Mathematical Biosciences and Engineering

MBE, 18(6): 9312–9335.
DOI: 10.3934/mbe.2021458
Received: 20 August 2021
Accepted: 12 October 2021
Published: 27 October 2021

http://www.aimspress.com/journal/MBE

Research article

Modeling-framework for model-based software engineering of complex Internet of things systems

Khurrum Mustafa Abbasi [1], Tamim Ahmed Khan [1,*] and Irfan ul Haq [2]

[1] Department of Software Engineering, Bahria University Islamabad, Pakistan
[2] Department of CIS, PIEAS Islamabad Pakistan

*** Correspondence:** Email: tamim@bahria.edu.pk.

Abstract: Internet of things (IoT) systems are composed of variety of units from different domains. While developing a complete IoT system, different professionals from different domains may have to work in collaboration. In this paper we provide a framework which allows using discrete and continuous time modeling and simulation approaches in combination for IoT systems. The proposed framework demonstrates on how to model Ad-hoc and general IoT systems for software engineering purpose. We demonstrate that model-based software engineering on one hand can provide a common platform to overcome communication gaps among collaborating stakeholders whereas, on the other hand can model and integrate heterogeneous components of IoT systems. While modeling heterogeneous IoT systems, one of the major challenges is to apply continuous and discrete time modeling on intrinsically varying components of the system. Another difficulty may be how to compose these heterogeneous components into one whole system. The proposed framework provides a road-map to model discrete, continuous, Ad-hoc, general systems along with composition mechanism of heterogeneous subsystems. The framework uses a combination of Agent-based modeling, Aspect-oriented modeling, contract-based modeling and services-oriented modeling concepts. We used this framework to model a scenario example of a service-oriented IoT system as proof of concept. We analyzed our framework with existing systems and discussed it in details. Our framework provides a mechanism to model different viewpoints. The framework also enhances the completeness and consistency of the IoT software models.

Keywords: Internet of things; modeling complex systems; modeling framework; software engineering; service-oriented computing; model driven engineering

1. Introduction

Internet of things systems are based on three basic elements. The first unit is hardware i.e. devices. The second unit is software i.e. programs for performing functions of those hardware. The third is connection to the internet i.e. communication and interaction with other devices. While modeling a system, viewpoint is an important parameter to consider. The purpose of the model varies on the basis of viewpoints of stakeholders. The target stakeholders and purpose of the model can be well defined by asking a set of questions. These questions may include: who is going to use this model? How much details should be provided to keep a balance between the abstraction level and complexity? Is the target a professional or a common person? Will a single model be sufficient for all the stakeholders or different models should be developed? After answering these questions, there is always possibility of compromise between the details, clarity and level of abstractions. To keep the balance and reduce the compromises different models are developed. Models demonstrate the composition of the system and behaviors and interactions of the subsystems.

Internet of things involves physical things connected through internet and often leads to development heterogeneous and complex systems [1,2]. Internet of Things is a ubiquitous connection of smart devices performing variety of tasks [3,4]. However, aggregation of various technologies to the Internet of Things based systems and some of these technologies are emerging such as Blockchain make systems more complex [5]. These things may have continuous-time data like temperature of an incubator, speed of a car and pressure of an electric rice cooker. For continuous time systems, we require continuous modeling and simulation approaches. Whereas on the other hand there are various IoT systems which are generating and using discrete time data. Such systems are usually state-based. State-based systems require discrete time modeling and simulation approaches and tools. Different aspects are required to consider while modeling IoT systems such as, either use a graph-based approach or an entity focused approach. Another point to focus is either to go for an Ad-hoc or general one.

Internet of Things systems design and development may involve personals from different backgrounds to work in collaboration. Mechanical objects may be equipped by electronic kit and controlled through software connected to internet. This combination promotes a need of mechanical, electrical/ control system, telecoms/ network and software personals to work in collaboration. Here, we focus on software engineering perspective of this system. The software engineer is required to develop software for electronic toolkit attached to any machine, for the purpose of communication, for developing services, for implementing business process and for implementing applications for the end user. Meanwhile, software engineers involve different roles such as requirement engineering, design and architecture, implementation, testing, deployment and evolution. In such a diverse system development, we may consider the system with the term used by Seymour Papert "an object to think with" [6]. We may use modeling approaches for the purpose of communication among such diverse stakeholders.

In software engineering models are used for different purposes. The models that are used for understanding the context information about a specific problem are called domain models. This type of model may not be aimed at providing a software solution but the main purpose is the representation of major concepts in real-world problem. It is used to represent the properties of domain under consideration such as to represent business processes. The second type of modeling used in software engineering is specification modeling. This type of modeling is used to represent the

specifications in a formalized way. The third type of modeling used in software engineering is design modeling which is also named as implementation modeling. The design modeling provides control flow and is used to represent the behaviors of various parts and allocate their responsibilities. It is used to represent different aspects of system such as modularization of the system, interaction of the components and composition of the system. However, there are different perspectives of modeling in software engineering.

Reference Architectures play vital role in elaborating building blocks of a system. Internet of Things Architecture (IoT-A) is one of the earliest and well recognized reference architectures [7]. Similarly, for industrial IoT, Industrial Internet Reference Architecture (IIRA) [8] and Reference Architecture Model for Industries 4.0 (RAMI 4.0) [9] have gained significant recognition. It is a complex task to build model using reference architectures as the reference architectures are very abstract and specifications of systems vary [10]. Hence, a framework for modeling internet of things systems in perspective of software engineering is required. In this paper we propose a framework which uses a combination of different modeling approaches that we have analyzed and mapped against different layers of IoT-A reference architecture in our previous article [11]. The modeling approaches which we integrated in this framework are agent-based modeling, aspect-oriented modeling, service-oriented modeling and contract-based modeling. The integration of these modeling approaches helps in developing a variety of models for IoT systems.

This paper is contributing in the following ways:

- Provides a framework which allows using discrete and continuous time modeling and simulation approaches in combination for IoT systems.
- The proposed framework demonstrates on how to model Ad-hoc and general IoT systems for software engineering purpose.
- It also considers the procedure for modularization and composition of the software for IoT systems.

In section-2, the background of research has been provided. In section-3, the proposed framework for modeling has been discussed. In section-4, the framework has been used to model a scenario of service-oriented Internet of things system. In section-5 we provide analysis and discussion. In the last section we concluded the article and provided some future directions.

2. Background

Internet of things business modeling is an important aspect. Business models are based on building blocks which help to enhance the business; a framework for IoT business model is required [12]. The three elements of IoT are interaction, communication and networking of interconnected objects [13]. There are three types of IoT infrastructure and business models i.e. Telecommunication, Internet and Industry [14]. Multimedia objects are very important part of internet of things. The feature of multimedia objects has been neglected by the research community for the internet of things [15]. These objects deal with cloud services. Abstracting of heterogeneity is required to represent the functionality of the objects [16]. There are two approaches for the payment of services in Internet of things [17]. There is a need of cooperative business model in Internet of Things [18]. Internet of things business market need to have a business model in which they can cooperatively interact. Interaction of different service providers will boost the trend. Also, there is a

need of mechanism to define the trustworthiness of applications [19]. Trustworthiness is an important feature. Hence there is also a need of trustworthiness of environment in which IoT applications run. Some features that internet of things model should support are heterogeneity, scalability, localization, self-organization, energy optimization and security as well as privacy [13]. The distributed model of IoT should have openness, security, reliability, viability, data management, interoperability and scalability [20].

Agent-based Modeling (ABM) is a technique which uses set of agents to analyze the behavior of interacting objects in model [21]. This approach may be used to model interacting and decision-making objects [22]. The benefits of agent-based modeling include flexibility and simulation friendliness. Simulations of agent-based models have been used to predict the behavior of certain agents in large scale emergency situations [23]. Hence, in disaster situations and emergency conditions it provides an effective way to analyze the emerging behavior of people [24]. Components of a system may be represented as agents to model complex systems [25].

Complex scenarios can be modeled using agent-based modeling by distinguishing the actors in the categories of agents and meta-agents [26]. For macro-economic research agent-based modeling may be used to model heterogeneous interacting agents in an efficient way [27]. Due to its user friendliness and providing a way to model continuous time systems in an efficient way, it has replaced the mathematical modeling at a large scale. Social sciences prefer agent-based modeling over mathematical modeling due to its easiness and effectiveness [28].

Contract is an agreement between two parties. According to [29], there are four basic types of contract models. The first type of contract models is Obligation free contracts. The second type of contract models is User Centric Contract. The third type of contract models is Provider Centric Contract. The fourth type of contract models is Customizable contract. Options are also a type of contracts in which the customer don't have any obligations but only have rights. American and the European options are two well known types of options normally termed as standard options [30]. Forward contract is a type in which there is firm commitment between both parties [31]. The inconsistency due to network latency can be managed by using predictive contracts [32]. Contracts are used in software engineering for different purposes. Some of the uses of contracts in software engineering are guarding one part from other part of the program [33], for checking the validity of the claims about different parts and flow of the values [34], as program contracts for describing the behavior [35], in web services in the form of service level agreements [36], in distributed systems for ensuring interoperability [37], interaction between components of software [38] and for e-commerce purpose [39].

Contract-based modeling has been used for software engineering. The contracts are composed of left-hand and right-hand side. In some contracts there is require and provide side. Whereas, the terminologies of assumption and grantee has also been used. Software may be validated against the specifications by using contract-based modeling [40]. Blockchain technology provides a way of interaction among trustless parties. It has been adopted by different industries and especially for crypto-currency [41]. It uses smart contracts which are programmable and are secured in blocks [42].

Aspect-orientation is used for modularization and for separation of concerns i.e. crosscutting and non-crosscutting concerns [43]. It provides a way to modularization, removal of conflicts among modules and interaction of the modules [44]. An aspect is often a requirement which is partially implemented in two or more classes [45]. Aspect oriented software engineering uses model driven architecture [46]. It may also use formal notations and graphical notation together [47]. For security

and quality focused systems, it is very effective to save the system from various attacks and to enhance quality of services [48]. It may also be useful for distributed internet of things software systems development [49].

Service-oriented software development is based on service-oriented architecture. It provides a flexible composition of the system. In combination with multi-agent systems, it may be used for development of intelligent systems [50]. It is now replacing application-oriented solutions. Reliability of service in IoT systems is hard to analyze and model [51]. It may be used in combination with model driven architecture to deal with challenging issues of enterprise information systems [52]. One of the differences between software services and IoT service is the dependency on physical objects in IoT. For IoT system Ontology of Semantic Markup for Web (OWL-S) model with some extensions may be [53]. There are four attributes to consider while using OWL-S that are inputs, outputs, preconditions and effects [53]. Micro-services are used for developing system by composition of subsystems [54].

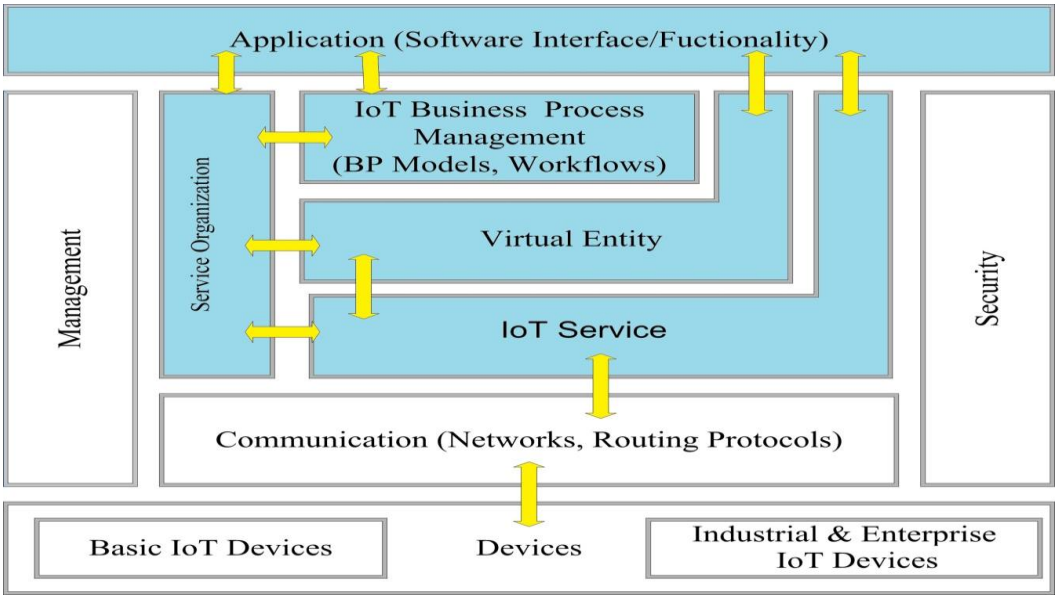

Figure 1. IoT-A Functional Model with highlighted software engineering layers [7].

3. Framework for modeling

The framework that we propose here uses a combination of four modeling approaches for modeling complex IoT systems for the purpose of software engineering. These approaches are agent-based modeling, contract-based modeling, aspect-oriented modeling and service-oriented modeling. These techniques have already been used for software engineering and especially model-based software engineering [11]. Due to the use of different techniques this framework provides a way to model IoT systems at different levels. Agent-based modeling is better for flexible models with minimum details. This approach may also be helpful in modeling where target people have limited domain knowledge and simulations are needed for demonstrations.

Aspect-oriented modeling provides more details compared to agent-based modeling. It is useful when target viewer has at-least some domain knowledge. It may be used to prone out the

requirements and remove the clashes among different modules.

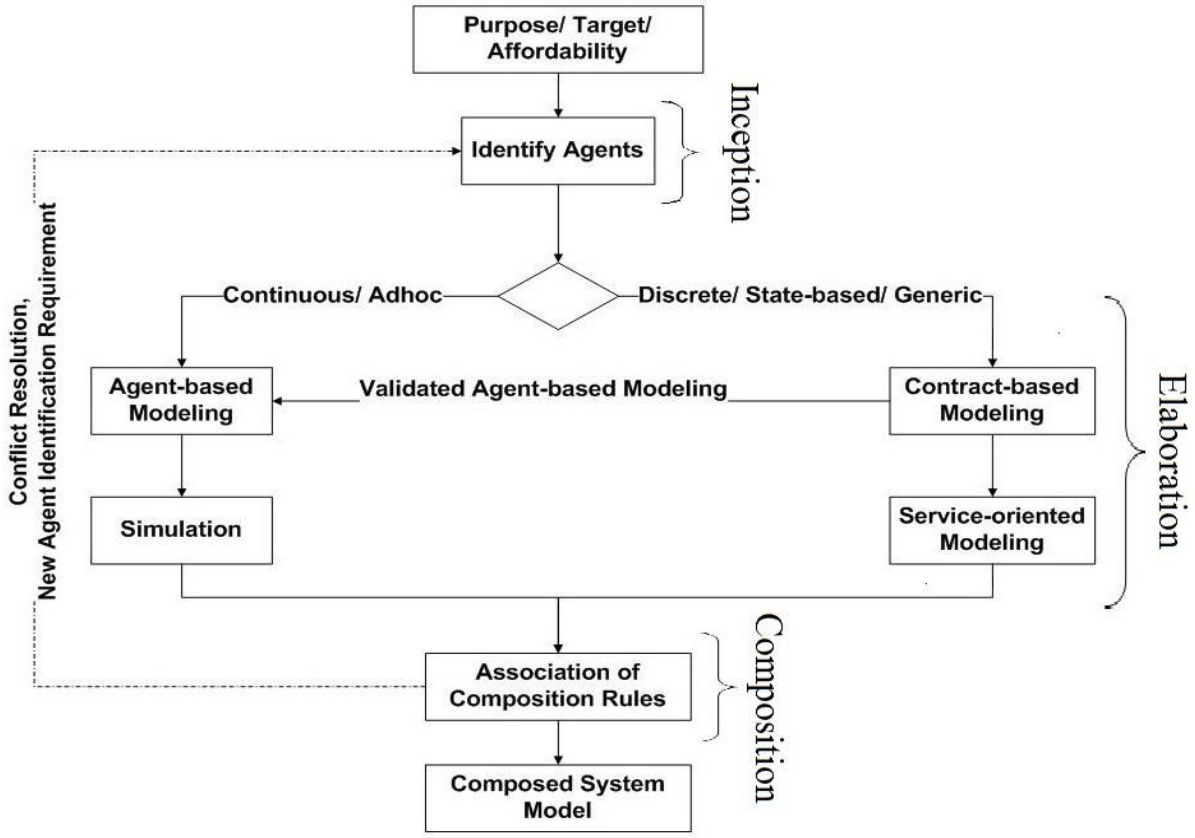

Figure 2. Block-diagram of Modeling Framework.

It may be used for requirements engineering of IoT systems. Contract-based modeling provides more details. It includes the features of the objects as well. It lies very close to the programming and hence it may be best understood by people of relevant domain. Service-oriented modeling is based on service-oriented architectures. It represents the system as a combination of services and micro services. Software services have provided interface but not requires interface. So, services are much independent components of the system. In Figure 1, we highlighted the layers of the IoT-A reference architecture where key contribution of software engineering for system development is expected. Our framework shown in Figure 2 provides mechanism to model system keeping these highlighted layers under consideration. The framework is based on three phases that are, Inception, Elaboration and Composition. In the Inception phase the agents which basically are components, are identified. In the Elaboration phase the details about the agent or components are elaborated. In the Composition phase the identified and elaborated agents are composed in system.

The steps involved in modeling while using our framework are: first of all, defining the purpose, target and affordability of the model. The second step is defining the building blocks as agents. The third step is defining the features of the agents for semantic representations. The fourth step is elaborating the behavior of the agents using the services concepts and agents relations. The fifth step is using the contracts and aspects for the composition of the system.

3.1. Purpose, target and affordability

There are various purposes of modeling. Modeling may be used to analyze systems based on the collected data, to transfer knowledge in an efficient way, to test a system or concept before implementation and to visualize any system or any component for design, development and testing. While modeling, the first and foremost important thing one has to define is the purpose of model. The second thing to know is the target viewer of the model. The modeling approach and the level of details highly depend on the target user of the model. The third point to consider is affordability of the model means how much efforts justify the importance of the model. The model may be of one time use or sometime it may have a lasting impact.

3.2. Inception by identification of agents

While modeling agents include any autonomous hardware or software or a unit of software. Agents may include but are not limited to Device, Component, Object, Service or a Human. Sub system or super system may also be treated as agent. There are some common characteristics of agents. The agent should be autonomous and should own a behavior for responding automatically. Other characteristics include Learning from environment, adoptability which means changing behavior according to situation and work in decentralized environment. Cognitive agents are a type of agents which have decision power. Agents have identity, name and behavior. There may be a group of agents with same behavior.

3.3. Elaboration by using contracts

Contracts are on record requirements and provisions by an agent explaining the way of interaction with other agents. In other words, it defines the features of the agents which lead to interaction. Contract is the conditions against which actions are taken. There are pre-conditions and post-conditions. These pre-conditions and post-conditions explain the behavior of entities. Pre-conditions and Post-conditions may also be termed as assumptions and guarantees. Contracts may be visualized for better understanding. Visual contracts may be used to describe the correlation of different entities.

We use contracts to describe agents' behavior in terms of their operations. We make use of visual contracts for this purpose, which are defined as a set of graphs representing pre- and post-conditions. We use double push out approach to model these contracts, as production rules, and we represent system as a typed attributed graph transformation system (TAGTS). These rules have associated rule signatures as explained in [55]. We represent class diagram as type graph and initial state of the system in terms of a start graph. We also make use of these contracts to define agent interactions where pre- and post-conditions explain behavior of entities, assumptions and guarantees.

3.4. Elaboration of services

When the agents identified are services then they are elaborated using the concepts of service-oriented modeling. Software Services are a part of service-oriented architecture. Service-oriented architectures are used for large software applications. In service-oriented architecture distributed, independent and deployed components are integrated. This architecture is

useful in development of web services and micro-services based systems. A service in service-oriented architecture may be a plateform independent module of software accessible by other application. For the purpose of composite modeling we term the service as a functionality provided by an agent to the other agent or a human.

Extensible Markup Language (XML) is commonly used for modeling service-oriented systems. There are several extensions of XML as well. XML uses tags which are not normally pre-defined. Based on XML new modeling languages for service-oriented systems have emerged. Web Service Description Language (WSDL) is based on XML. It is used to describe web services. XML Schema Definition (XSD) provides a way of formally describing the elements in XML. Extensible Stylesheet Language Transformations (XSLT) provides a way of transforming documents from XML to XML or other formats. Simple Objects Access Protocol (SOAP) is a protocol for web services which is also based on XML. Resource Description Framework (RDF) is also based on XML and is used to describe resources on web. XML and its extensions are open standards used by web services. So, XML plays a key role in modeling web services. Universal Description, Discovery and Integration (UDDI) contains information about the public interface of service, centralization of services and ease of publish/find operations. WSDL uses four major tags first one is < *types* >, this tag is used to define data types being used. Second one is < *message* >, this tag is used to define data elements. The third tag is < *portType* >, this tag normally contains the name of the port, the operation to be performed and the messages associated with the operation. The fourth tag is < *binding* >, this tag is used to define data format and protocol of port-type for the purpose of binding the service.

3.5. Separation of concerns and composition of system

Concern is information associated to the functionality of software. It may either be general or specific to a part of software. Concern is also term as aspect in software engineering. Concerns are Identified, Specified and Composed. So, discussing in terms of agents, aspects are the concerns of operation of any agent. A concern may be any activity required by an agent to perform its own task or any provision of functionality required by other agents to perform their tasks. Aspect-oriented modeling plays an important role in conflict resolution.

The first step in aspect-oriented modeling is identification of the concerns. As far as the concerns are identified the next step is specification of the concerns. The specification of concerns includes a set of parameters such as Name of the concern, Description of the concern, Sources of the information of concern, Classification means assigning the type to the concern that either it is emerged from functional requirement or non-functional requirement, Stakeholders who are linked with the concern, Responsibilities that mean what functionality it will have to provide, Contributions is the positive or negative interaction and Required Concerns that mean which other concerns are mandatory to perform its responsibilities. After specification, the concerns are composed. The concerns are combined and finding a match point which is a composition rule and this composition is continued till we find whole system. Composition rule starts and ends with a term tags. The composition operators are:

3.5.1. Enabling

The enabling inference of two agents is represented by $C1 >> C2$ which show that the

process$C2$ starts after the completion of the process $C1$. In composition of system, we use this symbol to show the relation between two agents that are sequentially interlinked.

3.5.2. Disabling

The disabling inference of two agents is represented by $C1$ [> $C2$, means $C1$ is interrupted by $C2$. In composition we use this symbol to represent the relation of two conflicting agents. If this relation exists between two agents then there should not be a direct link between those two agents.

3.5.3. Full synchronization

The full synchronization inference of two agents is represented by $C1||C2$, means both processes should execute parallel in synchronization. In composition we use this symbol to represent relation of two agents that have a direct link and execute in parallel.

3.5.4. Pure interleaving

The pure interleaving inference of two agents is represented by$C1|||C2$, means that the processes $C1$ and $C2$ can execute in parallel and if both the $C1$ and $C2$ are in ready state then either of them can execute first. In composition this symbol is used to represent relation of two agents that can execute in parallel as well as in sequence.

3.5.5. Direct link

The direct link of two agents is represented $C1 - C2$. In composition the direct link represents relation of two agents that directly interlinked. The expression $C1 - C2 - C3 - C4$ shows that $C1$ *and* $C2$, $C2$ *and* $C3$ also, $C3$ *and* $C4$ are directly linked but $C1$ is not directly linked to $C3$ or $C4$ in this expression.

4. Modeling of service-oriented IoT system

We take the scenario described in our previous paper [11]. The specifications of the system are:

- Higher Education Commission (HEC) a governing body at the top. HEC can add or delete Universities.
- University can add or remove resources. University can publish resources or call for resources. University can provide feedback after the utilization of resources.
- Universities can search for Resources in the registry. Resources may be binded to the Requester/Consumer. When the resources are rented out, there exist Contracts for service against that. Payment system for universities and billing of services.

Emerged Requirements are Log-in, Log-out, University Management by HEC, Resource Management by Universities and Security. Resources may be different smart devices such as Smart-parking, Trash collecting vehicles, University buses, Smart dustbins, Air Quality Equipments

and Cloud Storage. The resources may be composed in sub-systems such as Smart garbage monitoring and collection system, Smart parking system, Smart transportation system and Air quality control system.

4.1. Decomposition of the system for agents' identification

In the Inception phase first thing is to identify different components of system. Based on the requirements we draw an abstract representation of our system as shown in Figure 3. In this diagram we decompose the system in three parts. The first is application and business process connected to resources. The second one shows resources as subsystems and the third is aggregator for connection of resources with application. These subsystems include Smart Parking, Smart Buses, Air Quality Control and Smart Garbage Monitoring and Collection. So, we model this system as per decomposition to subsystems i.e. Agent *A*, Agent *B* and Agent *C*. Clearly, this is a heterogeneous system with multiple agents, multiple aspects and, multiple consumers and providers. We provide with the help of our framework a simplification in the form of an aggregator which helps us in understanding, identification, communication and composition.

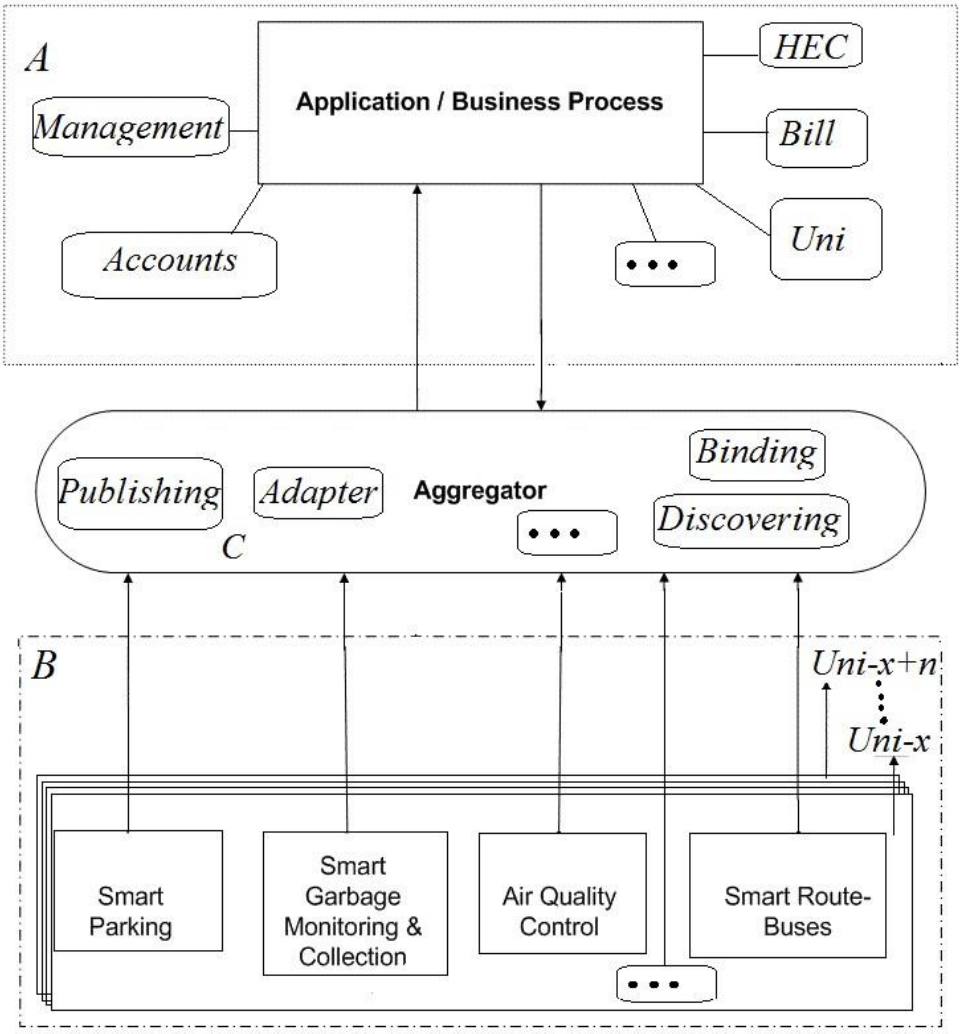

Figure 3. Decomposing the system into three parts.

4.2. Modeling for elaboration of agent A

In Elaboration phase we elaborate the identified agents. We treat agent A as a system composed of several agents. Firstly, we have to identify the agents. The agents in this system are HEC, University, Resource, Accounts, Bill and Interface. We use contract-based modeling for this system. We used Attributed Graph Grammar (AGG) for contracting based modeling of this system. Figure 4 shows a visual-contract in upper part and a type graph of Agent *A* modeled in AGG using contract-based modeling. The visual-contract has pre-condition on Left Hand Side (LHS) and post-condition on Right Hand Side (RHS). When the model in AGG is executed a start graph is generated. Figure 5 shows the start graph of this system. Figure 6 shows the use of XML for information modeling of a system *A*. XML may also be useful for modeling the system in hierarchical order.

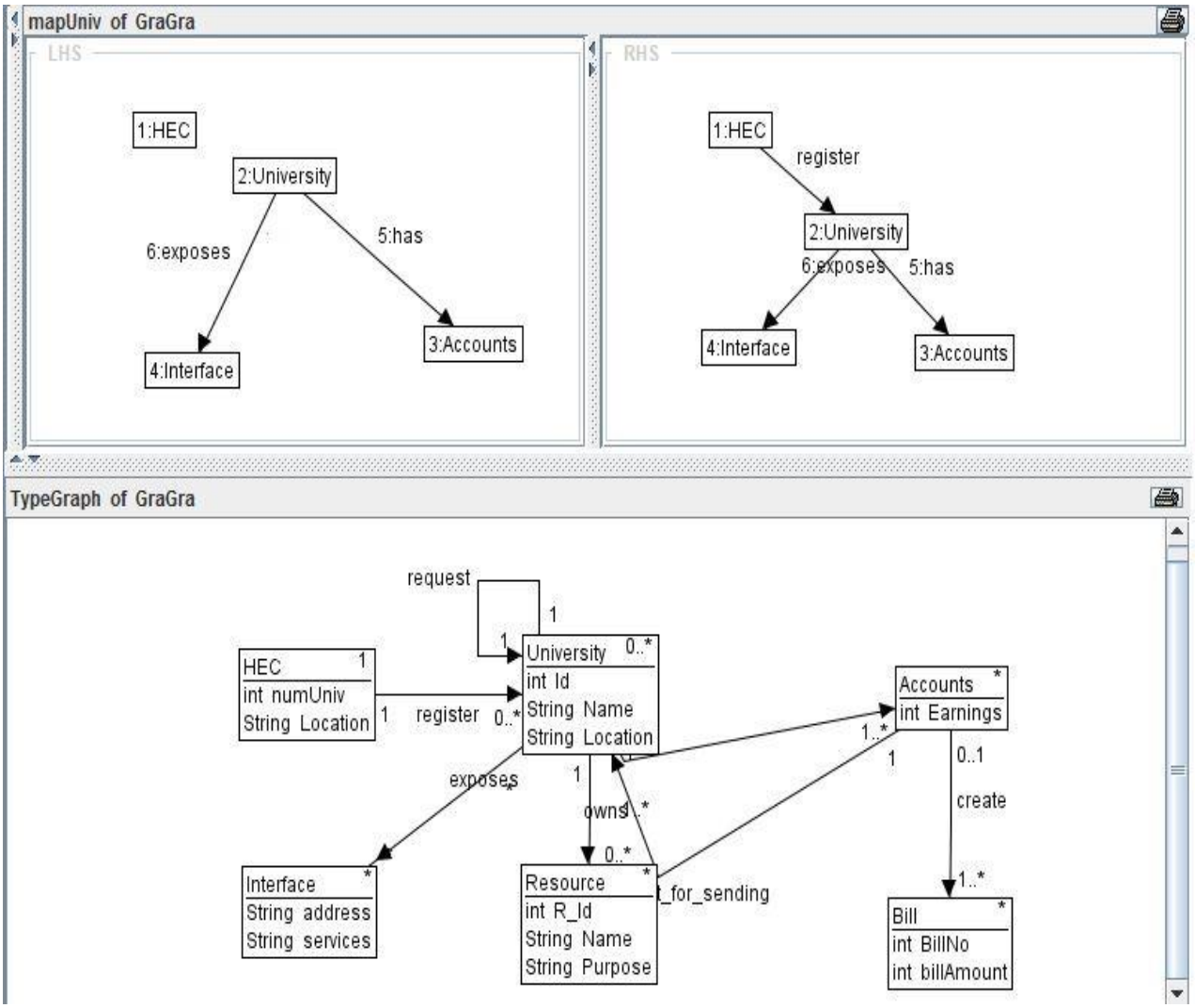

Figure 4. A contract and Type-graph of the system modeled in AGG.

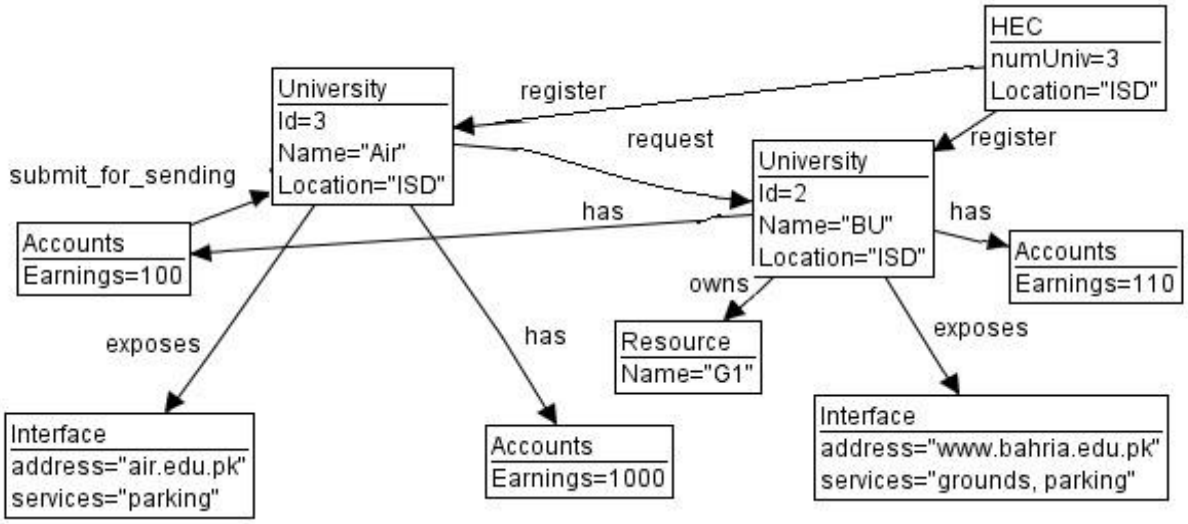

Figure 5. Start-graph of the system modeled in AGG.

```
▼<HEC>
  ▼<University category="">
     <Name lang="en"/>
     <Location/>
    ▼<Resource>
       <addResource Name="" Description=""> </addResource>
       <deleteResouce Name="" Identity=""> </deleteResouce>
       <freeResource Name="" Price=""> </freeResource>
     </Resource>
    ▼<accountManagement>
       <addAccount name="" Identity="" Role="" Password=""> </addAccount>
       <deleteAccount Identity=""> </deleteAccount>
     </accountManagement>
     <contract> </contract>
   </University>
  ▼<bill>
     <createBill billNo="" billAmount="" SenderId="" ReceiverId=""/>
     <payBill billNo="" billAmount="" SenderId="" ReceiverId=""/>
   </bill>
  ▼<account>
     <addAccount name="" Identity="" Role="" Password=""> </addAccount>
     <deleteAccount Identity=""> </deleteAccount>
   </account>
 </HEC>
```

Figure 6. XML representation for information modeling of system.

4.3. Modeling for elaboration of agent B

Here we are elaborating agent *B*. The agent *B* may be composed of different subsystems. These subsystems contain different devices and provide different functionalists. Here, we are considering four subsystems. These subsystems include Smart parking system, Smart Garbage monitoring and collection system, smart air quality control system and smart route bus system.

4.3.1. Elaboration of smart parking system

In next level of celaboration we elaborate the agents identified in elaboration of agent *B*. Smart parking system involves various sensors, display screens, signal devices, storage and application. Figure 7(a), shows work-flow of smart parking system. Here VDS stands for vehicle detection sensor. Space means vacant positions. Side means the left, right or forward indication on screen where there are multiple paths. These vacant positions are displayed usually on Light Emitting Diodes screen. OHD stands for over head indicators which are used to identify that the parking slot is empty or full. Cloud storage represents a service where data obtained from these processes is stored. One state change means that either a parking slot is filled by a vehicle or a vehicle leaves the parking slot. Filled means that vehicle covers the parking slot and vacant means that vehicle leaves the parking slot. Update means that data is stored on cloud storage. Figure 8 shows the description of different devices of smart parking system as services. It starts with description of vehicle detection sensor. Then over head display has been described. In last the WSDL describes LED direction screen. VDS only sends a message and OHD displays a message. LED direction screen receives message from VDS and provides direction as output. Figure 9 shows the WSDL representation of cloud storage.

4.3.2. Elaboration of smart garbage monitoring and collecting system

Smart garbage monitoring and collecting system is used to make the dustbins smart. These smart dustbins should be able to ask for service to vacate them. For the purpose of vacation there may be vehicles for outside placed and persons for in-building placed dustbins. Figure 7(b), shows the workflow of smart garbage monitoring and collection system. Here, SDB represents smart dustbin and on state change means dustbin changes from empty to half-filled, or half-filled to full. In-building means that the dustbin is placed inside the building where vehicle can't reach and somebody is assigned duty to collect the trash. CP means that the person to whom duty is assigned. CV means trash collecting vehicle. Accept means that one who accepts the duty of collecting trash. Complete means that the trash has been collected. Notify means to notify others who have received the request about the assignment of the duty and send the data to cloud.

4.3.3. Composition of Smart Garbage Monitoring and Collecting System

We consider *S DB*, *CP*, *CV* and *CloudStorage* as agents. While modeling this system using our proposed framework after modeling agents as services as described in Figure 8, we compose the system using Aspect-oriented composition rules. So, the composition rules will be as following:

1. *S DB >> CP*
2. *S DB >> CV*
3. *CV >> CloudStorage*
4. *CP >> CloudStorage*

According to the rules above, there are two paths that connect elements from *S DB* to *CloudStorage*. First one is *S DB−CP−CloudStorage* and second one is *S DB−CV −CloudStorage*. So, every agent is connected to atleast one other agent and there doesn't exist any conflict.

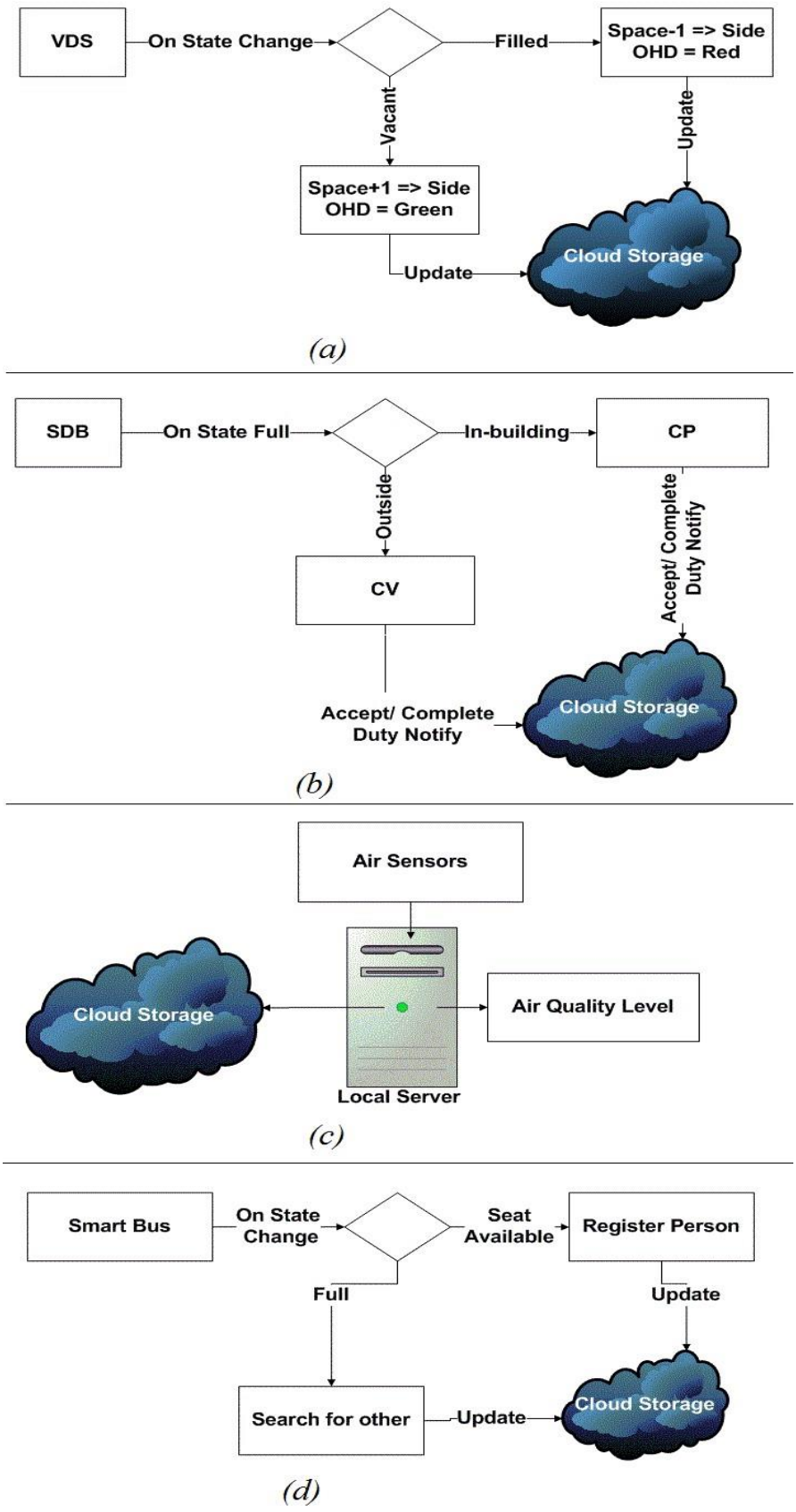

Figure 7. Work-flow Diagrams (a) Smart Parking (b) Smart Garbage Monitoring and Collection (c) Smart Air Quality Monitoring System (d) Smart-Route Bus System.

```
<Service-Discription-Language>
  <Vehicle-Detection-Sensor>
    <message name="SlotValue">
      <part name="SlotId" type="xs:string"/>
      <part name="Value" type="xs:string"/>
    </message>
    <portType name="VDS">
      <operation name="stateChange">
        <output name="SlotId" message="value"/>
      </operation>
    </portType>
  </Vehicle-Detection-Sensor>
  <Over-Head-Display>
    <message name="SlotValue">
      <part name="SlotId" type="xs:string"/>
      <part name="Value" type="xs:string"/>
    </message>
    <portType name="OHD">
      <operation name="stateChange">
        <input name="SlotId" message="value"/>
      </operation>
    </portType>
  </Over-Head-Display>
  <LED-Directional-Screen>
    <message name="SlotValue">
      <part name="SlotId" type="xs:string"/>
      <part name="Value" type="xs:string"/>
    </message>
    <message name="space">
      <part name="spaceSide" type="xs:string"/>
      <part name="Value" type="xs:string"/>
    </message>
    <portType name="LDS">
      <operation name="stateChange">
        <input name="SlotId" message="value"/>
        <output name="spaceSide" message="value"/>
      </operation>
    </portType>
  </LED-Directional-Screen>
</Service-Discription-Language>
```

Figure 8. WSDL used to show parking sensors service.

```
<Cloud-Storage>
  <message name="System-Id">
    <part name="SystemId" type="xs:string"/>
    <part name="Value" type="xs:string"/>
  </message>
  <portType name="storage">
    <operation name="Store">
      <input name="SystemId" message="value"/>
    </operation>
  </portType>
  <message name="System-Id">
    <part name="SystemId" type="xs:string"/>
    <part name="Value" type="xs:string"/>
  </message>
  <message name="Reply">
    <part name="RetId" type="xs:string"/>
    <part name="Value" type="xs:string"/>
  </message>
  <portType name="storage">
    <operation name="Retrieve">
      <input name="SystemId" message="value"/>
      <output name="RetID" message="value"/>
    </operation>
  </portType>
</Cloud-Storage>
```

Figure 9. WSDL of cloud storage service.

4.3.4. Elaboration of smart air quality monitoring system

Smart air quality monitoring system uses sensors to detect the level of harmful gases and oxygen in the air. Different sensors are installed at different positions and then accumulated air quality is measured. Figure 7(c), shows work-flow diagram of an air quality monitoring system. In this system the air sensors send information to local server and the local server processes the information. After processing the information, the data is stored in the cloud and also, the information is displayed to the people.

Due to the limitation of space, we are not repeating the service-oriented, contract-based and aspect-oriented approach for this sub-subsection and the next one.

4.3.5. Elaboration of Smart Route Bus System

Smart route bus system helps the registered students to track the bus and time to reach the desired stop. Hence, the work-flow of the registration system of the bus is shown in Figure 7(d). Smart Bus notifies when a registered student leaves or a new student is registered. So if after the notification there are number of registered persons is less than total capacity of the bus than new person can be registered. In other condition where the bus is already full someone interested to register on specific route will search for other buses. This information is updated on cloud storage.

4.4. An Ad-hoc and flexible representation of system

Here, the system is elaborated with the help of simulation tools and flexibility of change is provided. While modeling the system there may be a need of an abstract level model. This abstract level, Ad-hoc and continuous time models of the system or subsystems and devices may be developed using Agent-based modeling. Figure 10 shows an Ad-hoc and abstract representation of the system in the form of simulation implemented in Netlogo. This simulation creates a turtle HEC in the setup. Then universities are attached to the system and the number of universities may be selected from the slide-bar. The slide-bar provides flexibility in selecting the number of universities. In next step resources are added and randomly attached to the universities. Here we create two resources per university at this step. In the last step "Run" button may be used to change the links to resources to the universities i.e. showing service provision.

4.5. Composition of system using composition rules

In the composition phase we compose the systems from the elaborated subsystems and identify new agents if required. The last phase of our framework is associating the composition rules. However, what we find from modeling of agent *B* after decomposition is that *CloudStorage* is common in multiple subsystems. Thus, it's better to treat the *CloudStorage* as a fourth agent with the previous three i.e. agent *A*, agent *B* and agent *C*. Now, the relation between these four agents will be:

1. $A||C$
2. $B||C$
3. $CloudStorage||C$

The connection of the agents is $A - C - B$, $A - C - CloudStorage$, $B - C - CloudStorage$ that A is directly connected or has direct link with B and B is connected with $CloudStorage$ through aggregator C. It shows that all the agents are connected.

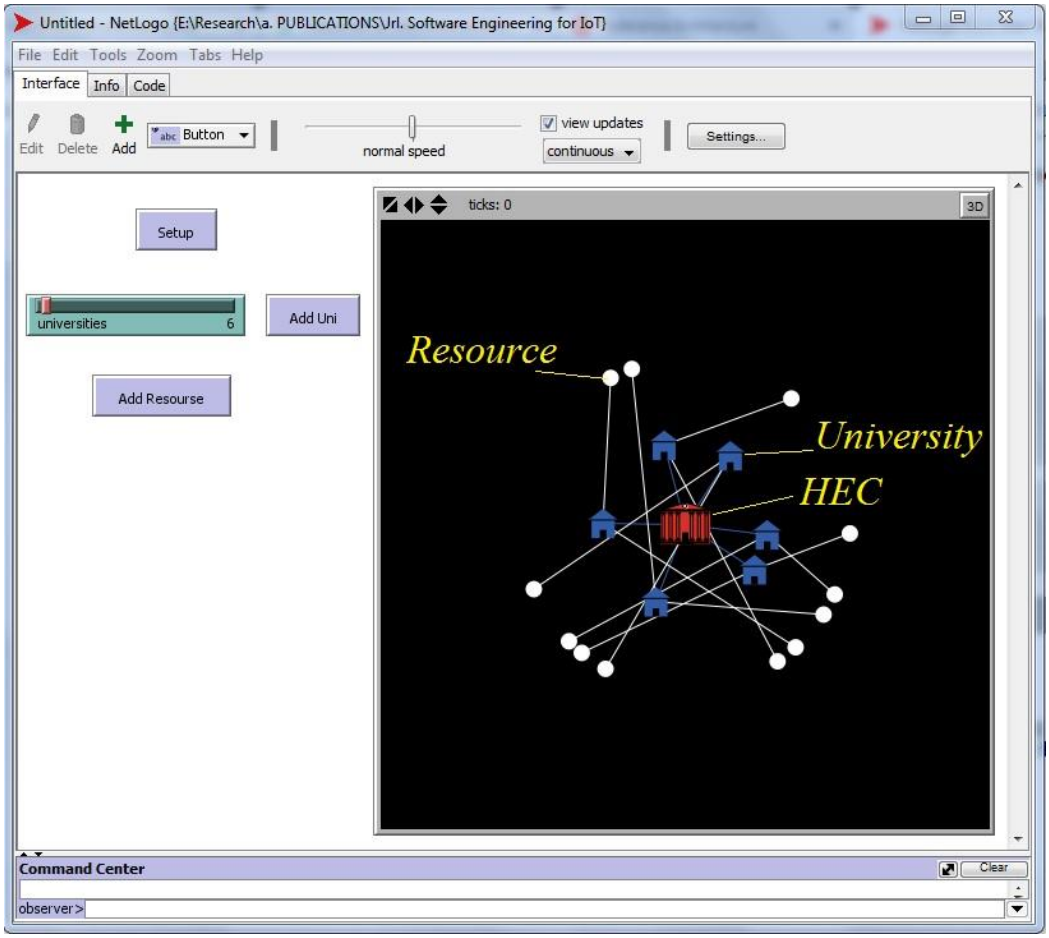

Figure 10. Netlogo model showing an Ad-hoc model where number of universities is flexible and can be changed.

5. Analysis and discussion

Reference architectures are abstract architectures of systems and they may differ from actual system [7]. IoT-A reference architecture provided a way to split an IoT system in different components. This architecture provided domain model, information model, functional model and communication model. The functional model of IoT-A reference architecture includes different layers such as device, communication, security, management, service organization, service, virtual entity, business process and application. While analyzing these layers it seems that layers such as application, business process, virtual entity, service organization and security involve major role of Software Engineer during design and development.

The industrial internet reference architecture (IIRA) focuses on internet of things with respect to industrial aspect [8]. It takes in account four viewpoints i.e. Business, Usage, Functional and Implementation Viewpoint. It also, discusses the crosscutting concerns and characteristics of the

system. The functional viewpoint has seven different domains such as Controls, Operations, Information, Application, Business, Functional and Crosscutting functions. The control domain contains the functions related to control systems. The operation domain functions include management, monitoring and optimization. The information domain includes data related functions. Application domain is same as application layer of IoT-A reference architecture which contains application logic. Business domain has almost similar function as business process layer of IoT-A reference architecture. Crosscutting domain includes the function like connection. Functional domain contains the technologies which support IoT systems such as cloud computing.

Reference Architecture Model for Industrie 4.0 (RAMI 4.0)) is also reference architecture for industrial internet of things [9]. This architecture is based on service-oriented concepts and has three dimensions i.e. Layers, Life cycle & Value Stream and Hierarchy Levels. The layers of RAMI 4.0 are Business, Functional, Information, Communication, Integration and Assets. The Business layer of RAMI 4.0 is similar to the business process layer of IoT-A where business models are mapped with processes. Functional layer contains the rules and decision-making logic and, services are also modeled at this layer. The information layer contains data to represent models based on formally described rules and execution of event related rules. Communication layer is similar as in IoT-A reference architecture. Integration layer provides interaction with human as well as interaction of devices with each other. Asset layer is similar to device layer of IoT-A reference architecture where there are physical entities. Life cycle & Value Stream and Hierarchy Levels are about the industry processes.

In [56], Service-oriented architecture has been used in-combination to SoaML for solving the heterogeneity issues. In [57], SysML has been used for modeling internet of things applications and to deal with system engineering problem. In [10], IoT-A reference architecture, model driven architecture and separation of concerns have been used in-combination for proposed framework. The framework is effective for Quality of Service attributes management in early stages of modeling. Both horizontal and vertical perspectives have been considered by using the principle of Separation of Concerns.

Service-oriented approach has been adopted in all reference architectures and also in above cited articles. The principle of Separation of Concerns has been used in IIRA as well as in [10]. Our framework uses four modeling approaches i.e. Service-oriented modeling, aspect-oriented modeling, agent-based modeling and contract-based modeling. When services are provided, there exists an agreement between service provider and service consumer. The agreement is a set of assumptions and guarantees. Contract-based modeling provides us a way to model system with pre-conditions or assumptions and post-conditions or guarantees. In internet of things there are smarter devices on nodes which are autonomous as well. Autonomous and smarter devices can be treated as agents. Agent-based modeling is useful while modeling random behaviors and Ad-hoc systems. Hence, we used these four modeling approaches in combination.

In software engineering the domain model is used to represent context information. Agent-based model can be used for the representation of domain model. Our framework can be used for domain models of IoT systems as shown in case study. Contracts can be used for modeling of specification since our framework provides a mechanism for specification modeling as well. The modularization and composition of system is a part of design model. Also, the attributed graph grammar (AGG) is used to represent the design model. Service-oriented modeling is also a part of design modeling. Hence, our framework provides a mechanism for modeling IoT based software system from

different aspects.

5.1. WSDL to TSDL

For service description of things, we need a standard language. The new language can be an extension of web service description language. Here we provide a possible extension of a WSDL for Internet of Things which we name Things Service Description Language (TSDL).

Elements for TSDL (< *message* >)

< *messagename* = ”*request− for−the− service*” >

< *partname* = ”*term*”*type* = ”*xs* : *string*”/ >

< /*message* >< *messagename* = ”*location − of − thing*” >

< *partname* = ”*value*”*type* = ”*xs* : *string*”/ >

< /*message* >< *messagename* = ”*communication − response − for − the − service*” >

< *partname* = ”*value*”*type* = ”*xs* : *string*”/ >

< /*message* >

The element < *message* > defines the name of all messages and data types used by these messages

Elements for TSDL (< *portType* >)

< *portTypename* = ”*Thing − service − name − or − thing − id*” >

< *operationname* = ”*ThingOperation*” >

< *requestmessage* = ”*request− for−the−service*”/ >

< *phyResponsemessage* = ”*action − perform − by − thing*”/ >

< *locationmessage* = ”*location − of − thing*”/ >

< *comResponsemessage* = ”*communication − response − for − the − service*”/ >

< /*operation* >

< /*portType* >

In *portType* element name will be the thing service name or service name for example garbage collection service is the name of the service for garbage collection in city. *ThingOperation* is the name of operation that defines by *portType* element. For example, collecting garbage is the operation of smart garbage collection system. Through < *request* > element proposed to use for request for the service of things for example someone request to smart garbage system to collect garbage from my house. < *phyResponse* > element proposed for action performance after receiving request for service message for example garbage collection van driver trace your location and move towards home where someone request. < *location* > of thing is the very important element for IoT system because on the base of location of both requester and responder action will be performed. < *comResponse* >

element proposed for the communication response from thing to requester after performing physical response. There are four types of operations in WSDL that are:

1. One-way
2. Request-response
3. Solicit-response
4. Notification

But in case of IoT there is physical existence of things therefore, we defined an extra element < *phyResponse* > within < *operation* > element with < *request* > element and < *comResponse* > element.

5.2. SOAP to SoTAP

To access services provided by things, Simple object access protocol (SOAP) also needs to be modified. Here we provide possible extension.

Envelope for SoTAP

< *sotap* : *Envelopexmlns* : *sotap* = "*asperpublication*/"*sotap* : *encodingStyle* = "*asperprovide*" > ... < /*soap* : *Envelope* >

Header for SoTAP

< *sotap* : *Header* >

< *tid* : *ThingID* > *xmlns* : *tid* = "*thingid*/"*sotap* : *mustUnderstand* = "1" > 234 < /*tid* : *ThinID* >

< *tow* : *Thingowner* > *xmlns* : *tow* = "*owner* − *name*/" < /*tow* : *Thingowner* >

< *tre* : *Thingrequester* > *xmlns* : *tre* = "*requesrer* − *name*/*id*/" < /*tre* : *Thingrequester* >

< /*soap* : *Header* >

Body for SoTAP

< *sotap* : *Body* >

< *m* : *Getservice* − *namexmlns* : *m* = "*thing* − *id*/*service* − *repository*" >

< *m* : *services* > *request* − *service* < /*m* : *service* >

< /*m* : *Getservice* − *name* >

< /*sotap* : *Body* >

6. Conclusion and future work

Internet of Things is an emerging area connecting various domains and aimed in targeting almost every aspect of life. Everything is going to be connected to the internet and everything is expected to be smart in near future. These smart systems are highly dependent on artificial intelligence and

connections. The increasing number of connected devices and their collaboration is demanding more and more complex systems. These complex systems require proper planning and guidelines for development. New scenarios are emerging on daily basis. Due to these reasons, Internet of Things systems development need complete and correct models. However, previous modeling approaches may not be sufficient to effectively model such systems. In this paper we have proposed a framework for complex IoT systems modeling for Software Engineering purpose. Our framework provides a way to model different types of systems and subsystems i.e. continuous, discrete, Ad-hoc, general, service-oriented systems including decomposition and composition. We have used well known elements of four modeling approaches i.e. agent-based modeling, aspect-oriented modeling, contract-based modeling and service-oriented modeling. The framework provides a mechanism to represent domain model, specification and design model of service-oriented internet of things systems. We validated the framework by using it for an IoT system scenario. We used different tools like Netlogo, Attributed graph grammar (AGG), XML and WSDL.

However, in future we aim on working for a unified tool based on this framework. We used WSDL for devices by using device as a portType. There is a need to customize WSDL to service description language for devices and provide alternate names to the tags. Similarly, SOAP needs to be updated as per IoT because here a physical object provides service. We provided an overview of TSDL and SoTAP in previous section but detailed study needs to be conducted for this purpose. Also, binding of simple object in web service differs from binding of a physical object in IoT service.

Conflict of interest

The authors have no conflict of interest.

References

1. Q. Jing, A. V. Vasilakos, J. Wan, Security of the internet of things perspectives and challenges, *Wirel. Netw.*, **20** (2014), 2481–2501.
2. Z. Sheng, S. Yangy, Y. Yuz, A. V. Vasilakos, J. A. McCanny, K. K. Leungy, A survey on the ietf protocol suite for the internet of things standards challenges and opportunities, *IEEE Wirel. Commun.*, **20** (2013), 91–98.
3. Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, S. W. Kim, The future of healthcare internet of things a survey of emerging technologies, *IEEE Commun. Surv. Tutor.*, **22** (2020), 1121–1167.
4. Z. Yan, P. Zhang, A. V. Vasilakos, A survey on trust management for internet of things, *J. Netw. Comput.*, **42** (2014), 120–134.
5. B. Bera, S. Saha, A. K. Das, A. V. Vasilakos, Designing blockchain based access control protocol in iot enabled smart grid system, *IEEE Int. Things J.*, **8** (2021), 5744–5761.
6. S. Papert, Mindstorms Children, Computers, and Powerful Ideas, in: *Basic Books Inc*, (1980).
7. A. Nettstrater, Internet of things architecture IoT-A deliverable D1.3 updated reference model for IoT v1 5, *Tech. Rep., FHGIML*, **7** (2012). URL: https://cocoa.ethz.ch/downloads/2014/01
8. S.-W. Lin, M. Crawford, S. Mellor, The industrial internet of things volume G1: Reference architecture, *Tech. Rep.*, Industrial Internet Consortium (2017). URL: https://www.iiconsortium.org

9. J. Gayko, Alignment report for reference architectural model for industrie 4.0/intelligent manufacturing system architecture, *Tech. Rep.,* Sino-German Industrie 4.0/Intelligent Manufacturing Standardisation SubWorking Group (2018). URL: www.plattform-i40.de/PI40/Redaktion/EN/Downloads
10. M. P. Alves, F. C. Delicato, P. F. Pires, IoTa-md: A model-driven approach for applying QoS attributes in the development of the IoT systems, in: *Proc. ACM Symp. Appl. Comput.,* (2017), 1773–1780.
11. K. M. Abbasi, T. A. Khan, I. U. Haq, Hierarchical modeling of complex internet of things systems using conceptual modeling approaches, *IEEE Access,* **7** (2019), 102772–102791.
12. R. Dijkman, B. Sprenkels, T. Peetersa, A. Janssen, Business models for the internet of things, *Int. J. Inf. Manage,* **35** (2015), 672–678.
13. D. Miorandi, S. Sicari, F. D. Pellegrini, I. Chlamtac, Internet of things vision applications and research challenges, *Ad. Hoc. Netw.*, **10** (2012), 1497–1516.
14. X. Jia, J. Wang, Q. He, IoT business model and extended technical requirements, in: In *Proc. ICCTA2011*, 2011.
15. S. A. Alvi, B. Afzal, G. A. Shah, L. Atzori, W. Mahmood, Internet of multimedia things vision and challenges, *Ad Hoc Netw.*, **33** (2015), 87–111.
16. J. Kiljander, A. D'elia, F. Morandi, P. Hyttinen, J. T. Mattila, A. Y. Oja, et al., Semantic interoperability architecture for pervasive computing and internet of things, *IEEE Access,* **2** (2014), 856–873.
17. L. Guijarro, V. Pla, J. R. Vidal, M. Naldi, Maximum profit two sided pricing in service platforms based on wireless sensor networks, *IEEE Wireless Commun. Lett.*, **5** (2016), 8–11.
18. A. Ghanbari, A. Laya, J. Alonso-Zarate, J. Markendahl, Business development in the internet of things a matter of vertical cooperation, *IEEE Commun. Mag.*, **55** (2017), 135–141.
19. K. Kang, Z. Pang, L. D. Xu, L. Ma, C. Wang, An interactive trust model for application market of the internet of things, *IEEE Trans. Industr. Inform.,* **10** (2014), 1516–1526.
20. R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed internet of things, *Comput. Netw.*, **57** (2013), 2266–2279.
21. H. Van Dyke Parunak, R. Savit, R. L. Riolo, Agent based modeling VS equation based modeling: A case study and users guide, in: *International workshop on multi-agent systems and agent-based simulation Springer Berlin Heidelberg*, (1998).
22. C. M. Macal, M. J. North, Tutorial on agent based modeling and simulation, in: *37th WSC 2005: Orlando, FL, USA.*, (2005).
23. G. I. Hawe, G. Coates, D. T. Wilson, R. S. Crouch, Agent based simulation for large-scale emergency response a survey of usage and implementation, *ACM Comput. Surv.*, **45** (2012), 1–51.
24. S. Vasanthapriyan, S. Thuseethan, Prediction of human flow in disaster situations a multi agent-based modelling and simulation, in: *2nd International Symposium on DCIT Wuhan*, (2015).
25. W. N. Robinson, Y. Ding, A survey of customization support in agent-based business process simulation tools, *ACM Trans. Model. Comput. Simul.,* **20** (2010), 1–29.
26. J. A. Paravantis, From game theory to complexity science and agent-based modeling in world politics, *6th Int. Conf. Inf. Intell. Syst. Appl. IISA*, (2015).
27. M. Richiardi, The future of agent based modelling, *Eastern Econ. J.,* **43** (2017), 271-287.
28. G. Bruno, A. Genovese, A. Sgalambro, An agent-based framework for modeling and solving location problems, **18** (2010), 81–96.

29. T. D. Cao, T. V. Pham, Q. H. Vu, H. L. Truong, D. H. Le, S. Dustdar, Marsa a marketplace for realtime human sensing data, *ACM Trans. Internet Technol.*, **16** (2016), 1–21.
30. B. Chen, J. Wang, I. J. Cox, M. S. Kankanhalli, Multi keyword multi click advertisement option contracts for sponsored search, *ACM Trans. Intell. Syst. Technol*., **7** (2015), 1–29.
31. A. Y. Du, S. Das, R. D. Gopal, R. Ramesh, Risk hedging in storage grid markets do options add value to forwards, *ACM Trans. Manag. Inf. Syst*., **2** (2011), 1–23.
32. X. Zhang, T. Ward, S. Mcloone, Comparison of predictive contract mechanisms from an information theory perspective, *ACM Trans. Mult. Comput. Commun.*, **8** (2012), 1–18.
33. T. S. Strickland, C. Dimoulas, A. Takikawa, M. Felleisen, Contracts for first class classes, *ACM Trans. Program. Lang.*, **35** (2013), 1–58.
34. C. Dimoulas, M. Felleisen, On contract satisfaction in a higher order world, *ACM Trans. Program. Lang.*, **33** (2011), 1–29.
35. J. Yi, D. Qi, S. H. Tan, A. Roychoudhury, Software change contracts, *ACM Trans. Softw. Eng. Methodol.*, **24** (2015), 1–43.
36. G. Castagna, N. Gesbert, L. Padovani, A theory of contracts for web services, *ACM Trans. Program. Lang.*, **31** (2009), 1–61.
37. T. T. H. Le, R. Passerone, U. Fahrenberg, A. Legay, Contract based requirement modularization via synthesis of correct decompositions, *ACM Trans. Embed. Comput. Syst.*, **15** (2016), 1–26.
38. R. Barga, D. Lomet, G. Shegalov, G. Weikum, Recovery guarantees for internet applications, *ACM Trans. Internet Technol.*, **4** (2004), 289–328.
39. V. Ungureanu, Using certified policies to regulate e commerce transactions, *ACM Trans. Internet Technol.*, **5** (2005), 129–153.
40. I. Dragomir, I. Ober, C. Percebois, Contract based modeling and verification of timed safety requirements within sysml, *Softw. Syst. Model*, **16** (2015), 687–624.
41. K. Christidis, M. Devetsikiotis, Blockchains and smart contracts for the internet of things, *IEEE Access*, **4** (2016), 2292–2303.
42. C. K. Frantz, M. Nowostawski, From institutions to code towards automated generation of smart contracts, in: *IEEE 1st International Workshops on FAS*W*, (2016).
43. W. Retschitzegger, W. Schwinger, E. Kapsammer, A survey on UML based aspect-oriented design modeling, *ACM Comput. Surv.*, **43** (2011), 1–59.
44. C. Chavez, A. Garcia, U. Kulesza, C. SantAnna, C. Lucena, Crosscutting interfaces for aspect-oriented modeling, *J. Brazilian Comput. Soc.*, **12** (2006), 43–58.
45. J. Fox, A formal foundation for aspect-oriented software development, in: *Research on Computing Science CIC*, (2005), 1665–9899.
46. J. Zhang, Y. Chen, Y. Zhang, H. Li, Aspect oriented modeling and mapping driven by model driven architecture, in: *2nd IEEE ICCSIT*, (2009), 180–184.
47. J. Liu, L. Zhang, QOS modeling for cyber-physical systems using aspect-oriented approach, in: *Second International Conference on Networking and Distributed Computing (ICNDC)*, (2011), 154–158.
48. A. Wasicek, P. Derler, E. A. Lee, Aspect oriented modeling of attacks in automotive cyber physical systems, in: *ACM DAC 14 San Francisco CA* (USA, 2014), 1–6.
49. T. Cerny, Aspect oriented challenges in system integration with microservices SOA and IoT, *Enterp. Inf. Syst.*, **13** (2019), 467–489.

50. P. Vrba, V. Marik, P. Siano, P. Leitao, G. Zhabelova, V. Vyatkin, et al., A review of agent and service-oriented concepts applied to intelligent energy systems, *IEEE Trans. Industr. Inform.*, **10** (2014), 1890–1903.
51. R. K. Behera, K. H. K. Reddy, D. S. Royb, Modeling and assessing reliability of service-oriented internet of things, *Int. J. Comput. Appl.*, **41** (2019), 195–206.
52. B. B. Traore, B. K. Foguem, F. Tangara, X. Desforges, Service-oriented computing for intelligent train maintenance, *Enterp. Inf. Syst.*, **13** (2019), 63–86.
53. I. L. Yen, F. Bastani, S. Y. Hwang, W. Zhu, G. Zhou, From software services to IoT services the modeling perspective, in: *International Conference on Serviceology*, (2017), 215–223.
54. F. Rademacher, S. Sachweh, A. Zundorf, Analysis of service-oriented modeling approaches for viewpoint specific model driven development of microservice architecture, *CoRRRademacher2018 abs/1804.09946* (2018).
55. T. A. Khan, R. Heckel, On model based regression testing of web-services using dependency analysis of visual contracts, in: *D. Giannakopoulou, F. Orejas (Eds.), Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg*, (2011), 341–355.
56. B. Costa, P. F. Pires, F. C. Delicato, Modeling SOA based IoT applications with soaml4iot, in: *IEEE 5th World Forum on Internet of Things WF-IoT*, (2019), 496–501.
57. B. Costa, P. F. Pires, F. C. Delicato, Modeling IoT applications with sysml4iot, in: *IEEE 42th EUROMICRO Conference on Software Engineering and Advanced Applications*, (2016), 157–164.

AIMS Press