, 2015.
- [47] T. Qu, S. P. Lei, Z. Z. Wang, D. X. Nie, and X. Chen, “IoT-based real-time production logistics synchronization system under smart cloud manufacturing,” *Int. J. Adv. Manuf.*

- Technol.*, pp. 1–8, 2015.
- [48] D. Zhang, J. Wan, C.-H. R. Hsu, and A. Rayes, “Industrial technologies and applications for the Internet of Things.,” *Comput. Networks*, pp. 1–4, 2016.
  - [49] M. Xia, T. Li, Y. Zhang, and C. W. De Silva, “Closed-loop design evolution of engineering system using condition monitoring through In-ternet of Things and cloud computing,” *Comput. Networks*, no. 2016, 2015.
  - [50] M. Wei, S. Ho, and M. Alam, “An IoT-based energy-management platform for industrial facilities,” *Appl. Energy*, vol. 164, pp. 607–619, 2016.
  - [51] R. Ramakrishnan and L. Gaur, “Application of Internet of Things ( IoT ) for Smart Process Manufacturing in Indian Packaging Industry,” *Inf. Syst. Des. Intell. Appl.*, pp. 339–346, 2016.
  - [52] J. Rivera and R. van der Meulen, “Gartner Says 4.9 Billion Connected ‘Things’ Will Be in Use in 2015,” 2014. [Online]. Available: <http://www.gartner.com/newsroom/id/2905717>. [Accessed: 05-May-2016].
  - [53] J. Brooke, “SUS - A quick and dirty usability scale,” *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.
  - [54] A. Bangor, P. Kortum, and J. Miller, “The system usability scale (SUS): An empirical evaluation,” *Int. J. Hum. Comput. Interact.*, vol. 24, no. 6, pp. 574–594, 2008.
  - [55] A. Bangor, P. Kortum, and J. Miller, “Determining what individual SUS scores mean: Adding an adjective rating scale,” *J. usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.
  - [56] D. G. Korzun, S. I. Balandin, and A. V. Gurtov, “Deployment of Smart Spaces in Internet of Things : Overview of the Design Challenges,” *Internet Things, Smart Spaces, Next Gener. Netw.*, no. August 2013, pp. 0–12, 2016.
  - [57] S. Helal and S. Tarkoma, “Smart Spaces,” *PERVASIVE Comput.*, pp. 22–23, 2015.
  - [58] M. Carmen, “An overview of the Internet of Things for people with disabilities,” *J. Netw. Comput. Appl.*, vol. 35, no. 2, pp. 584–596, 2012.
  - [59] D. Zafra, J. Medina, L. Martinez, C. Nugent, and M. Espinilla, “A Web System for Managing and Monitoring Smart Environments,” *Bioinforma. Biomed. Eng.*, vol. 1, pp. 677–688, 2016.
  - [60] Y. Hsueh, N. Lin, C. Chang, W. Lie, and I. Engineering, “Abnormal Event Detection Using Bayesian Networks at a Smart Home,” in *8th International Conference on Ubi-Media Computing (UMEDIA)*, 2015, pp. 273–277.
  - [61] B. Chen, Z. Fan, and F. Cao, “Activity Recognition Based on Streaming Sensor Data for Assisted Living in Smart Homes,” in *International Conference on Intelligent Environments Activity*, 2015.
  - [62] W. S. Lima, E. Souto, T. Rocha, R. W. Pazzi, and F. Pramudianto, “User Activity Recognition for Energy Saving in Smart Home Environment,” in *20th IEEE Symposium on Computers and Communication (ISCC) User*, 2015, pp. 751–757.
  - [63] T. Abiodun, M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C. Lung, “Smart Home : Integrating Internet of Things with Web Services and Cloud Computing,” in *IEEE*

*International Conference on Cloud Computing Technology and Science Smart*, 2013, no. April 2016.

- [64] J. Zhu, X. Jia, and X. Mei, “Smart home control system based on Internet of Things,” *Appl. Mech. Mater.*, vol. 738–739, pp. 233–237, 2015.
- [65] R. Piyare, “Internet of Things : Ubiquitous Home Control and Monitoring System using Android based Smart Phone,” *Int. J. Internet Things*, vol. 2, no. 1, pp. 5–11, 2013.
- [66] J. Kim, “HEMS ( Home Energy Management System ) Base on the IoT Smart Home,” *Contemp. Eng. Sci.*, vol. 9, no. 1, pp. 21–28, 2016.
- [67] A. Anvari-moghaddam, H. Monsef, and A. Rahimi-kian, “Optimal Smart Home Energy Management Considering Energy Saving and a Comfortable Lifestyle,” *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 324–332, 2015.
- [68] M. Choi, W. Park, and I. Lee, “Smart Office Energy-Saving Service Using Bluetooth Low Energy Beacons and Smart Plugs,” in *IEEE International Conference on Data Science and Data Intensive Systems Smart*, 2015, pp. 247–251.
- [69] C. Rottondi, M. Duchon, D. Koss, G. Verticale, and B. Schätz, “An Energy Management System for a Smart Office Environment,” in *International Conference and Workshops on Networked Systems (NetSys)*, 2015, pp. 1–6.
- [70] J. Jin, J. Gubbi, and S. Marusic, “An Information Framework for Creating a Smart City Through Internet of Things,” *IEEE INTERNET THINGS J.*, vol. 1, no. 2, pp. 112–121, 2014.
- [71] R. Jalali, K. El-khatib, and C. Mcgregor, “Smart City Architecture for Community Level Services Through the Internet of Things,” in *18th International Conference on Intelligence in Next Generation Networks*, 2015, pp. 108–113.