Towards semantic representation of the IoT ecosystem and smart home applications

Alexey Andrushevich, Iosif Vojteshenko
Belarusian State University
Minsk, Belarus
Email: andrushevich@bsu.by

***Abstract*—This paper describes an original approach to building a cross-industry ecosystem of the Internet of Things and smart home applications through its semantic representation based on OSTIS technology. The results will improve the efficiency of the component approach to application development in the Internet of Things in the future, as well as enable automatic synchronization of different versions of components, increasing their compatibility and consistency.**

***Keywords*—Internet of things, smart home, semantic representation, multi-component model**

I. Introduction

Multi-agent and situational (contextual) processing has found wide application in Internet of Things applications, such as the smart home [1]. However, despite the significant progress in recent years in the development of sectoral communication systems and automated control systems, the following problems are still relevant:

- heterogeneity of standards and technologies for receiving, storing, processing, and transmitting data in IoT applications;
- limited to the industry domain functionality;
- low adaptability, interoperability and scalability of applications.

Thus, it is necessary to state that there is no unified comprehensive approach to the design of an ecosystem of inter-industry universal Internet of Things (IoT).

II. Analysis of existing solution approaches

It has to be noted that the concept of a ubiquitous, cross-industry, context-aware, secure Internet of Things is rapidly gaining popularity due to the rapidly developing convergence of technology packages, system functions, platforms and services. Vivid examples of such convergence mechanisms are the emergence and development of common system functions in the Internet of Things, such as basic context awareness services / services, geolocation and information security. Different branches of IoT applications also often implement and use similar functional blocks, including user authentication and authorization, geo-temporal localization, event processing, contextual awareness and many others. For example, the scientific community has already published [2]–[4] the development results of underlying technologies and algorithms to acquire and provide information about geo-temporal location and context. In fact, the vision of a ubiquitous, pervasive Internet of Things includes not only a multitude of technologies, but also a dynamic changing environment and a virtual information environment (space) of users. The collection and use of contextual information goes beyond conventional closed-loop applications that use only simple location information. For example, universal geo-temporal information is successfully modernizing IoT applications in industries such as home, office, industry, commerce, transportation, healthcare, and cities.

III. Proposed approach

This paper proposes to take as a basis the well-known components and libraries of OSTIS [5] technology to formalize the description of the subject domain of the Internet of Things ecosystem. The systems developed on the basis of OSTIS Technology are called *ostis-systems*. At the heart of *OSTIS Technology* is a universal way of semantic representation (coding) of information in the memory of intelligent computer systems, called as *SC-code*. In this way, a better convergence of heterogeneous standards and IoT technologies can be achieved, which will contribute to the cross-industry integration of industry-specific communication systems and automated control systems into a single ecosystem.

So, let's give the following definitions of the considered subject area of the Internet of Things in *SC-code*:

Internet of Things
:= [A set of physical objects connected to the Internet and exchanging data. The term "Internet of Things" was first introduced in 1999 by Kevin Ashton, an entrepreneur and co-founder of Auto-ID Labs (a distributed research group in radio-frequency identification and new sensor technologies) at the Massachusetts Institute of Technology (MIT).]
:= [The concept of a data network between physical objects ("things") equipped with built-in means and technologies for interacting with each other or with the external environment]

Web of Things

:= [A global network of automatically generated web pages that are automatically read by computing devices embedded in physical objects (parking lots, paving tiles, windows, doors, children's toys, dishes, clothes, soil, etc.). The Web of Things is characterized by convergence and integration with artificial intelligence technologies]

:= [W3C's approach to using web technologies in the Internet of Things to eliminate fragmentation in Internet of Things development standards]

Context-dependent information system

:= [An information system that uses the concept of context to provide relationally personalized dynamic information and / or services to the user, where the degree of relationship depends on the user's current environment, interests, objectives, intentions, and actions]

API with representative state transfer

:= [The concept of building a distributed application based on the client-server type, where each request (REST-request) of the client to the server contains comprehensive information about the desired response (representative state) of the server without the need to store information about the client state (client session)]

Based on the authors' many years of professional experience, it can be argued that the organization of a universal generic cross-industry (or "horizontal") method of technological implementation of system architecture and applications in the Internet of Things can be represented as three development streams, each dedicated to the implementation of one system platform of the Internet of Things. These 3 fundamental application-independent platforms together form a generic IoT architectural platform that integrates into an interconnected heterogeneous system of IoT subsystems, offering common functionality to application developers and leaving developers more time to focus on the business logic of their applications. The *Cloud Server Platform* has high computing resources and large amounts of memory, manages the entire IoT system to the greatest extent. *Mobile Platform* is responsible for presenting the resulting and useful information to users. The *Connected Object Platform*, integrates sensory information about the environment and is capable of performing simple actions through connected objects.

Let us describe the above platforms in more details:

- Smart Server platform: a set of different servers to provide API building blocks in the cloud for integration of applications and services. This platform has high computational power, memory resources and can interface with any standards. One can consider the computational resources and capabilities as unlimited. Its role is to collect, aggregate and / or interpret context-sensitive information from connected IoT application nodes to make it efficient and meaningful (customer value) for mobile device users. It also provides application developers with APIs of local or remote universal core components that implement server-side functionality for all applications, such as: big data processing and analytics, complex event processing, a localization engine, a context management framework, a user profile and user behavior model, security policies, access control (including user registration, authentication and trust schema). The most common examples of Smart Server platform are Yandex.Cloud, Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Cisco IoT Cloud Connect, SAP, Oracle IoT, ThingsBoard, SiteWhere, Predix IoT, Thinger.io;
- Smart Mobile platform: a reference platform for mobile applications that provides functionality on mobile devices for all IoT applications. Mobile devices are still most often owned by human end users. Common functions of mobile clients are: common user interface components, user profiling, authentication and authorization, information security (privacy), search and discovery of smart servers, communication with server components, receiving notifications from servers, "local" data mining algorithms (thick client). This platform provides IoT applications with the ability to use the contextual information they need without having to worry about how that contextual information is fetched. The mobile platform can communicate both with the Smart Server platform and directly with some connected IoT nodes. Finally, the mobile UI platform is responsible for the last stage of consumer value creation through the multimedia presentation / output of consistent and interesting information to users. The most common examples of the Smart Mobile platform are Zetta, HP Enterprise Universal, Carriots, ThingsBoard, ThingWorx, and Xively. To reduce the development time of mobile applications, so-called hybrid applications are often used, the code of which is adapted using frameworks for several mobile hardware platforms at once. The most popular frameworks are: PhoneGap, Rhodes, Appcelerator, Xamarin, Ionic, Appy Pie, Native Script.
- Smart Object platform: integration of various gateways, network hubs and related connected object technologies. Connected objects can perceive physical environmental data (temperature, pressure, light, humidity, vibration, etc.) through sensors or be actuators (switch, actuator, solenoid, etc.). Data transmission is usually organised through low-power communication standards. Such devices may also be equipped with a "lightweight" operating system.

They are very heterogeneous and can operate in very different ways. However, all implementation features of connected objects (things) must be hidden from other devices in order to provide a homogeneous way of communicating with other system components, regardless of what data they perceive or how they are arranged. Consequently, a single object software interface is a key to being able to easily integrate different gateways that provide access to different connected objects via the same standards, without requiring the designers of the UI application to know the complexity of the relationship between objects and the gateway and what functionality (data collection or actuation) is performed on those objects. This platform has common functions, such as: object detection, security, and management. The most common examples of the Smart Object platform are open source IoT Kaa, Arduino, Flutter, Qualcomm's IoT Development Kit, Particle.io, ESP8266, Intel Edison, Raspberry Pi, Beagle Bone.

IV. Multi-component model

This section is devoted to the theoretical definition of a universal multi-component model of a typical IoT application for the platforms from the previous subsection. A description for a "canonical" or "classic" Internet of Things application architecture consisting of cloud or server capacity, various data delivery protocols, user devices (PCs, laptops, tablets, phones, embedded user interfaces in any devices), any kind of data collection sensors and any "device-executor-hands". Taking into account the features and properties of the three parts of the horizontal platform (Smart Server, Smart Mobile and Smart Object), as well as the characteristics of Internet of Things applications, the following universal tree-like representation of a multi-component model of application architecture in the Internet of Things is proposed.

All the components and parameters of this hierarchical tree model are given as an example and are subject to change in specific IoT applications.

(0) Internet of Things application S = A * B * C

(1) Smart Server Platform A = D * E

(1.1) Access Control D = G * H * I

(1.1.1) Registration G: G1 (Users), G2 (Machines)

(1.1.2) Authentication H: H1 (Periodic), H2 (On Demand)

(1.1.3) Trust Scheme I: I1 (Application Managed), I2 (Policy Managed), I3 (Profile Managed), I4 (Risk Managed)

(1.2) Data Processing E = J * K * L

(1.2.1) Context management J: J1 (Event-driven), J2 (Polling)

(1.2.2) Flow processing K: K1 (Offline), K2 (Semi-Online), K3 (Online)

(1.2.3) Data storage L: L1 (Simple), L2 (Hierarchical), L3 (Structured), L4 (Dynamic)

(2) Smart Mobile Platform B = M * Phi

(2.1) General Mobility Characteristics M = O * P

(2.1.1) Basic data traffic type O: O1 (Multimedia), O2 (Perception / reading), O3 (Control), O4 (Well-balanced)

(2.1.2) Location information P: P1 (On / Off mode), P2 (Based on accuracy)

(2.2) User Interface Phi = R * F

(2.2.1) User Interface R: R1 (Single Media), R2 (Multimedia), R3 (Adaptive)

(2.2.2) Content delivery F: F1 (Service-based), F2 (Device-based)

(3) Smart Object Platform C = Q * T

(3.1) Common Node Characteristics Q = W * V * U

(3.1.1) Power Source W: W1 (Passive), W2 (Active)

(3.1.2) Physical Interaction V: V1 (Read), V2 (Action), V3 (Combined)

(3.1.3) Encryption U: U1 (Yes), U2 (No)

(3.2) Connectivity T = X * Y * Z

(3.2.1) Media X: X1 (Wireless), X2 (Wired)

(3.2.2) Network type Y: Y1 (Grid), Y2 (Point-to-Point)

(3.2.3) Configuration Z: Z1 (Adaptive), Z2 (Static)

Based on the proposed multi-component model of the Internet of Things ecosystem, such Internet of Things applications as smart home, office, industry, trade, transport, healthcare, and smart cities can be implemented.

Let us focus on the popular smart home application and describe its functions in more detail.

V. Examples of functional smart home tasks

Typical functional tasks implemented in the form of components / applications in the design and development of smart home software and hardware are [6]:

- task of access to the living quarters;
- task of monitoring lonely elderly people [7];
- task of controlling the lighting;
- task of energy management and energy efficiency [8].

Thus, the functional classification of smart home applications can be defined using the SC code as follows:

Smart home application

$:=$ [separate hardware and software combination that operates according to functional requirements]

$\Rightarrow$ *partitioning**:
Partitioning a class of smart home applications by functionality
= {
- *physical access control application*
  $\supset$ *monitor / control the status of all entrances and exits*
- *supervisory application for elderly*
  $\supset$ *domestic safety for elderly*
- *housing light control application*
  $\supset$ *monitor and control the natural and artificial light*
- *energy management and energy efficiency application*
  $\supset$ *monitor and control all resources and energy*

}

Let us describe in more details the functionality of the aforementioned smart home applications.

A. Housing access system

The key task of such a system is the identification of people who come near to the house entrance door and the correct processing of identification results: if the person who came to the door should have access to the house, the door opens automatically, in the opposite case the system offers to talk to the occupants of the house or apartment. In this case, it is desirable that the system is able to determine whether there is someone in the house. If no one is at home, the system must inform the visitor that he can not come in now.

In terms of devices, the system is equipped with a camera as well as lamps for lighting. The camera is triggered by the motion sensor, and the lights are also triggered if the environment is too dark for the camera to work. The system tries to recognize the visitor from the camera images.

In addition to the camera data, the system also has the residents' location data at its disposal. If no one is home according to the geo-location data, the system rejects visitors. There is also an option for the user to enable this mode manually. This can be useful, for example, if residents are not to be disturbed for a period of time.

The system should also contain a graphical user interface, through which one can both monitor selected indicators and the status of devices, and control the behavior of the system. In addition, through the user interface, residents can be notified when someone tries to enter the premises.

Thus, the following functional requirements for the system can be identified:

1) The system allows to set multiple resident profiles so that they can be identified on entrance.
2) If the identification is positive, the system opens the door automatically.
3) In the default state, the camera and lights are turned off, but they must be turned on as needed.
4) The system can determine whether or not anyone is in the house. If no one is home, the system should inform the visitor. If not, the system should offer to talk to people inside the house.
5) The system must also support the "Do Not Disturb" mode. The behavior of the system in this case corresponds to the case when no one is in the room.
6) The system must notify residents about visitors.

In addition to the functional requirements, the following non-functional requirements were also developed for the system:

1) The system can recognize up to 10 different resident profiles.
2) The system must respond correctly to device signals, even if there is a break in the connection.
3) Manual operation is also allowed, in particular the use of a key to open the door.
4) System extension with more devices is supported.

B. Surveillance System for Elderly People Living Alone

With the declining birth rate, the proportion of the elderly in society is increasing, while the proportion of people of working age is decreasing. Elderly people often need help with health care, free movement and in case of unwanted accidents. Through the use of Internet of Things technology, such people could live in greater safety and comfort. The field of the Internet of Things related to elderly supervision is also called AAL (ambient or active assisted living).

In [7] the following areas of application of IoT technologies were identified:

1. information assistance (easy accessibility of all necessary information),

2. intelligent situational behavior (the environment should recognize typical patterns of behavior and offer appropriate assistance),

3. prediction of undesirable events (recognition of such situations on the basis of behavioral and physiological indicators, and application of preventive measures),

4. recognizing and reacting to undesirable situations,

5. security (protection against intrusions using authorization and authentication mechanisms),

6. confidentiality (minimization of privacy interference).

To meet these needs, the system can rely on both historical and real-time data. Two groups of sensors are distinguished. Environmental data comes from environmental sensors, e.g., temperature, motion. Data on human behavior and condition comes from wearable sensors. The system may also include feedback devices for visual and voice notification of various events. State-of-the-art wireless technology allows for a reliable, easy-to-install, and inexpensive data communications infrastructure.

Central to AAL systems is the collection and storage of data on user behavior, highlighting patterns and identifying undesirable situations based on deviations from them. The user's state can be determined based on data from several sensors located in the dwelling [7].

Among the undesirable situations, let's especially emphasize falls. One way to detect falls is to use a wearable acceleration and atmospheric pressure sensor together. In the case of unexpected acceleration, the system reads the pressure data twice, on the basis of which it draws a conclusion about the position of the human body [7].

In terms of modeling such systems, three categories of indicators can be distinguished. The physical aspects include both environmental conditions and human health parameters. Due to the continuous nature of this category, the method of system dynamics combined with stochastic modeling to account for the role of randomness seems most natural. On the other hand, discrete-event modeling allows one to account for emerging events, such as falls. Finally, spontaneous user behavior is best modeled using the agent-based method.

C. System for dwelling light control

Approximate functional requirements for the application: When entering a room, the light comes on in response to movement and 10 seconds after leaving the room, the light turns off. The brightness of the light must be adjusted to the level of street light entering the dwelling. In addition, lighting from 11 p.m. to 6 a.m. shall operate in nightlight mode, with reduced brightness.

D. Energy consumption and efficiency management

Due to the growing energy crisis and the inadequacy of traditional centralized energy systems to new challenges, a new model has emerged: hybrid distributed energy production with a significant contribution of renewable energy sources. This model is also characterized by a bidirectional flow of information and electricity. There are solutions for all stages of energy production, but on the consumer side, the mainstream is the Internet of Things, in particular home automation. Such systems are called HEMS (home energy management system) [6]

Energy management systems have the following requirements [8]:

1) The system collects real-time data on energy consumption and production, as well as the status of devices.
2) The system stores and analyzes historical data.
3) The system manages the devices to ensure optimal energy consumption.
4) The user can control the devices directly and remotely.
5) The system warns the user in case of undesirable situations.

Let's consider the choice of metrics to evaluate the performance of energy management systems.

The environment in which energy management systems operate - residential premises, households, even office or industrial buildings - leads to the multipurpose nature of such systems [8], that is, the need to find a compromise between several objectives. While the global goal is always to increase energy efficiency, there are constraints on its achievement, in particular on user comfort. In addition, the formulation of a specific goal can vary depending on the context: emissions reduction, monetary savings, balancing the load on the energy system are some possible directions.

Energy management objectives can be expressed through cost. While the most obvious and predominant factor is the price of energy consumed, factors such as the initial installation costs of the system, the penalty for contributing to the total load, the projected depreciation of equipment, and the [9] greenhouse gas emissions tax can also be included in the overall calculation formula. In this way, the system can follow the single goal of reducing the monetary costs derived from this formula. Choosing the correct ratio can be difficult if there are no established monetary values for some goals.

The main objective of those that are difficult to represent in monetary terms is the comfort of users, that is, their inconvenience associated with the quality of service provided in the delivery of energy. The system should not lead to a significant change in their lifestyle. Various penalty functions can be used to assess the impact of the system on users' comfort: value limits, deviation, no-service penalty [9]. In other words, if cost represents some ratio to be minimized, then comfort requirements impose constraints on possible strategies.

The described above considerations can be useful to testing general approaches in optimizing energy consumption. For example, the tasks to estimate the benefits of adding a particular device and compare them with the purchase and installation costs, or to select the best algorithm among several algorithms under development.

From the end user's point of view, the system may have requirements such as access to historical data and trends, the ability to remotely manage devices, and warnings about undesirable situations [8]. Data privacy, reliability, and system efficiency also play a role in evaluating the quality of the system. Such requirements are more typical for practical products intended for direct intended use rather than research prototypes.

VI. Technical implementation details

Design and software implementation of the above components / applications was performed using the Node-RED visual programming tool combined with the use of cloud technologies, in particular Yandex IoT Core and AWS IoT Core. Node-RED is a flow programming tool for connecting hardware devices, APIs and online services. Their combined use allows prototyping Internet of Things systems without using real devices, allowing to focus through the architecture of the system before its physical implementation. One of the important advantages of Node-RED is that the tool can be used to create a simple prototype system in the same environment in which the main development is or will be carried out. Such an approach makes development much easier. Moreover, Node-RED also provides the means for easy visualization of the resulting system, which allows to prototype a graphical interface within the same environment.

The use of cloud services by applications, which include Yandex IoT Core and AWS IoT Core, significantly reduces system infrastructure costs while providing better scalability and fault tolerance. Cloud technologies offer computing, storage and communication resources.

For message transfer within the system we have used MQTT - a network protocol that uses a publisher-subscriber architecture and runs on top of TCP/IP using queuing (Message Queuing Telemetry Transport). It is convenient when used with low network bandwidth.

The implementation of a prototype system to control the lighting was carried out as described [10].

A. Running Node-RED on cloud-based virtual machine

To run the Node-RED environment, a virtual machine is created in the Yandex Compute Cloud service. This service is part of the Yandex Cloud and provides scalable computing power for creating and managing virtual machines. This service offers a wide range of virtual machine settings, from different operating systems to fine-tuning the resources used.

We have used a virtual machine based on the CentOS 8 operating system. As Node-RED didn't require much