Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 6309509, 20 pages
https://doi.org/10.1155/2018/6309509

Research Article

A Model of Socially Connected Web Objects for IoT Applications

Sajjad Ali,[1] **Muhammad Golam Kibria**,[1] **Muhammad Aslam Jarwar**,[1] **Hoon Ki Lee**,[2] **and Ilyoung Chong**[1]

[1]*Department of Information and Communications Engineering, Hankuk University of Foreign Studies, Yongin-si, Republic of Korea*
[2]*Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea*

Correspondence should be addressed to Ilyoung Chong; iychong@hufs.ac.kr

Received 2 September 2017; Revised 26 November 2017; Accepted 19 December 2017; Published 28 January 2018

Academic Editor: Nathalie Mitton

The Internet of Things (IoT) is evolving with the connected objects at an unprecedented rate, bringing about enormous opportunities for the future IoT applications as well as challenges. One of the major challenges is to handle the complexity generated by the interconnection of billions of objects. However, Social Internet of Things (SIoT), emerging from the conglomeration of IoT and social networks, has realized an efficient way to facilitate the development of complex future IoT applications. Nevertheless, to fully utilize the benefits of SIoT, a platform that can provide efficient services using social relations among heterogeneous objects is highly required. The web objects enabled IoT environment promotes SIoT features by enabling virtualization using virtual objects and supporting the modularity with microservices. To realize SIoT services, this article proposes an architecture that provides a foundation for the development of lightweight microservices based on socially connected web objects. To efficiently discover web objects and reduce the complexity of service provisioning processes, a social relationship model is presented. To realize the interoperable service operations, a semantic ontology model has been developed. Finally, to evaluate the proposed design, a prototype has been implemented based on a use case scenario.

1. Introduction

The Internet of Things (IoT) envisions billions of objects connected to the Internet that continuously generate data about the physical environment. Several state-of-the-art applications can be built using the information and services provided by the pervasive and heterogeneous IoT objects. However, building applications based on these objects raises some challenges. Examples include how the growing number of objects will interact or coordinate to render valuable services, how the complexity generated by the coordination of objects can be handled, and how a large amount of diverse data sensed from the surroundings can be managed properly.

Numerous promising solutions have been provided to handle the complexity and the heterogeneity of IoT objects. One of the prominent approaches is to develop virtual environments to resolve the issues caused by the heterogeneous objects. Virtualization has become a key component in many IoT architectures, in the form of either virtual objects or virtual entities. A virtual object (VO) is defined as a digital representation of a real-world object [1] and provides a way for the IoT services to discover and compose features that cannot be achieved directly with a real-world object (RWO). It has been realized that issues such as heterogeneity of dissimilar objects and scalability can be effectively resolved using VOs. Hence, a crucial role of a VO is to bridge the gap between the physical world and the virtual world by hiding the physical characteristics of an IoT device and acquiring and interpreting its data and context.

Another way to reduce the complexity caused by a large number of objects is to link these objects in a social network [2], which provides a more exact solution than using a sole individual object. This principle has given birth to the concept of Social Internet of Things (SIoT), which is emerging as a new paradigm with the merger of social networks and IoT [3–7]. It has been realized as an efficient way to facilitate the development of complex future IoT applications. It supports sharing of information generated by

the people and the devices based on the social relationships. These relationships further promote efficient discovery of the objects and effective service compositions.

Let us have a glimpse of the application scenarios in which SIoT will make a profound effect on our daily life in the near future. Imagine a group of vehicles that form a mobile social network on the move when heading towards the same location or when they are bound in a common relationship. Vehicles may form diverse social communities; for example, small cars create a social network to share available parking space information or bicycles share information about the vacant paths using a network [8]. Vehicles commuting from home to office can join a social network to share traffic congestion experience, accident warnings, or other common information (e.g., a meeting plan) with colleagues by accessing their vehicles' social network. Let us analyze another SIoT use case where a user enters a public museum with an IoT environment, which incorporates smart objects that exploit social relationships to share information. At the entrance to the premises, the user's smartphone is prompted with a beacon signal to recommend a service app installation. When the user grants the permission, an app is installed and a social agent acquires his profile and friendship details. The app incorporates a microservice which uses a smartphone object to establish a social relationship with the museum visit service. The museum is equipped with a network of smart objects that monitor the user's location and provide services to enhance his/her experience such as personalizing the displays, assisting in the navigation, and recommending the relevant services.

To support the above-mentioned use cases, most of the current IoT architectural approaches are not suitable as they are based on the traditional monolithic approaches, which further hinder the development of efficient, modular, and independent services that cannot scale well with the increasing user's service demands, as already witnessed in many studies [9–12]. Using monolithic approaches, we cannot fully achieve the benefits offered by SIoT including efficient information discovery, improved scalability, and simplified interconnection of objects.

These limitations motivate the current paper to define a design for the development of SIoT based services with efficient and lightweight mechanisms to exploit SIoT features for improved service provisioning. The proposed design is based on the microservices concept which promises a more solid practice of SOA. There is no particular definition of microservices architectural style; however, it is explained in [13] by Martin Fowler as "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API." Following the microservices based proposed design, SIoT applications can be developed, deployed, and maintained more efficiently and independently, which will promote system modularity and interoperability.

Moreover, a SIoT design needs to facilitate the efficient information discovery based on the social relationships among objects. To achieve this, incorporating the existing social relationship models in the proposed design can be useful. However, applications with intelligent service features require dynamic object selection. Therefore, objects need to acquire the ability to infer new relationships for interconnection with other objects in the system. This view further motivates the present article to develop a social relationship model based on the semantic ontology. The utilized ontology-based approach is highly useful to dynamically infer new social relationships for efficient service provisioning.

Furthermore, to fully achieve the benefits of both, the SIoT and microservices, selecting an IoT environment is necessary where not only can the social interconnection among heterogeneous objects be utilized but also the lightweight services can be developed to apply these relationships for effective IoT service provisioning. To fulfill this goal, the Web of Objects (WoO) [14] platform has been selected. According to the ITU recommendation (ITU-T Y.4452) [15], "the Web of Objects (WoO) is a realization way of the IoT services, where virtualized objects (i.e., virtual objects (VOs) and composite virtual objects (CVOs)) are connected, controlled, and incorporated with resources to facilitate the development, deployment, and operation of IoT services on the World Wide Web."

To exploit the opportunities provided by the above discussed technological advancements, the main contributions are highlighted as follows:

(i) This article contributes a novel design based on the microservices concept with lightweight and modular services to support the development of SIoT applications. The proposed design involves mechanisms to enhance object discovery and to reduce the magnitude of complexity generated due to the huge number of objects.

(ii) A social relationship ontological model is developed, which helps identify the hidden and unidentified relationships among objects with reasoning mechanisms.

(iii) A use case with the implementation prototype based on the WoO platform is developed to exploit the social relationship model, supporting the proposed design for efficient IoT service provisioning.

Further, to support the above contributions, virtual objects, serving on real-world objects, have been developed, which can be reusable in other services as well. CVOs are implemented, where each update of the information in the real-world object is reflected in the VO and propagated to the CVO. Also, the Social IoT notion is incorporated into virtual objects to provide easy discovery and efficient execution while maintaining collaboration using social connections. Moreover, an experimental analysis has been carried out to evaluate the discovery time of VOs and the time required for service execution, with or without social relationships among virtual objects.

The remainder of this article is organized as follows. In Section 2, the related work is described. Section 3 presents the proposed design of the social web objects accompanied by the social interaction model. In Section 4, an IoT environment

use case is explained with the details of the prototype implementation and discussion of the performance evaluation. Finally, Section 5 provides a conclusion of this paper.

2. Related Work

This section presents the related work with respect to the SIoT and microservices. Further, the significance of both technologies pertaining to the IoT environment has been discussed.

2.1. Social Internet of Things (SIoT). The concept behind SIoT is to enable smart objects to create a social network similar to a human social network. The objective is to exploit the social relationships among objects in an IoT environment to facilitate the effective information discovery, to promote the scalability, to enhance the interactive communication between objects that are friends, and to achieve trustworthiness [3].

The initial notion of a social network of objects was proposed by [16], which discussed how the wireless devices form social connections on the temporary bases that are controlled by the node owners. As the work was carried out before the introduction of IoT, it did not consider the SIoT concept. Later studies [5] investigated the objects' social interaction where objects formed social networks and communicated with each other based on the rules already set.

The idea of using the Web of Things in a social ecosystem was presented by [17, 18], where things were shared using the social network infrastructures, for example, Twitter, Facebook, and LinkedIn. These implementations consisted of objects that communicated either through the built-in embedded servers or through smart gateways. Further, in these settings, the web-enabled things owned by a person provided an interface to his/her social group of friends so that they can interact with the things using a social network.

Lysis [19] is another good example of Social IoT platform based on cloud infrastructure. It uses the platform as a service (PaaS) model and focuses on the deployment of applications in the cloud space. However, dynamic inferencing of new social relationships from existing relationships is not supported by this system. Moreover, making a modular interoperable design by employing concepts like microservices and semantic web technologies is not considered in this work.

Paraimpu [20] provided a social Web of Things platform to connect virtual and physical things to the web. In this system, the social concept only relates to humans by providing them with the capability to share things with each other using a human social network. Also, virtual things are not considered as VOs but rather viewed as services on other IoT platforms.

Using SIoT at the edge of the network was investigated by [21]. In this system, an approach was proposed to cope with the communication delay due to the objects being remotely located in the cloud. This approach exploits the computing resources at the network edge to deploy virtual objects.

The ThingSpeak solution was used as the basis for the implementation of an early SIoT platform [22]. This platform provided the object social behavior functions incorporated at the centralized server. In this implementation, functions such as the creation and the management of the social relationships were developed. However, VO concept was limited only to the records in the remote database.

Another featured SIoT framework was contributed by [4]. This work presented how to combine services, devices, users, and their social interaction to enable interactivity, discovery, and recommendation of services. Social relationships have also been investigated in communities [23], where a proposed framework enables the identification of communities in the social networks.

Although most of the research contributions in the SIoT domain incorporate a notion of the social association of objects, however, limited information has been provided about the social relationship model and the details on how that can be used by IoT services. Furthermore, existing approaches are lacking lightweight and modular mechanisms which can be utilized by services to exploit social relationships efficiently. In our approach, we have developed a semantic ontology-based social relationship model that provides the capability to infer new connections in a network of social objects. Another distinction of the proposed system is a microservices based design that utilizes the developed social model to provide lightweight, modular, simple, and interoperable IoT services.

2.2. Microservices. The microservices architecture is emerging as a new trend among the practices of developing distributed web applications. To develop SOA based systems, microservices have become a prominent approach in real-world settings [24]. In the microservices based architectural pattern, each application incorporates a collection of small services which execute independently and use lightweight communication mechanisms [9]. The microservices are targeted for doing one thing well at a time based on the principle of single responsibility [25], as defined by Robert C. Martine: "Gather together those things that change for the same reason and separate those things that change for different reasons" [26]. Also, the microservices are conceived as autonomous entities, which means they can be changed and deployed independently of each other without requiring their consumers to change [24].

In the last few years, several IoT architectural designs were proposed and developed. Some of these designs provided innovative features such as the European FP7 project iCore [27] which proposed a cognitive framework for IoT application development. This project provided an architecture with the concept of virtual objects and their composites. However, in iCore, services are designed using a traditional SOA based monolithic style, whereas the microservices based architecture provides a better option to develop IoT services. Furthermore, in iCore, VOs lack the mechanisms to semantically represent real-world objects (RWOs), which limits extensibility and interoperability.

To support IoT applications, microservices based architectures are being proposed in many research initiatives and applied in several research projects. In [28], microservices are used for M2M applications, realizing the fact that monolithic approaches for M2M cannot provide a real solution. Another

work [10] provided an analysis of using microservices architecture for large-scale distributed applications. This work demonstrated that significant benefits can be achieved as compared to generic SOA approaches when using microservices in an IoT platform for smart city applications.

Moreover, Almanac FP7 EU Project [29] provides smart city services based on the IoT environment. The project uses microservices to employ scalability in a horizontal and vertical fashion. Also, in the industry, microservices have become a solution for developing large-scale applications. Netflix, Pivotal [30], and Amazon use microservices in their software bases.

The increased importance of microservices for the IoT applications is due to the fact that they simplify complex systems. By dividing a system into smaller parts, higher cohesion and lower coupling can be achieved, which makes it highly scalable. In scenarios where requirements keep changing continuously, microservices can help make a system easily modifiable. Moreover, the major benefits that are envisaged when using microservices in an IoT environment are as follows: Microservices leverage scalability to provide a highly decoupled pattern and can allow scaling individually. Their usefulness can be realized as when the demand for the requirement increases the system complexity also rises. In this case, a system can be supported by creating new instances of existing services. Microservices scalability [9] fits in three-dimensional space, that is, horizontal scalability (typical scalability), vertical scalability (splitting different individual microservices), and z-axis scalability (splitting similar things, such as DB partitioning). Microservices architecture supports a plug-and-play behavior where system components become loosely coupled. To acquire new functionality or replace a failed service, pull the plug from one microservice and plug into a new microservice [31]. One of the major problems that IoT is facing today is how to deal with the heterogeneity of incompatible solutions. To harmonize the heterogeneity, semantic web technologies with microservices are leveraged to provide interoperable exchange and communication of data. Microservices realize a decentralized and autonomous behavior, operating on their own priorities and schedule. This way, they provide several benefits for being utilized in an IoT design.

3. The Proposed Design of Social Web Objects

Recently, there have been several research efforts [3–6, 17, 20, 32] for defining the Social IoT. This new aspect minimizes the complexity generated by the communication of billions of real-world objects and enables sharing of information. The social notion can be incorporated into the WoO platform to provide a semantically rich base for IoT applications that can utilize real-world object relationships for efficient information discovery. Moreover, this section provides a brief introduction of WoO reference architecture in Section 3.1. A classification of social relationships among web objects (virtual objects and composite virtual objects) is discussed in Section 3.2 and a social relationship model is presented in Section 3.3. Further, the details of functional components for the proposed architecture are explained in Section 3.4.

3.1. Web of Objects (WoO) Reference Architecture. To enable the deployment of IoT services on the World Wide Web, WoO provides a reference framework. In other words, it realizes the IoT services in such a way that virtualized objects are interwoven with resources to support the development, deployment, and operations of IoT services [33, 34]. The WoO platform provides service functionality by merging VOs and features of web applications. Moreover, the WoO platform uses semantic web technologies to enable interoperability among heterogeneous resources. This realization provides the basis for the composition and harmonization of objects to provide smart services in an IoT domain. Building services based on WoO platform using diverse technologies, including microservices [35], social networks, and semantic web, helps reduce the complexity and fosters the easy and efficient development of IoT applications.

Virtualization has become a major concept in IoT to address the heterogeneity of diverse types of physical objects. VO is a digital representation of a real-world object that can be anything, living or nonliving, mobile or stationary, concrete or immaterial. In WoO, VO is defined as domain-specific semantic ontology based on the VO information model [36]. It is uniquely identified using URI and provides information updates on the representative real-world object. On the other hand, WoO also provides a notion of CVO which aggregates one or more VOs to enable service features that satisfy the application requirements. CVOs chain semantically interoperable VOs together. The WoO platform incorporates features to efficiently reuse existing VOs and handles the complexity generated by self-management and control mechanisms [37]. In the WoO, service level decides which CVOs and VOs will take part in creating an object mashup to fulfill service objectives. The service logic is saved in the form of templates and stored in the template repository [38–40]. The domain expert or knowledge engineer defines the service templates that are used to instantiate new services.

WoO reference architecture is shown in Figure 1. In this architecture, the service layer handles requests and provides several management functions, whereas the CVO level consists of functions to manage and instantiate CVOs or reuse the existing ones based on the provided service execution logic. New objects such as sensors are registered in the WoO platform using a registration function and their templates are generated to make them digitally available in the form of VOs. Another most important aspect is the semantic representation of data at three-layered WoO architecture [23, 39]. The database at each layer is supported with semantic web technologies.

3.2. Types of Social Relationships. Previous research [3] on SIoT has derived some basic relationship types among objects. These relationships are characterized as follows. *Parental Object relationship* (POR) exists between the objects that belong to the same batch, such as objects that are created in the same production process or at the same time by a common manufacturer. *Cowork object relationship* (CWOR) is the relationship among objects that are grouped together based on some commonly shared job to be done by them. Similarly, *colocation object relationship* (CLOR) is established

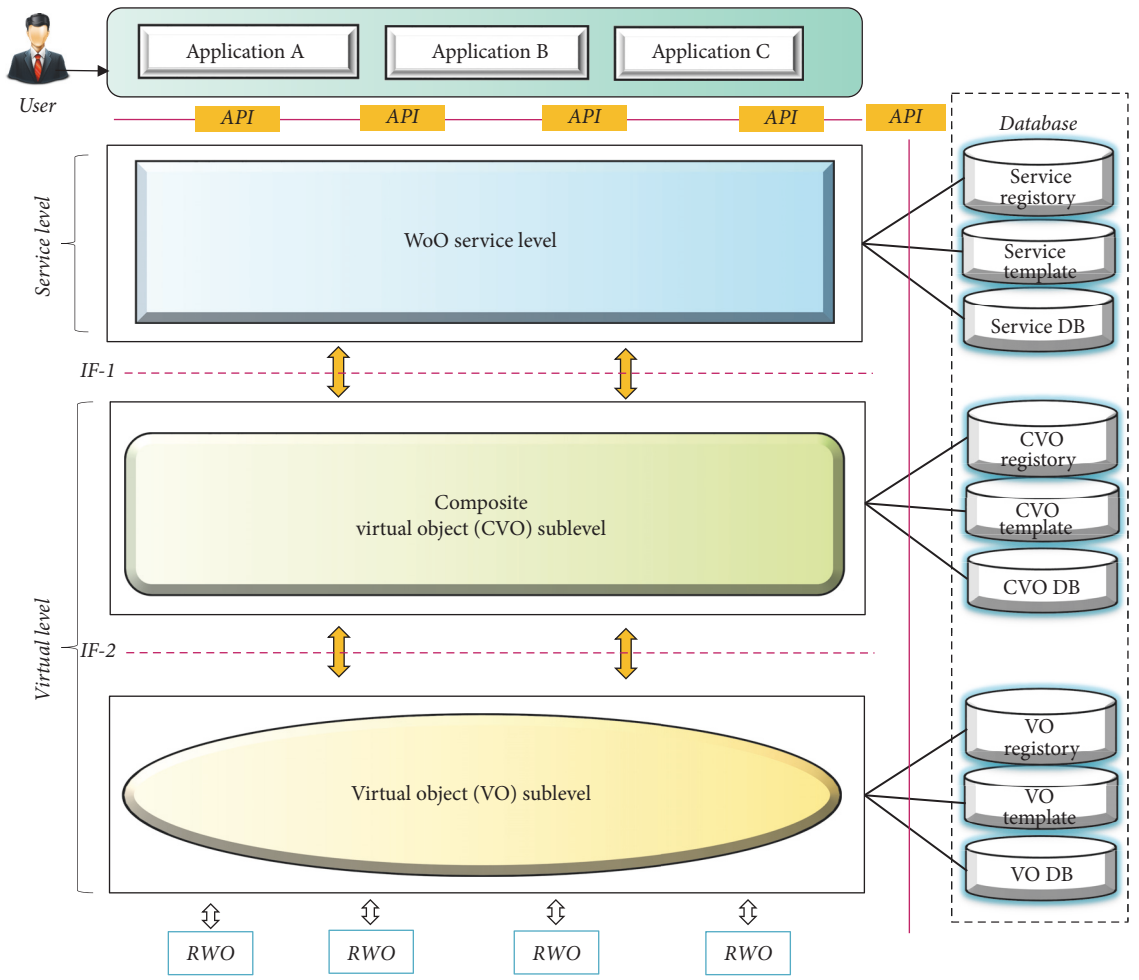

Figure 1: Web of Objects reference architecture.

between objects belonging to the same location such as home or office. Different or same types of objects can be combined based on the common location or premises. On the other hand, *Ownership Object Relationship* (OOR) is formed among objects belonging to the same owner. *Social Object Relationship* (SOR) is another kind of relationship that is created among objects when they come close to each other, either random in time or periodically. This relationship can be envisaged when the owners of objects come into contact with each other. Apart from the above relationships, in [41], the authors define the *Guardian Object Relationship* (GOR) in the Internet of Vehicles (IoV) scenario where on-board units of vehicles become children in relation to the super nodes of Road Side Units. This child and parent association gives a special meaning to a new hierarchical relationship.

In addition to the relationships defined in other studies, we have identified few more relationships suitable for some diverse scenarios. These include *Sibling Object Relationship* (SIBOR) that is created among objects that belong to a group of friends or family members. This relationship among objects extracts the property of trustworthiness based on the relationship of their owners. Another relationship type is the *guest object relationship* (GSTOR) which is formed between objects that belong to the users in the guest role, for example, when a person visits a friend's home and gets the privilege as a guest. The same can be applied to the objects that move in guest relation (GSTOR) from one place to another; they will have some privilege of accessibility of information as compared to other objects. *Stranger Object Relation* (STGOR) applies to objects that encounter the presence of each other in an anonymous environment such as on the go or in the public environment. Similar to people who meet each other sometimes regularly but are anonymous to each other or are not fully aware of each other, this relationship can be used among objects to form different trust levels or to form strict restrictions to promote secure connections. Moreover, in *service object relationship* (SVOR), objects form a relationship while coordinating in the same service composition to fulfill a service request. Generally, in the proposed scheme, service request execution generates a mashup of objects, and objects that belong to the same mashup are assumed to be in SVOR.

We believe that the above defined new social relationships are beneficial in several distinctive scenarios. For example, SIBOR is more useful in scenarios where trust is highly important for establishing a connection. Unlike the more generalized relation POR, SIBOR is established

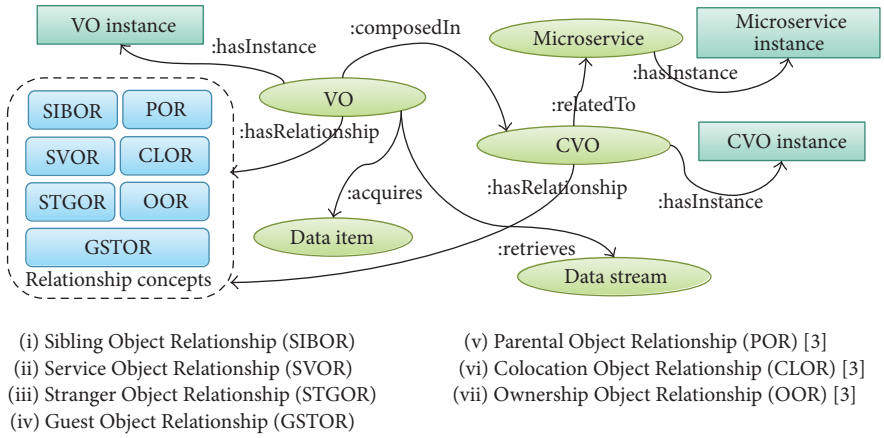

Figure 2: Ontological model representing web objects' relationships.

among objects from the same manufacturer having similar features but different behaviors. Similarly, GSTOR is suitable in scenarios where objects use relationships for privileged services. Imagine a person as a guest speaker in a conference with his smart devices, where he can connect to the network easily, and his personal devices are recognized as a guest (with GSTOR) in the venue network to avail free services like high-speed Internet, special notifications, and others. Stranger Object Relation (STGOR) is used in scenarios where objects are on the go, and to get some information, they have to compromise with a limited trust level. For instance, in a public transit system, one mobile node on the move wants to use crowdsourced information from another mobile node which is not fully trusted. Also, SVOR is useful in service composition specific scenarios, where object mashups are formed to facilitate service requirement, and it is also useful in a particular orchestration of objects.

To model information about microservices, social web objects, and their relationships, a semantic ontological model has been developed as depicted in Figure 2. Conceptualization of objects and their social relationships in semantic structures are highly beneficial to infer new connections among a network of objects with the help of reasoning techniques. The ontology facilitates the representation of objects, as well as what they measure in terms of observations, their processing, and functions. Web objects' relationship ontology contains the concepts that define each object in terms of a class. The data properties are used to represent object values and the object properties are defined to identify the link between two objects in the ontology. Major concepts in the ontology are microservices, CVOs, VOs, and their social relationships. Further, microservices have an individual instance that uniquely defines a microservice object and associated data with it. Similarly, CVOs and VOs are also instantiated and contain data properties to retain their values. With the increasing number of objects, a semantic ontology model is used to perform reasoning function that helps identify new connections. Further, to analyze the social relationship among objects (such as VOs and CVOs), their properties are provided in Table 1.

3.3. Social Relationship Model for Web Objects. At each level of service life cycle, objects form a social relationship with each other to accomplish a service task. Codifying the relationship among web objects can provide many benefits, such as efficient information discovery from related objects and better composition and reuse. To perceive social interaction between web objects, we assume every service is based on one or more microservices and other objects including CVOs and VOs. At VO level, VOs create several relationships with each other based on the RWO they represent. CVOs associated with microservices are incorporated in relationships as well.

Object-to-object associations are categorized as either vertical associations (interobject relations) or horizontal associations (intraobject relations) as shown in Figure 3. The relationships that flow from the bottom to the top in WoO based architecture are known as interobject relations (i.e., from VO to CVO or vice versa). On the other hand, relationships are known as intraobject relations if they are generated within objects at each level such as between one VO and another, or within one CVO to a relative CVO. The newly defined relationships among objects are maintained at different levels, which enables information discovery more efficiently. It is considered that not only will the relationships be maintained when two objects (i.e., VOs) are used in the same CVO or service, but also they will be maintained if objects are used in different CVOs or services.

As depicted in Figure 3, VO1 and VO2 bind in the SIBOR relationship, considering the fact that they belong to the same family of CVO1. Similarly, VO4, VO5, and VO6 are combined in a CWOR, as they are used in the same service. However, all of them do not have a common family relation that makes their trustworthiness restrictive to some level as compared to SIBOR. Moreover, though VO7 and VO8 are not used together in a service setup, still they form a STGOR relationship. This relationship also helps codify those VOs that have similarity in premises during a certain point in time regularly but they had never combined in any service scenario before. For example, if a person visits a subway regularly but stays very shortly, in such scene, the user's smartphone VO and subway station point

Table 1: Social relationship attributes.

Social object relationships types	Attributes/properties
Sibling object relation (SIBOR)	O_x *has relationship* with $O_y \rightarrow$ SIBOR, if objects' *ownership* is defined as: == SameFamily
Service object relationship (SVOR)	O_x *has relationship* with $O_y \rightarrow$ SVOR, if objects' *serviceStatus* is defined as: == SameServiceComposition
Stranger object relation (STGOR)	O_x *has relationship* with $O_y \rightarrow$ STGOR, if objects' *authentication* is defined as: == Anonymous
Guest object relationship (GSTOR)	O_x *has relationship* with $O_y \rightarrow$ GSTOR, if objects' *ownership* is defined as: !=SameFamily && == GuestFriend
Parental object relationship (POR)	O_x *has relationship* with $O_y \rightarrow$ POR, if objects' *creation* is defined as: == SameBatch ‖ == SameProduction
Colocation object relationship (CLOR)	O_x *has relationship* with $O_y \rightarrow$ CLOR, if objects' *proximity* is defined as: == Neighborhood ‖ *location* is defined as: == SameLocation
Ownership object relationship (OOR)	O_x *has relationship* with $O_y \rightarrow$ OOR, if objects' *proprietorship* is defined as: == SameOwner

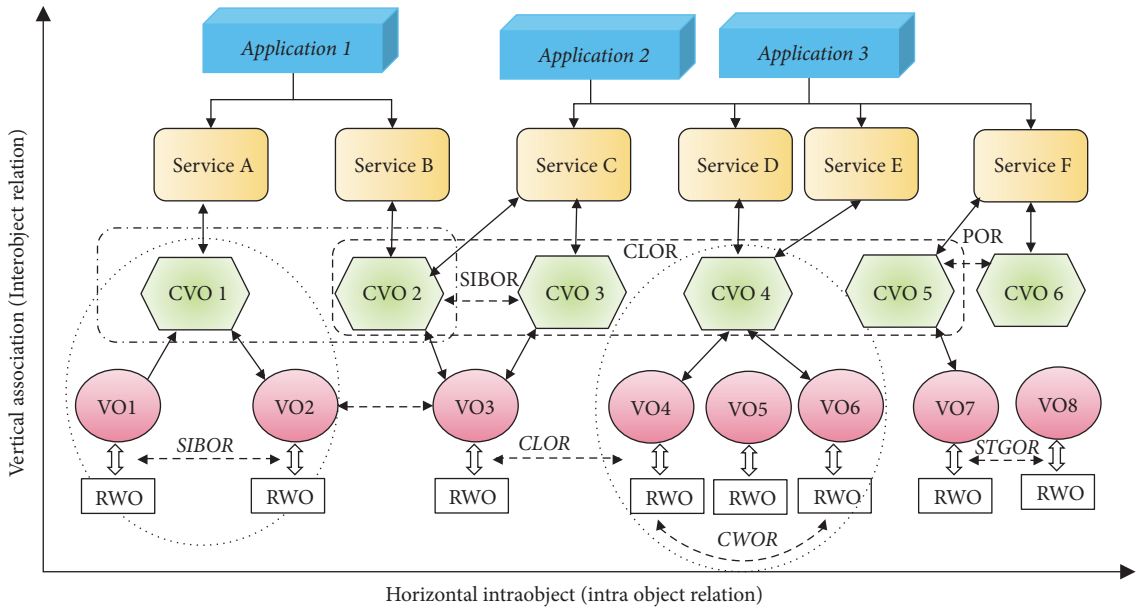

Figure 3: Hierarchical social relationships of web objects.

VO or subway security camera VO may establish STGOR. This may help reuse VOs in service scenarios where only for short duration a VO object is required. At CVO level, CVO2 forms two different relationships, which increases the level of connectivity of the CVO. CVO2 first shares a CLOR with CVO1 and CVO3, CVO4, and CVO5 together. Secondly, CVO2 is combined in a SIBOR with CVO3 and becomes in the same service family. Further, in the selected use case scenario described in Section 4, several other social interactions have also been elaborated.

3.4. Social Web Objects Architecture. SIoT envisages a system where the social framework will bring smart devices and people to interact with each other. By incorporating web technologies like SOA based microservices, IoT services can be rendered on top of the social framework. The proposed social web objects architecture (as shown in Figure 4) has been decomposed into three levels: service level, object virtualization level, and aggregated object virtualization level. At the service level, to support social relationships within web objects, several microservices have been designed, which are discussed further in the following section. Moreover, the data management function in the proposed design helps each layer to interconnect with semantic databases. The SPARQL endpoints have been defined at the service and the object virtualization levels that expose interfaces to retrieve, store, and modify RDF graphs. Several interfaces allow the knowledge engineer, domain expert, and the developer to create service templates and VOs, as well as update the RWK model, user profile, and policies.

3.4.1. Service Level Functional Components. The service level functions handle operations from the request inception to the execution of services. This level is supported by some core functions common to be used in each service life cycle. These include the management function to handle service

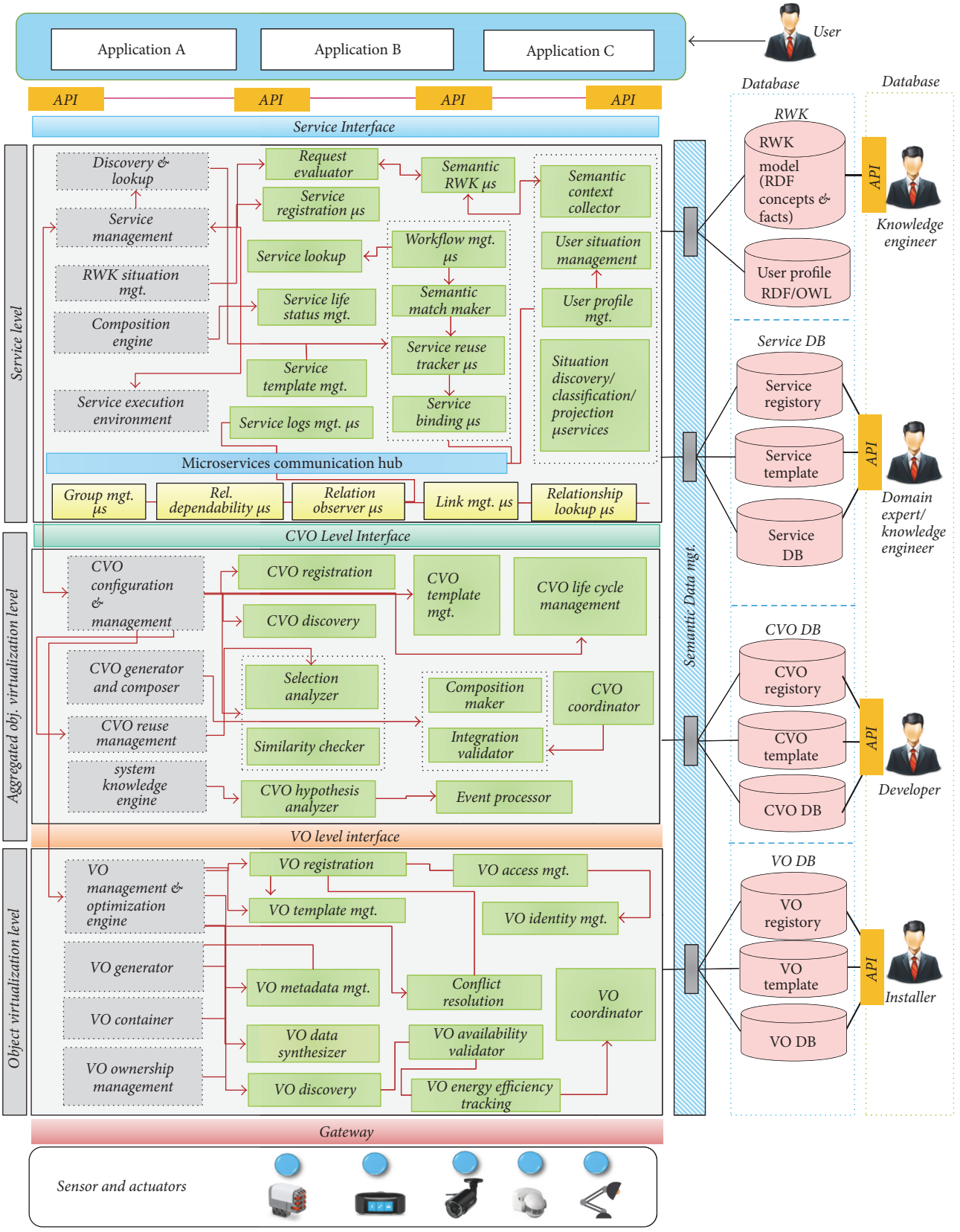

Figure 4: Social web objects functional architecture.

requests and initiate service provisioning processes, the service discovery function to select the available services, the composition engine to enable service mashups, and the Real-World Knowledge (RWK) and situation management function to acquire RWK and continuously maintain it. Besides these core functions, several microservices have been used to provide the plug-and-play feature in the proposed design; that is, in case of service failure, it is replaced with another one. These services include the Request Evaluator microservice which interprets a service request and matches a request query to the service template. The registration microservice registers newly added service objects to the registry. When a new template is instantiated, objects are put in service, and to know the existence of available service instances, they are recorded in the service registry. The lookup microservice is defined to search for the required service objects using the semantic representation. The representations available in RDF/OWL format are queried from the registry based on the request parameters. Service life cycle management keeps track of the states of service objects during their life cycle, whereas RWK management microservice is a semantic data processing service which processes facts about user preferences, profile, and situational information.

To realize service provisioning, it is always not possible to satisfy the service requirement with a single service function. Service composition and harmonization features are required to fulfill a demanding service request. Service composition is not a single operation; many microservices are used to compose the individual features. One of the major components of the composition is the Workflow Management (WFM) microservice which is responsible for decomposing the service requirement if no single service is available to satisfy the task. WFM microservice identifies the service objects using input and output interfaces. In this concern, it uses the matchmaking microservice to identify the approximate matches that can serve the required service functionality. Here, service objects are semantically represented in the form of ontologies. The semantic ontology alignment algorithm is used to match the service up to a certain threshold level that is defined by the matchmaking microservice. The reuse tracker microservice is used to identify the current service instance usage by the application; it incorporates a coordination mechanism before instantiating any new instance, and this helps in service reuse by many clients. Binding microservice enables tying of services selected by WFM; this function synchronizes input and output of services in a workflow to produce a single service output based on the generated composition plan.

Furthermore, user characterization is another important element to support the IoT services by matching user requirements more closely. The User Profile Manager microservice handles many facts related to the human user. These include user profile, preference, context, and policies. Semantic Context Collector microservice organizes the services requested by the user and the context in which these services were requested. This helps in identifying user interest with respect to the service context. User context or situation helps in service selection more efficiently if a user requests again with a similar service requirement. Some microservices have been designed to handle user situation information; these microservices include situation discovery, classification, and recognition functions. Situation discovery is used to process sensor data that is provided by CVOs to prompt a situation, whereas machine learning algorithms are applied to remove the false values and data normalization is performed in case inappropriate triggers in the data exist. Situation classification incorporates the reasoning methods and rules to generate a relationship over detected events and refine the missing knowledge between the events. Situation projection analyzes the events data with the help of machine learning methods and provides an output in the form of predicted facts that are related to the situation being detected. Moreover, all the microservices coordinate using a communication hub to share information with each other.

A flow of service inception to service execution in the proposed architecture is depicted in Figure 5 where service management is responsible for initiating the above-mentioned functionalities and instantiating the microservices. This process further results in the generation of CVO/VO mashup graphs to be handled by CVO management (discussed in Section 3.4.2). Situational information has also been used to improve request analysis and service provisioning, whereas VO management (discussed in Section 3.4.3) acquires and aggregates data from sensors and other RWOs.

Social Relationship Management Microservices. To support the social relationships among web objects at each layer of the proposed system, relationship management microservices are defined, which support the codification of relationships among web objects (VOs and CVOs) and their management. These microservices are defined as follows. Group management microservice identifies the grouping of objects into specific sets based on the type and interaction of objects; it builds a social graph of the object relationships. Link management microservice incorporates mechanisms to maintain relationships among different objects and it includes several other subfunctions. The first subfunction is object selection which involves choosing candidate objects that are likely to form a particular association. The second subfunction is matchmaking service which provides exact relationship match based on the type of objects. The third subfunction is the association life cycle manager, which maintains the relationship status between objects; it checks whether they are in active relation or not. It keeps track of relationship validity and also the duration of relationships. Relationship observer microservice enables the object activities to be observed, and it also handles the mutual sharing, which defines what information an object is allowed to share or use of another object based on the relationship among them. Relationship dependability recognizer microservice identifies, maintains, and continuously improves the level of reliability among objects. If objects A and B are frequently interacting and always perform a task together that leads to successful execution of an application to serve the requirements, then they are more likely to form a reliable relationship with each other. The consistency of the relationship allows objects to share their information with fewer constraints as compared to unpredictable association among objects. Relationship

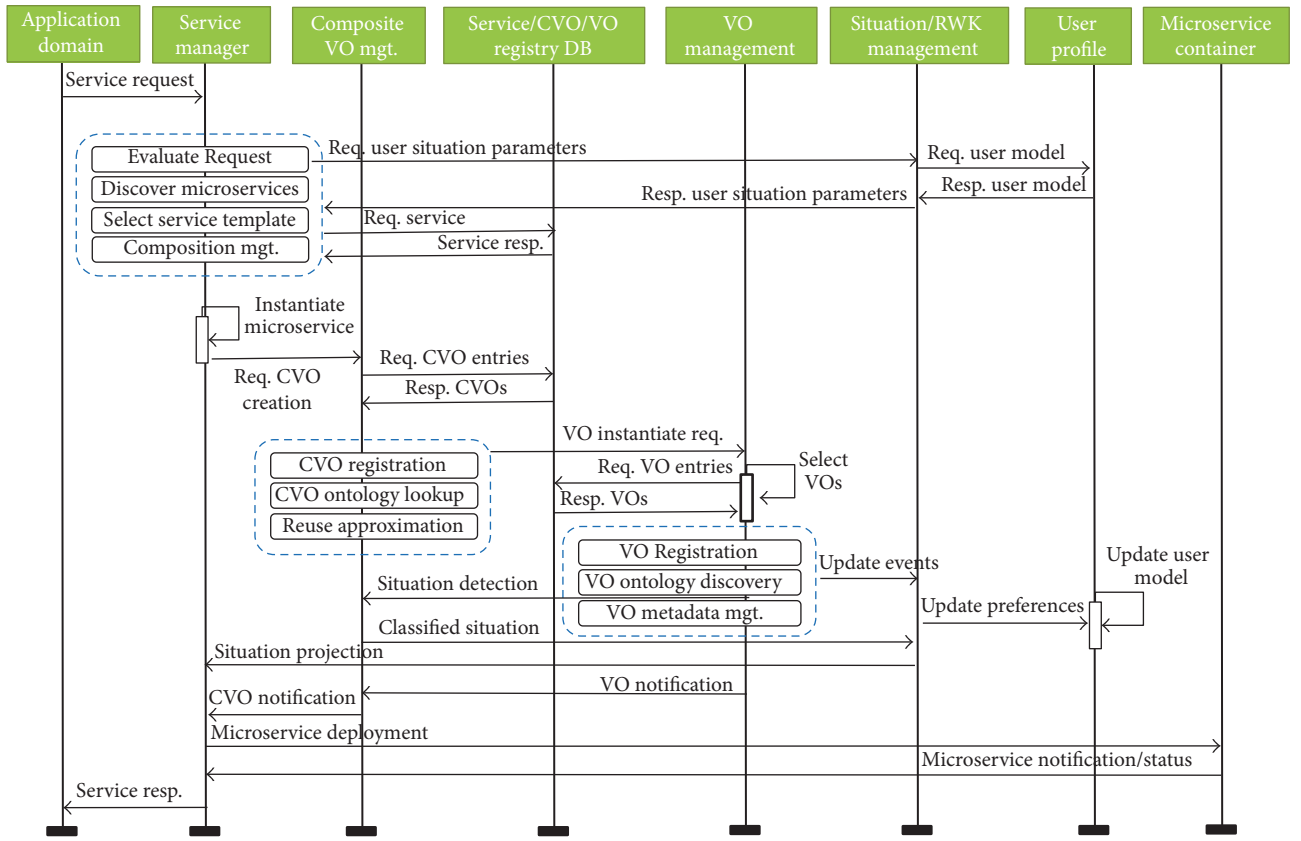

Figure 5: Request initiation and service execution flow in the proposed architecture.

lookup microservice incorporates the discovery algorithm (as given in listing of Algorithm 1) to discover the social relationships and maintains the list of all objects and their potential association rating. The rating defines the opportunity for objects to form a relationship with each other.

Moreover, the operation flow of social microservices has been illustrated in Figure 6, where VO management retrieves the social relationship among objects this is queried using link management, which uses a lookup service. Associations are maintained in a social relationship graph using semantic ontology and queried using the relation observer service. The social graph is also updated and maintained continuously with respect to changing links among objects, on the basis of which VOs are selected.

3.4.2. Functional Components in Aggregated Object Virtualization Level. Aggregated object virtualization (AOV) level involves all the necessary functions to instantiate, manage, and continuously monitor CVOs. Besides, it includes the interface required by the service level to interact with CVO components. Service level provides the mashup graphs of CVOs and VOs that are required to fulfill a service request. Aggregation level also supports some core functions; these are categorized as CVO configuration and management to decompose requests from service level and manage CVO instances, CVO generator to instantiate CVOs, composition function to form CVO and VO mashups, reuse management to select existing CVOs that approximately match service requirement, and system knowledge engine to grow knowledge for the optimal use of CVO resources. Further, other aggregation level functionalities are as follows: registration function records the CVO entry into the semantic CVO registry in the form of RDF/JSON format. CVO includes ontologies that specify conditions to be applied on the VO data. Also, at this level, templates are created by a domain expert or knowledge engineer, which consist of functionalities associated with each CVO type. Template management module provides an interface to include templates to the CVO template repository and also supports the modification mechanism. CVOs are discovered using their semantic annotation via the discovery function. CVO selection analyzer function provides estimated selection of a CVO for reuse if there is no single CVO that can exactly provide the required service. This is supported by similarity checking mechanism to analyze the level of similarity before recommending for reuse. At aggregation level, another important function is event processing; event streams generated by object virtualization level are highly valuable to infer situational awareness and building RWK that is used by high-level functions. Event processing module facilitates processing of events and provides generated facts as output. Event processing is supported by CVO hypothesis analyzer that incorporates several hypotheses which are trained over data to build system level knowledge. Aggregation level also

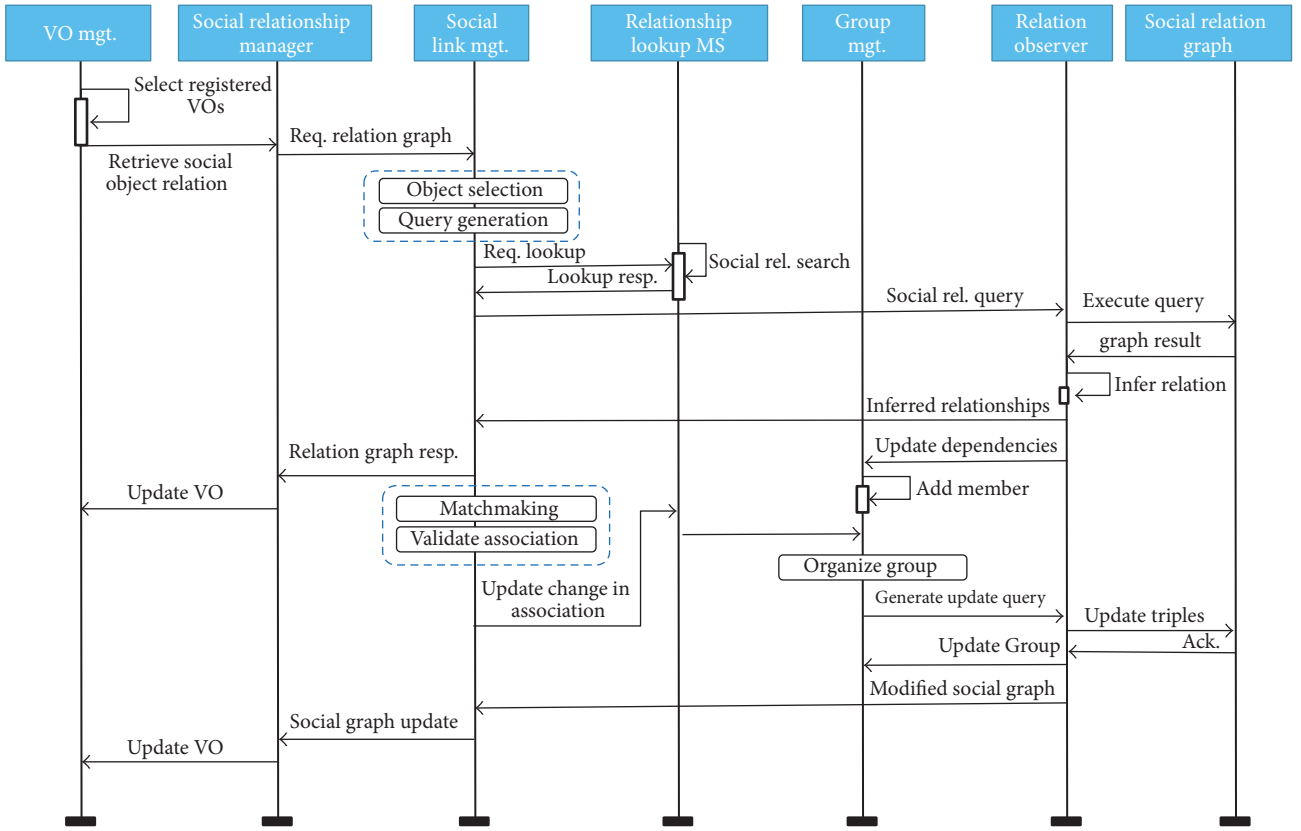

Figure 6: Sequence of operation required to establish a social relationship among objects.

provides composition function which provides the basis for merging multiple VOs to generate single service feature and validation function to validate the compositions of multiple VOs. Moreover, the coordinator helps in resolving conflicts in case of simultaneous access of single CVO by multiple services together.

3.4.3. Functional Components in Object Virtualization Level. Although IoT middleware virtualizes sensors and actuators in some form of virtual entities, however, in WoO, every living or nonliving thing can be represented in the virtual world. VOs are supported by an information model [14, 15] which includes the representation of RWO that can be ICT or non-ICT. In the proposed architecture, object virtualization level includes functions to create, maintain, and coordinate VOs such as VO generator, VO management, and optimization engine. VO registration function provides an interface to register VOs with the help of VO ownership management function where the VO ownership is maintained. Template management function handles templates or loads updated ones as provided by the domain expert. Metadata management maintains data associated with VOs; as VO is a digital representative of real-world entities, there is a high need to store metadata information to recognize the class of sensed data. Use of metadata provides the real value of the data in the form of the context stored within it. Furthermore, VO data synthesizer service handles the discrepancies if found in the data corresponding to VO; this includes data manipulation and missing value rectifications. Conflict resolution function helps in case the same VO is used by multiple service instances, to resolve the conflicts. Also, VO access management function controls access for VOs in case access right information has been defined by the VO owner, whereas identification management module maintains identification information about VOs so that each VO can be uniquely identified in the system.

3.4.4. Social Relationship Discovery and Composition. To identify associations among web objects within a social relationship graph, an algorithm is elaborated briefly. In the following algorithm, the microservices that are in execution are retrieved first, represented as $\mathbb{\Sigma}M\mu$; next, all the CVOs associated with each of those microservices are recovered in $\mathbb{\Sigma}$Mc and, further, the corresponding VO model $\mathbb{\Sigma}$Mv is iterated. In the next step, executing the query statement routine returns the results denoted as $\overline{\mathbb{T}}$. The process of finding the relationships is carried out through lines (6)–(19) in multiple nested iterations as shown in the pseudocode of Algorithm 1 listing.

Algorithm 2 facilitates the composition of objects based on social association types. This algorithm takes as input the set of object social relationships denoted by $\mathbb{Z}$ and registry entries of the objects that are currently available in the system, represented as $\mathbb{R}$e. After the object model denoted

Require: ($\mathbb{\Sigma}\phi$), ($\mathbb{R}_e$)
(1) **Output:** $\hat{\mathbf{S}}$ (social relationship graph of objects)
(2) $\mathbb{\Sigma}\mathrm{M}_{\underline{\mu}} \leftarrow$ Load Microservices in model
(3) $\mathbb{\Sigma}\mathrm{M}_{\underline{c}} \leftarrow$ Load CVOs in model
(4) $\mathbb{\Sigma}\mathrm{M}_v \leftarrow$ Load VOs in model
(5) $\overline{\mathbb{\tau}} \leftarrow$ *executeQueryStatement* (M_μ, M_c, M_v, Q)
(6) **for All** μ in $\underline{\mu}\overline{\mathbb{\tau}}$ **do** (iterate Microservice instance being used)
(7) **for All** $O \overline{\in} \mu$ **do** (check CVO and VOs used by the Services)
(8) **if** O_i == output of C_R in a relationship set R_c **then**
(9) add C_R to $\mathbb{\Pi}_L$
(10) **else**
(11) **if** O_i == output of V_R in relationship set R_v **then**
(12) add V_R to $\mathbb{\Pi}_L$
(13) $\mathbb{N}_i \leftarrow$ tag (Assign relationship tag to each object entry)
(14) **else**
(15) add O_i to $\mathbb{\Pi}_U$
(16) **end if**
(17) **end if**
(18) **end for**
(19) **end for**

Algorithm 1: Social relationship discovery.

Require: ($\mathbb{\Sigma}\phi$), ($\mathbb{R}_e$), ($\mathbb{Z}$)
(1) **Output:** $\mathbb{C}$ **(Composite Service)**
(2) $\mathbb{\Sigma}\mathrm{M} \leftarrow$ List all objects available (Microservices, CVOs, VOs) in model
(3) $\overline{\mathbb{\tau}} \leftarrow$ *executeQueryStatement* ($\mathbb{\Sigma}\mathrm{M}$, Q)
(4) $\mathbb{Z} \leftarrow$ *executeQueryStatement* ($\mathbb{\Sigma}\mathrm{M}$, Q')
(5) **for All** $\sigma i \in \overline{\mathbb{\tau}}$ **do**
(6) **for All** $\boldsymbol{\lambda} i \in \sigma$ **do**
(7) **if** $\boldsymbol{\lambda} i$ == *any* z in $\mathbb{Z}$ **then**
(8) add $\boldsymbol{\lambda} i$ to $\mathbb{\Pi}_{\mathrm{Lm}}$
(9) store $\mathbb{\Pi}_{\mathrm{Lm}}$
(10) **end if**
(11) **end for**
(12) **for All** $\boldsymbol{\varpi}\mathbf{i} \in \mathbb{\Pi}_{\mathrm{Lm}}$ **do**
(13) **if** $\boldsymbol{\varpi}\mathbf{i}$ == *any* $\boldsymbol{\varpi}\mathbf{i}$ in $\mathbb{\Pi}_{\mathrm{Lm}}$ **then**
(14) $\partial \leftarrow$ ranking (Assign rank to each service)
(15) add $\boldsymbol{\varpi}\mathbf{i}$ **to** $\mathbb{\Pi}_{\mathrm{LR}}$
(16) store $\mathbb{\Pi}_{\mathrm{LR}}$
(17) **end if**
(18) **end for**
(19) **for All** $\sigma j \in \mathbb{\Pi}_{\mathrm{LR}}$ **do**
(20) **if** $\sigma j(\mathrm{R}\partial) > \kappa$ **then**
(21) add σj to Wf
(22) Pq $\leftarrow$ assign to queue
(23) Sort Pq
(24) Formulate Wf
(25) **end if**
(26) **end for**

Algorithm 2: Composition of objects based on social relationship.

as $\mathbb{\Sigma}\mathrm{M}$ is loaded, the first step is to compose objects based on their relationship types. This way, the SPARQL queries are executed; the first query is denoted as Q to retrieve the available service templates and the second query is represented as Q' to extract the relationships associated with the services. From line (5) to (11), first, the social relationship is searched among objects and then the found relative objects are composed in groups. The design philosophy employed over here is to assign a ranking to each selected group based on the relationship type that helps in a more efficient

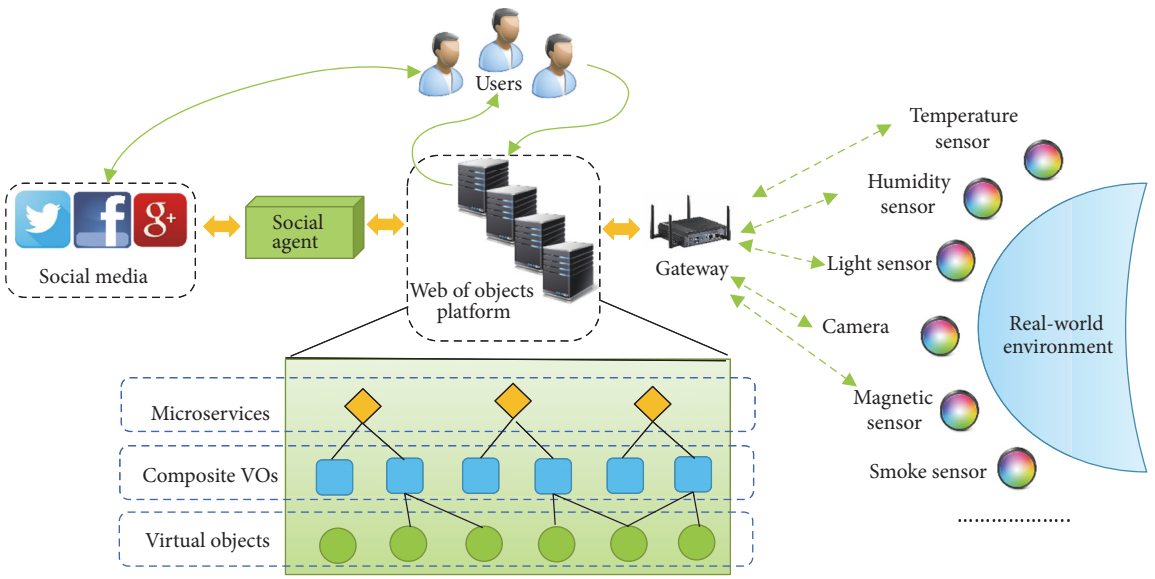

Figure 7: Use case scenario.

composition of service objects (from line (12) to (18)). The last part of Algorithm 2 will continue to iterate to compose the object composition workflow by checking first the assigned relationship ranking with the defined threshold, and then the selected composition workflow graphs are sorted and stored in queues for service execution.

4. Use Case and Prototype Implementation

In this section, first, to realize IoT service provisioning based on social web objects, a use case about IoT enabled museum environment has been presented and then the details on the prototype implementation are furnished.

4.1. Use Case Scenario. This use case is based on a user experience in a public museum where the IoT environment is already set up with the WoO platform incorporating web objects that use social relationship model and a social agent which obtains the user's profile and his friendship details from social media (as shown in Figure 7). The user enters the premises of the museum and his smartphone is prompted with a beacon signal to recommend service app installation where he accepts the request. The phone app incorporates a microservice which establishes a social relationship (GSTOR) with museum visit service objects. The system monitors the interaction of the user in the museum environment; the installed camera network and motion detectors detect the user presence and this feed is collected by the museum visit service. The user shares his current location with the system, which is used by the location navigator service to assist him in moving around the museum by providing the direction to the locations where different museum items are situated. It is worth noticing that the VOs for museum visit service and location navigator service are bound in CLOR relationship due to the same location points. The user shares his social network profile which is used to acquire the list of friends and choices for things the user is interested in. This information is acquired by the recommendation service which suggests the best things to do with respect to the user's interests, making his visit most enjoyable. Recommendation service suggests facilities available in the museum such as places a user should visit, cafeteria food menu, and prices for the day. This service also shares with the user the past experiences of his friends at the museum to help him get most out of his visit. Moreover, personalization service provides special features such as adjusting contents on displays or optimizing the luminosity, to support users with disabilities. To enable these service features, objects exploit the SVOR relationship. Also, the fire management service maintains the status of all VOs that can detect an unusual fire situation in the museum. The temperature VOs acquire readings from temperature sensors and smoke detector VOs get a feed from smoke detection sensors. These VOs form a CLOR relationship with each other in the common proximity. Fire situation monitoring CVO acquires VOs data. One or more CVOs are managed by the fire management service. In case of a fire breakout, the fire management service executes with the data from the fire detection CVOs and the temperature and smoke detector VOs. Meanwhile, services incorporate high-level functionality to react on the data provided by virtual entities in the system. It is important to notice that sharing of information based on social relationships has become easy in the proposed use case. That is because the user information and device information are communicated with social links of web objects and are utilized by the services.

4.2. Proof-of-Concept Details. To evaluate the proposed design, a proof of concept has been furnished, which realizes services based on web objects as considered in the above use case. These objects include, for instance, fire situation monitoring CVO to handle emergency fire breakout situations, location navigator CVO to find the shortest path to the user

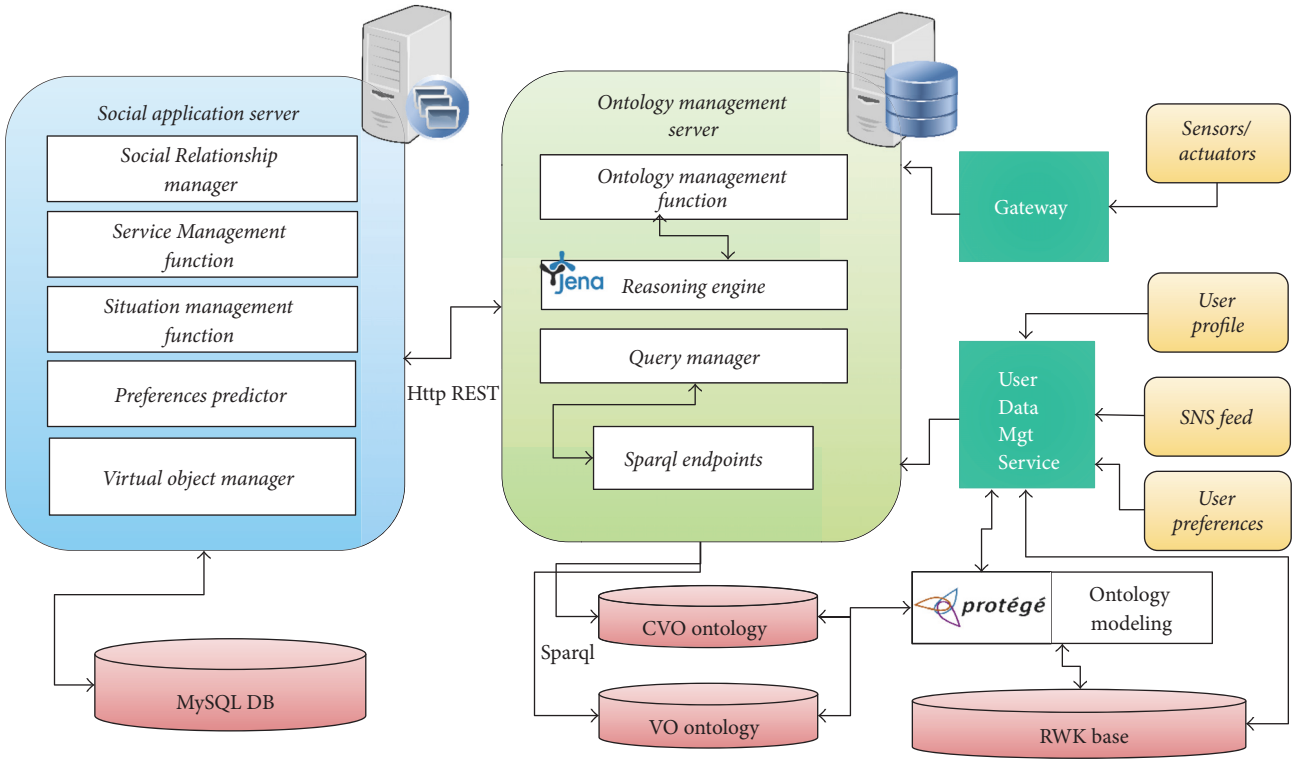

Figure 8: Prototype implementation components.

intended destinations, and user profile reader CVO which acquires user personal information from the social media profile. Several other CVOs have been developed that acquire diverse data and metadata from VOs where each represents a different RWO. These VOs include a user VO, smartphone VO, camera VO, wristband VO, motion detector VO, and others that are associated with the chosen environment (such as CO_2 sensor VO, humidity VO, temperature VO, and light VO). Each VO has several instances; for example, temperature sensor instances are represented as temperatureSesor0001 and temperatureSesor0002. Further VO data is collected and analyzed by CVOs with the rules incorporated in the ontologies. CVOs composed in microservices help analyze different situations and initiate inference over data using inference engines. Considering the system requirement, several aspects have been evaluated, including identification of web objects for effective IoT service provisioning, finding the social relationships among web objects, efficient discovery of web objects based on these relationships, and composition and harmonization of web objects.

4.3. Prototype Implementation Details. To analyze the feasibility of the proposed architecture, a prototype on the discussed use case has been implemented. Figure 8 illustrates the components of the prototype, which constitute social application server (SAS), ontology management server (OMS), and databases to hold VO and CVO graphs that are supported with SPARQL endpoints. The gateway connecting sensors and actuators has been incorporated in the implementation. Additionally, user data management services are implemented to get user profile and preferences history along with social network data.

The application server comprises five major components: social relationship manager (SRM), service management function (SMF), situation/context management function (CMF), preference predictor, and virtual object manager (VOM). The SMF is an entry point for the service request; it interprets and evaluates the service request. It is also responsible for matching the service request with the available template. SMF communicates with VOM to check the VO or CVO availability. A compiled list of VOs is provided to SMF which generates object mashups to satisfy the service request. The execution of VO mashup graph is done using microservices. The second function, VOM, maintains instances of CVOs and associated VOs and coordinates with OMS to use available ontologies in the system. It records the association of services and VOs and enables the reuse of VOs by more than one service; it also resolves conflicts using conflict resolution mechanism. VOM is implemented using RESTful web services. Further, SRM is implemented to maintain object relationships in the form of RDF concepts represented in ontologies. It also involves several microservices, each implemented for a specific task. For example, group management microservice is developed to group objects that have similar characteristics. Similarly, other tasks are distributed in different microservices; for example, link management microservice is used to identify the dependency among objects. Relationship observer microservice is implemented to monitor the objects when they change their state at runtime, whereas to search a web

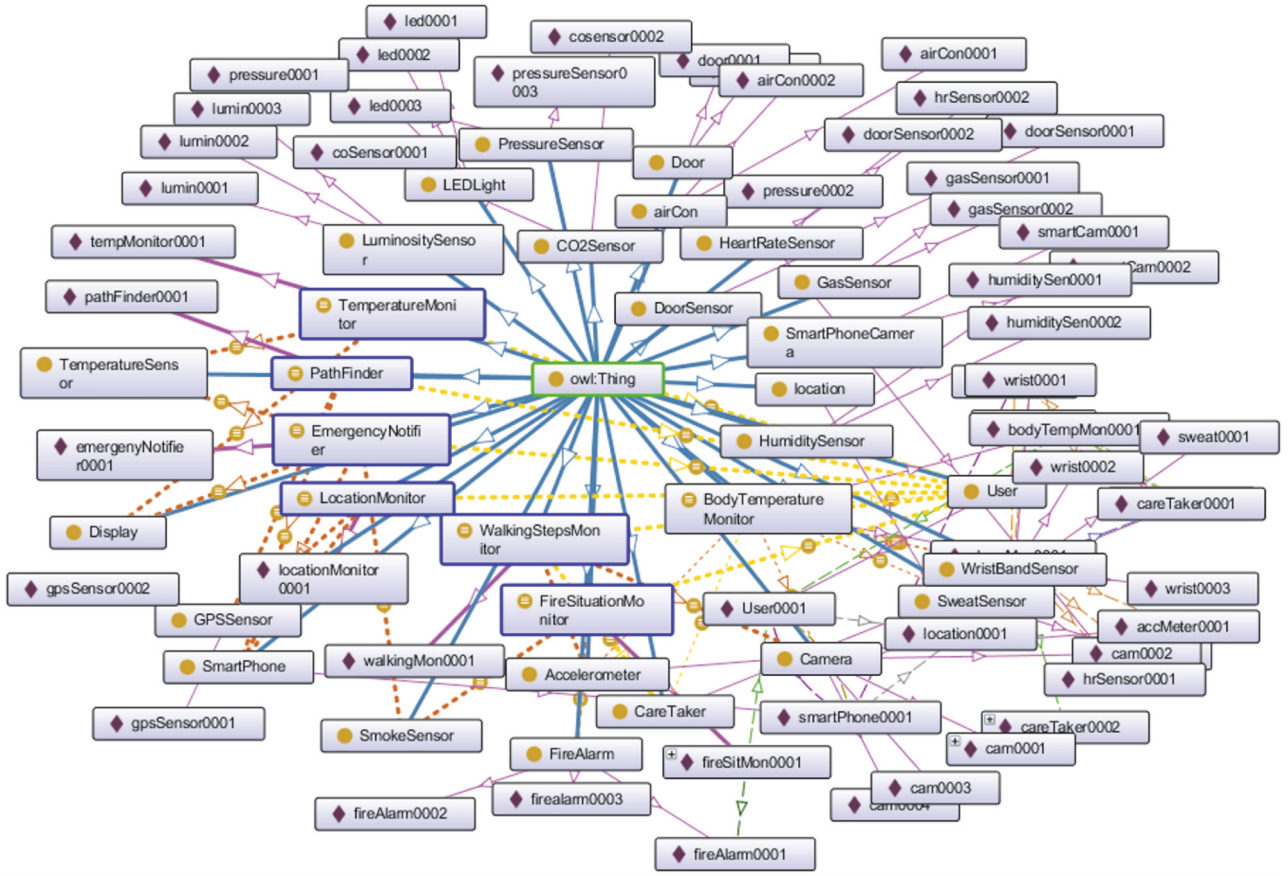

Figure 9: Ontological model of social web objects.

object relationship, the lookup microservice is used. The CMF function has been developed to process the situational information and it selects the most suitable VOs based on the context. The situational information contains the current location of the user, location of the RWOs, and the area under observation. CMF helps form a VO configuration specific to the user current situation. Additionally, to incorporate the real-world facts about the user and the situation, RWK base has been developed which is updated continuously. This also helps in adapting services according to the current situation. VOs provided data is accumulated in RWK model and the reasoning engine is used to infer new facts based on the existing data. The last function of social application server is the preference predictor which provides predictions by using machine learning methods. To implement the learning function, Weka Library [42] has been used. For the objective of demonstration, user preference patterns are collected from the social data and user preference history is used to predict the future patterns.

On the other hand, OMS deals with the semantic representation of VOs. The first functional component of OMS is the ontology management service, which is used to handle VO and CVO ontologies and provides an interface to use VO graphs. Further, to manage ontologies in the developed system, we have used Apache Jena framework [43], and to model VO and CVO ontologies, Protégé [44] has been used. Models generated in Protégé are utilized by ontology management service and stored in OWL format in semantic data stores. To take full advantage of ontology-based implementation, a reasoning engine has also been used to get extra facts related to the modeled concepts. Reasoning engine helps to grow RWK facts stored in VO and CVO ontologies. For the proof of concept, the prototype has used Apache Jena inference engine along with Jena provided reasoner. For the deployment of persistent storage of the RDF data, Jena TDB has been used in our prototype, which is the Jena native high-performance triple store with API support [43]. Query manager is another important component of OMS which provides an interface to query CVO and VO ontologies saved in RDF/XML format, using SPARQL. Also, web services are implemented for the execution of SPARQL queries such as extracting and updating RDF triples in the database. Other than SPARQL query processing, semantic matching of existing concepts is also performed.

Moreover, to achieve interoperable relationships among web objects, a semantic ontology model has been developed as shown in Figure 9. The model constitutes the VO and CVO ontologies and their relationships. Several CVOs have been instantiated, for example, the location navigator, fire situation monitor, emergency notification, and many others that are

```
<!-- http://www.webofobjects.com/hufs/smartservice#01 -->
 <owl:Class rdf:about="http://www.webofobjects.com/hufs/smartservice#01#SmartMuseumService">
     <!-- http://www.webofobjects.com/hufs/smartservice#01#fireSitMon0001 -->
     <owl:NamedIndividual rdf:about="http://www.webofobjects.com/hufs/smartservice#01#fireSitMon0001">
         <rdf:type rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#FireSituationMonitor"/>
         <smartservice:CapturedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#cam0001"/>
         <smartservice:NotifyTo rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#fireAlarm0001"/>
         <smartservice:NotifyTo rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#smokeSensor0001"/>
         <smartservice:checkedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#User0001"/>
     </owl:NamedIndividual>

     <!-- http://www.webofobjects.com/hufs/smartservice#01#PathFinder0001 -->
     <owl:NamedIndividual rdf:about="http://www.webofobjects.com/hufs/smartservice#01#PathFinder0001">
         <rdf:type rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#PathFinder"/>
         <smartservice:checkedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#User0001"/>
         <smartservice:checkedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#surveillanceSystem0001"/>
         <smartservice:notifiedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#location0001"/>
         <smartservice:notifiedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#smartPhone0001"/>
         <smartservice:notifiedBy rdf:resource="http://www.webofobjects.com/hufs/smartservice#01#motionDetector0002"/>
     </owl:NamedIndividual>
 </owl:Class>
```

Figure 10: CVOs and VOs used in the ontology.

```
<!-- http://www.webofobjects.com/hufs/smart-service-ontology#smartPhone0001 -->

<rdf:Description rdf:about="http://www.webofobjects.com/hufs/smart-service-ontology#smartPhone0001">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
    <rdf:type rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#SmartPhone"/>
    <hasLocation rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#location0001"/>
    <hasOwner rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#User0001"/>
    <batteryStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string">70%</batteryStatus>
    <deviceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Samsung-S5</deviceName>
    <modelNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SHV-E770S</modelNumber>
    <networkStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string">WIFI</networkStatus>
    <osVersion rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Andriod version 5.0.1</osVersion>
    <smartPhoneId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">S5-00:33:55:11:00</smartPhoneId>
    <smartPhoneManufacturer rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Samsung</smartPhoneManufacturer>
</rdf:Description>
```

Figure 11: Smartphone VO used in the ontology.

based on the composition of numerous VOs. This development has been supported by OMS and semantic web tools such as Jena. Using OMS, CRUD operations are performed like create, update, and delete on CVOs and VOs using SPARQL. In the developed semantic relationships, two CVO instances, namely, "FireSitMon0001" and "PathFinder0001," have been used in the code snippet provided in Figure 10, for fire situation monitoring and location navigation, respectively. We can see the data properties and object properties associated with each object to maintain the relationship with other objects in the system. Further, to understand the VO representation, a smartphone VO "SmartPhone0001" is shown in Figure 11 with the defined properties. A list of VOs used in our prototype has been given in OWL description as shown in Figure 12.

In the test bed environment, the sensor to the gateway connectivity is enabled using ZigBee and BLE. Another source of information that has been incorporated to provide user-centric service capability is SNS feed, which consists of Twitter and Facebook data collected using social media APIs. On the other hand, user profile and preferences are updated using data management service, and this information is used to grow RWK base.

To test the prototype, an android application has been developed. After the user logs in to the app for WoO based assisted services, the app starts the configuration of services according to the user profile. A list of services is presented on the user's screen from which the user can select any service. These services include museum visit service, location navigator service, recommendation service, and emergency notification. The selected service starts collecting information with the interaction of the user or when the devices start pushing values about any change in the observed environment. Based on these observed parameters, alerts are generated by the app; for example, the museum visit service generates alerts about the available facilities in the museum and the environmental conditions including the indoor and outdoor temperatures. On the other hand, the location navigator service helps the user to navigate around the museum by showing directions to several visitor attractions. Moreover, the emergency notification service triggers alerts when a temporary fire is created to mimic an unusual situation such as fire breakout. The resulting fire increases the temperature reading which is detected by the temperature sensor. This change in reading is forwarded through the gateway and analyzed by the fire situation monitor CVO which checks its threshold value and generates an emergency notification on the user's app screen. Also, the recommendation service is used to generate suggestions about the museum events and the places to visit.

```
<!-- http://www.webofobjects.com/hufs/smart-service-ontology#deviceName -->

<rdf:Description rdf:about="http://www.webofobjects.com/hufs/smart-service-ontology#deviceName">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#CO2Sensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#Camera"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#DoorSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#FireAlarm"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#GPSSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#GasSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#HeartRateSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#HumiditySensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#LEDLight"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#LuminositySensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#PressureSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#SmartPhone"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#SmartPhoneCamera"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#SmokeSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#SweatSensor"/>
    <rdfs:domain rdf:resource="http://www.webofobjects.com/hufs/smart-service-ontology#WristBandSensor"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Description>
```

Figure 12: List of VOs used in the ontology.

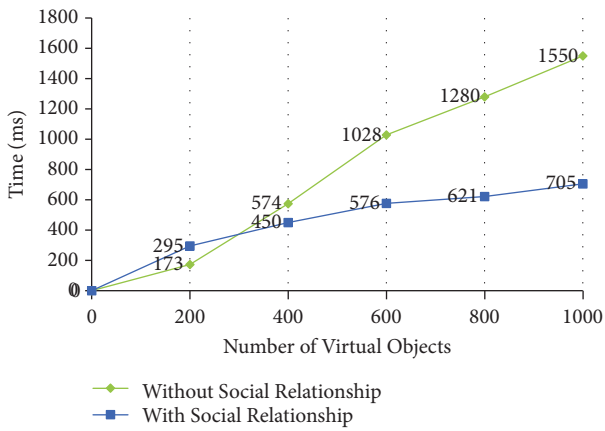

Figure 13: VO discovery time with the increasing number of virtual objects.

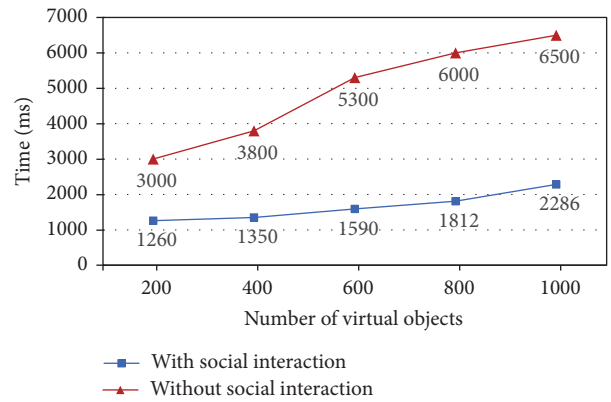

Figure 14: Service execution time with the increasing number of virtual objects.

4.4. Performance Evaluation. The proposed architecture has been analyzed from two different perspectives: system scalability and resource consumption. The results with respect to the experiments and the performance criterion are discussed in this section. The first experiment involves the analysis of the time required to discover the virtual objects in the proposed system with or without social relationship criterion as shown in Figure 13. As it is apparent from the results, initially, to form a social graph, VO discovery process incurs delay due to the communication for establishing social relationship links. However, direct lookup for VOs in the registry at the start of the discovery process comparatively requires less time, but as the number of VOs increases, the delay rises. Traversing the social links to discover objects as the number of virtual objects grows considerably reduces the total lookups in the registry database.

The second experiment provides an analysis of the execution time required for the CVOs as shown in Figure 14. To test this in a real environment, an android application has been developed. The app allows access to the parameters from the available VOs. When the app requires retrieving observations, it iterates through VOs in the repository using a query interface. This requires high CPU utilization with more execution time required to retrieve the sensor values against each app generated request. This is due to the fact that the decision for the selection of VOs has been done at runtime; it requires more time to select the VO graphs, execute them, and get the relevant data from the VOs. However, when object interaction is based on the social relationship, it requires less time for the retrieval as friends in the social network share the data on other friends' requests, which results in less computation as compared with the previous scheme.

Moreover, Figure 15 depicts the time to discover VOs in four different types of services. These services are the museum visit service, the location navigator service, the emergency notification service, and the recommendation service. The discovery time varies from one service to another depending on the number of VOs used. However, as compared with the impact of social connections, it can be

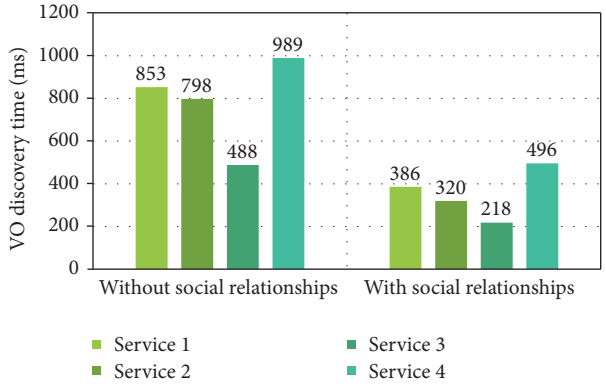

Figure 15: VO discovery time for four different services with or without social relationship utilization.

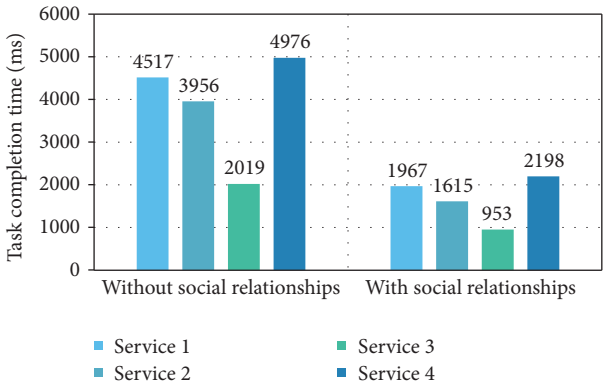

Figure 16: Task completion time required for each service with or without social relationship exploitation.

realized that the services can considerably reduce the time of object discovery and execution by using the relationship model. This is due to the fact that with social association among web objects a service does not have to query the registry for all VOs and rather VOs in the same social graph are utilized. Also, VOs collected observations are shared to save the service time for probing each VO. To view this, Figure 16 shows the overall time required for services to execute their tasks. An important factor to notice here is that the overall task completion time is also affected when social links are used to collect observations from VOs that are used in the service execution.

5. Conclusion

The Internet of Things is bringing the next technological revolution by connecting billions of objects on the earth and providing intelligent IoT services. However, it also carries two most important challenges: the first one is the complexity of handling a huge number of heterogeneous objects and the second one is how to deal with the monolithic approaches for providing services over existing IoT infrastructure. To address these challenges, we have proposed an integrated design based on the principles of SIoT, where the complexity has been handled using socially connected web objects; to overcome the monolithic approaches, microservices that can compose new service features independently have been adopted. In the proposed design, a social relationship model has been presented, which enables the efficient discovery of web objects and reduces the complexity of service provisioning process with the algorithms to discover and compose web objects. A semantic ontology model has also been developed to realize the interoperable social interaction among heterogeneous objects. Finally, a prototype implementation based on a use case scenario has been demonstrated; to evaluate the system with respect to the performance issues, experimental results pertaining to the VO discovery and service execution time have been rendered.

Details of Notations Used in Algorithms

$\mathbb{\Sigma}\phi$: Service object ontologies based on semantic representation where ϕ is a replacement for M_μ which represents microservices M_c which represents CVO and M_v are VO ontologies

$\mathbb{\Pi}_L$: Collection or list of all relationships that are retrieved from service objects

$\mathbb{\Pi}_U$: List of the objects that do not have a relationship or are not associated

$\hat{S}$: Social relationship graph of objects

U: Correlation union of templates

$\underline{\mu\overline{\top}}$: All service objects returned in response to the query request

$\overline{\top}$: All objects returned in response to the query request

μ: The single instance iterated from the collection of microservices

M: Data model based on the specified ontology, where $\mathbb{\Sigma}M_s$ represents service data model, $\mathbb{\Sigma}M_c$ represents context data model, and user profile model is represented as $\mathbb{\Sigma}M_u$

Q: SPARQL query to retrieve the available service templates for user rating

Q': SPARQL query to retrieve the relationship associated with services

$\mathbb{R}_e$: Registry entries of service objects

C_R: Relationship associated with CVO

V_R: Relationship associated with VO

R_c: All possible relationships that can be associated with CVO

R_v: All possible relationships that can be associated with VO

O_i: Iterator object instance for iterating the list of CVO and VO objects

$\mathbb{Z}$: Set of objects' social relationships

σi: Iteration item of the list of service objects

λi: An instance of iteration items in the list of service objects

$\mathbb{\Pi}_{Lm}$: List of all matched service items

ϖi: An instance of iteration items in the list of matched service items

$\mathbb{I}_{LR}$: List of all matched ranking
∂: Ranking value assigned to a service object
σj: Iterator item for the ranking list
κ: Threshold to rank a service object
Wf: Workflow for the composition of service objects
Pq: Priority queue to store ranked object instances
R∂: Assigned relationship ranking.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Korea Institute for Advancement of Technology (KIAT) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) (N040800001, Development of Web Objects Enabled EmoSpaces Service Technology).

References

[1] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the internet of things: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2016.

[2] J. Surowiecki, *The Wisdom of Crowds*, Anchor Books, 2005.

[3] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT)—when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[4] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, 2014.

[5] L. Atzori, D. Carboni, and A. Iera, "Smart things in the social loop: Paradigms, technologies, and potentials," *Ad Hoc Networks*, vol. 18, pp. 121–132, 2014.

[6] L. Atzori, I. Antonio, and G. Morabito, "Making things socialize in the Internet -Does it help our lives?" in *Kaleidoscope 2011: The Fully Networked Human?-Innovations for Future Networks and Services (K-2011), Proceedings of ITU. IEEE*, 2011.

[7] L. Atzori, A. Iera, and G. Morabito, "SIoT: giving a social structure to the internet of things," *IEEE Communications Letters*, vol. 15, no. 11, pp. 1193–1195, 2011.

[8] A. M. Vegni and V. Loscrí, "A survey on vehicular social networks," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, article no. A3, pp. 2397–2419, 2015.

[9] L. Krause, *Microservices: Patterns and Applications: Designing Fine-Grained Services by Applying Patterns*, 2015.

[10] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud '15)*, pp. 25–30, Rome, Italy, August 2015.

[11] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," in *Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016*, Germany, September 2016.

[12] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How To Make Your Application Scale," *Computing Research Repository*, 2017.

[13] "Microservices Guide Pages - Martin Fowler," 2017, https://martinfowler.com/articles/microservices.html.

[14] "Web of Objects. ITEA 3 · Project · 10028 , D2.1: State-of-the-Art Relevant to the Web of Objects," 2017, https://itea3.org/project/web-of-objects.html.

[15] "Y.4452: Functional framework of web of objects," 2017, http://www.itu.int/rec/T-REC-Y.4452-201609-P.

[16] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl5, and H.-W. Gellersen, "Smart-its friends: a technique for users to easily establish connections between smart artefacts," in *Ubicomp 2001: Ubiquitous Computing*, pp. 116–122, Springer, Berlin, Germany, 2001.

[17] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable Web of Things," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2010*, pp. 702–707, Germany, April 2010.

[18] M. Kranz, L. Roalter, and F. Michahelles, "Things that twitter: social networks and the internet of things.' What can the Internet of Things do for the Citizen (CIoT)," in *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive 2010)*, 2010.

[19] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for iot distributed applications over socially connected objects," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 40–51, 2017.

[20] A. Pintus, D. Carboni, and A. Piras, "Paraimpu: A platform for a social Web of Things," in *Proceedings of the 21st Annual Conference on World Wide Web, WWW'12*, pp. 401–404, France, April 2012.

[21] I. Farris, R. Girau, L. Militano et al., "Social Virtual Objects in the Edge Cloud," *IEEE Cloud Computing*, vol. 2, no. 6, pp. 20–28, 2015.

[22] R. Girau, M. Nitti, and L. Atzori, "Implementation of an experimental platform for the social internet of things," in *Proceedings of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013*, pp. 500–505, July 2013.

[23] M. A. Jarwar, R. A. Abbasi, M. Mushtaq et al., "CommuniMents: A framework for detecting community based sentiments for events," *International Journal on Semantic Web and Information Systems*, vol. 13, no. 2, pp. 87–108, 2017.

[24] S. Newman, "Building microservices: designing fine-grained systems," 2015.

[25] "The Single Responsibility Principle - Programmer 97-things," 2017, http://programmer.97things.oreilly.com/wiki/index.php/The_Single_Responsibility_Principle.

[26] F. Viktor, "The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices," 2016.

[27] "iCore: Internet Connected Objects for Reconfigurable Ecosystems, European FP7 Project," 2017, http://cordis.europa.eu/project/rcn/100873_en.html.

[28] D. Namiot and M. Sneps-Sneppe, "On microservices Architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.

[29] D. Bonino, M. T. D. Alizo, A. Alapetite et al., "ALMANAC: internet of things for smart cities," in *Proceedings of the 3rd*

International Conference on Future Internet of Things and Cloud (FiCloud '15), pp. 309–316, IEEE, Rome, Italy, August 2015.

[30] "Developing Microservices for PaaS with Spring and Cloud Foundry," 2017, https://www.infoq.com/presentations/micro-services-pass-spring-cloud-foundry.

[31] "Microservices in action, Part 2: Containers and microservices — a perfect pair," 2017, https://www.ibm.com/developerworks/cloud/library/cl-bluemix-microservices-in-action-part-2-trs/index.html.

[32] J. E. Kim, X. Fan, and D. Mosse, "Empowering End Users for Social Internet of Things," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation - IoTDI '17*, pp. 71–82, Pittsburgh, Pa, USA, April 2017.

[33] Z. U. Shamszaman, S. S. Ara, I. Chong, and Y. K. Jeong, "Web-of-Objects (WoO)-based context aware emergency fire management systems for the Internet of Things," *Sensors*, vol. 14, no. 2, pp. 2944–2966, 2014.

[34] S. Ali, M. G. Kibria, and I. Chong, "WoO enabled IoT service provisioning based on learning user preferences and situation," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 474–476, Da Nang, Vietnam, January 2017.

[35] M. A. Jarwar, S. Ali, M. G. Kibria, S. Kumar, and I. Chong, "Exploiting interoperable microservices in web objects enabled Internet of Things," in *Proceedings of the Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 49–54, Milan, Italy, July 2017.

[36] M. Kibria, S. Ali, M. Jarwar, S. Kumar, and I. Chong, "Logistic Model to Support Service Modularity for the Promotion of Reusability in a Web Objects-Enabled IoT Environment," *Sensors*, vol. 17, no. 10, p. 2180, 2017.

[37] M. Kibria, S. Fattah, K. Jeong, I. Chong, and Y. Jeong, "A User-Centric Knowledge Creation Model in a Web of Object-Enabled Internet of Things Environment," *Sensors*, vol. 15, no. 9, pp. 24054–24086, 2015.

[38] M. A. Jarwar and . Ilyoung Chong, "Exploiting IoT services by integrating emotion recognition in Web of Objects," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 54–56, Da Nang, Vietnam, January 2017.

[39] S. Ali, H.-S. Kim, and I. Chong, "Implementation model of WoO based smart assisted living IoT service," in *Proceedings of the International Conference on Information and Communication Technology Convergence, ICTC 2016*, pp. 816–818, October 2016.

[40] S. Kumar, M. G. Kibria, S. Ali, M. A. Jarwar, and I. Chong, "Smart spaces recommending service provisioning in WoO platform," in *Proceedings of the International Conference on Information and Communications (ICIC)*, pp. 311–313, Hanoi, Vietnam, June 2017.

[41] K. M. Alam, M. Saini, and A. El Saddik, "Toward social internet of vehicles: concept, architecture, and applications," *IEEE Access*, vol. 3, pp. 343–357, 2015.

[42] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," 2017, https://www.cs.waikato.ac.nz/ml/weka/.

[43] "Jena, Apache. "A free and open source Java framework for building Semantic Web and Linked Data applications.&quot," 2017, http://jena.apache.org/.

[44] Protégé, "A free, open-source ontology editor and framework for building intelligent systems," 2017, https://protege.stanford.edu/.