Simulation Modelling Practice and Theory 65 (2016) 1–10

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Editorial

Analyzing and visual programming internet of things and autonomous decentralized systems

Yinong Chen

School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287-8809, USA

ARTICLE INFO

Keywords:
Internet of Things
autonomous decentralized system
Visual programming
IoT education

ABSTRACT

The development of Internet of Things, fueled by cloud computing and big data processing from upper level, and by ubiquitous sensory and actuator devices from the lower level, has taken a sharp turn towards integrating the entire information, computing, communication, and control systems. This special issue selected seven papers from the 2015 IEEE twelfth International Symposium on Autonomous Decentralized Systems (ISADS). These papers cover the latest research on IoT and ADS based system science and system engineering methods; the wearable sensor network development and applications; and data analysis for security and reliability in IoT and ADS applications. As an addition to these selected topics, this guest editorial paper also adds IoT education and dissemination aspects to this special issue. As the IoT research and applications expand explosively into all the domains, schools and universities must prepare students to understand and to be able to program the IoT devices. This paper presents a visual programming environment that allows students without programming background to learn the key concepts of computing and IoT devices, and to program IoT devices into different application systems.

1. Introduction

The development of Internet and cloud computing has pushed the desktop-based computing platform into an internet and Web-based computing infrastructure. It has changed the concept of physical products or things into services. The endorsement and commitment to building and utilizing cloud computing environments as the integrated computing and communication infrastructure by many governments and major computing corporations around the world have led the rapid development of many commercial and mission-critical applications in the new infrastructure, including the integration of physical devices, which gives Internet of Things the unlimited computing capacity.

Internet of Things (IoT) was initially proposed and applied in the Radio-Frequency Identification RFID-tags to mark the Electronic Product Code (Auto-ID Lab) [1]. IoT concept is extended to refer to the world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes [2]. Internet of Intelligent Things (IoIT) deals with intelligent devices that have adequate computing capacity. Distributed intelligence is a part of the IoIT [3]. According Intel's report, there are 15 billion devices are connected to the Internet, in which 4 billon devices include 32-bit processing power, and 1 billon devices are intelligent systems [4].

A wireless sensor network (WSN) is spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location [5]. The combination of WSN with IoT enables WSN to the resources on Internet, particularly the cloud computing and big data analysis capacity, which enormously enlarges the capacity of WSN in today's applications.

http://dx.doi.org/10.1016/j.simpat.2016.05.002
S1569-190X(16)30180-0/

In addition to IoT and wireless sensor networks, a number of related concepts and systems have been proposed to take the advantage of Internet and cloud computing.

An autonomous decentralized system (ADS) is a distributed system composed of modules or components that are designed to operate independently but are capable of interacting with each other to meet the overall goal of the system. ADS components are designed to operate in a loosely coupled manner and data are shared through a content-oriented protocol. This design paradigm enables the system to continue to function in the event of component failures. It also enables maintenance and repair to be carried out while the system remains operational. ADS and the related technologies have a large number of applications in industrial production lines, railway signaling and robotics [6,7]. ADS concepts are the foundation of the later technologies such as cloud computing and Internet of Things.

A cyber-physical system (CPS) is a combination of a large computational and communication core and physical elements that can interact with the physical world [8,9,10]. CPS can be considered the extended and decentralized version of embedded systems. In CPS, the computational and communication core and the physical elements are tightly coupled and coordinated to fulfill a coherent mission. The U.S. NSF (National Science Foundation) issued a program solicitation in 2008 on CPS, envisioning that the cyber-physical systems of tomorrow would far exceed those of today in terms of adaptability, autonomy, efficiency, functionality, reliability, safety, and usability. Research advances in cyber-physical systems promise to transform our world with systems that respond more quickly (e.g., autonomous collision avoidance), are more precise (e.g., robotic surgery and nano-tolerance manufacturing), work in dangerous or inaccessible environments (e.g., autonomous systems for search and rescue, firefighting, and exploration), provide large-scale, distributed coordination (e.g., automated traffic control), are highly efficient (e.g., zero-net energy buildings), augment human capabilities, and enhance societal wellbeing (e.g., assistive technologies and ubiquitous healthcare monitoring and delivery) [11].

Kakuda presented the concepts, technologies, and case studies of the next generation of assurance networks [12]. The study made use of the available redundant computing and communication resources for dependability purposes. The paper laid out the roadmap to the design and implementation of the new generation assurance networks. These networks inherit different features from a number of systems, including cyber-physical system, scalability and service orientation from cloud computing, the adaptability and autonomy from autonomous decentralized systems [6], fault tolerance and real-time computing from the responsive systems [13], and distributed real-time and embedded (DRE) systems. DRE systems are based on a model driven architecture and model integrated computing [14] and are applied in the situation where application requirements and environmental conditions may not be known *a priori* or may vary at run-time, mandate an adaptive approach to management of quality-of-service (QoS) to meet key constraints such as end-to-end timeliness. Different DRE middleware systems have been developed by the members of the Distributed Object Computing (DOC) Group, which is a consortium consisting of universities and industry partners [15]. The new generation assurance networks also include many other common features such as trustworthiness and mobility [16,17]. Scheduling tasks and allocating resources in such environments require new strategies and techniques. The gang scheduling and real-time scheduling techniques in ad hoc distributed systems and on the elastic cloud environment ensure that related tasks are scheduled in groups and are running simultaneously [18,19,20], which can be used to coordinate the intelligent devices.

Robot as a Service (RaaS) is a cloud computing unit that facilitates the seamless integration of robot and embedded devices into Web and cloud computing environment [3,21]. In terms of service-oriented architecture (SOA), a RaaS unit includes services for performing functionality, a service directory for discovery, and service clients for user's direct access [22]. The current RaaS implementation facilitates SOAP and RESTful communications between RaaS units and the other cloud computing units. Hardware support and standards are available to support RaaS implementation. For example, Devices Profile for Web Services (DPWS) defines implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices between Web services and devices. The recent Intel IoT-enabled architecture, such as Galileo and Edison, made it easy to program these devices as Web services. From different perspectives, An RaaS unit can be considered a unit of Internet of Things (IoT), Internet of Intelligent Things (IoIT) that have adequate computing capacity to perform complex computations [3], a Cyber-physical system (CPS) that is a combination of a large computational and communication core and physical elements that can interact with the physical world [4], and an Autonomous decentralized system (ADS).

This special issue selected seven papers from the IEEE twelfth International Symposium on Autonomous Decentralized Systems (http://isads2015.asia.edu.tw/). These papers cover the latest studies on IoT and ADS based system science and system engineering methods in natural disaster resilience; the wearable sensor networks development and applications; and data analysis for security and reliability in IoT and ADS applications, including forensics analysis and aviation data processing.

As the IoT research and applications expand explosively into all the domains in computing, information, and control systems, schools and universities must prepare students to understand and to be able to program the IoT devices [23]. As an extension to the domains covered by the selected papers, this paper presents a visual programming environment that allows novice developers to learn the key features IoT devices and to program IoT devices without dealing with the low level technique details. It uses the concepts of IoT as a Service and workflow integration to address the programming issues of all the aforementioned systems that are connected to the Internet and can be accessed through Internet. This programming environment is particularly useful for in IoT and programming education.

There are a number of great visual programming environments for computer science education. MIT App Inventor [24] uses drag-and-drop style puzzles to construct phone applications in Android platform. Carnegie Mellon's Alice is a

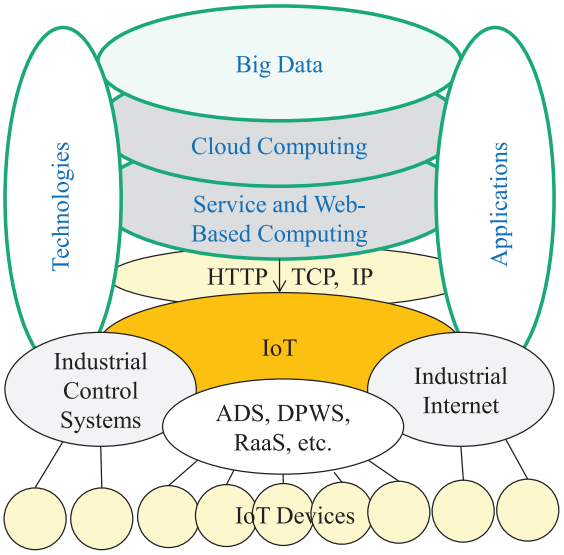

Fig. 1. Internet of things and devices

3D game and movie development environment [25]. It uses a drop-down list for programmers to select the available functions in a stepwise manner. App Inventor and Alice allow novice programmers to develop complex applications using visual composition at the workflow level.

Microsoft Robotics Developer Studio (MSRDS), with a Visual Programming Language (VPL), is specifically developed for robotics applications [26]. It is a milestone in software engineering, robotics, and computer science education from many aspects. MSRDS VPL is service-oriented; it is visual and workflow-based; it is event-driven; it supports parallel computing; and it is a great educational tool that is simple to learn and yet powerful and expressive. Sponsored by two Innovation Excellence awards from Microsoft Research in 2003 and in 2005, the author participated in the earlier discussion of the service-oriented robotics developer environment at Microsoft. MSRDS VPL was immediately adopted at Arizona State University (ASU) in developing the freshman computer science and engineering course CSE101 in 2006. The course grew from 70 students in 2006 to over 350 students in 2011. The course was extended to all students in Ira A. Fulton Schools of Engineering (FSE) at ASU in 2011 and was renamed FSE100, which is offered to thousands of freshman engineering students now.

Unfortunately, Microsoft stopped its development and support to MSRDS VPL in 2014 [27], which lead to our FSE100 course, and many other schools' courses using VPL, without further support. Particularly, the latest version of VPL (MSRDS R4) does not support LEGO's third generation EV3 robot, while the second generation NXT is out of the market.

To keep our course running and also help the other schools, we take the challenge and the responsibility to develop our own visual programming environment at Arizona State University (ASU). We name this environment VIPLE, standing for Visual IoT/Robotics Programming Language Environment.

ASU VIPLE is based our previous eRobotics development environments [23,28]. It is designed to support as many features and functionalities that MSRDS VPL supports as possible, in order to better serve the MSRDS VPL community in education and research. To serve this purpose, VIPLE also keeps similar user interface, so that the MSRDS VPL development community can use VIPLE with little learning curve. VIPLE does not replace MSRDS VPL. Instead, it extends MSRDS VPL in its capacity in multiple aspects. It can connect to different physical robots, including EV3 and any robots based on the open architecture processors. ASU VIPLE has been pilot tested at ASU in summer 2015 and in Spring 2016 as well as in several other universities. ASU VIPLE software and documents are free and can be downloaded at: http://venus.eas.asu.edu/WSRepository/VIPLE/

The rest of the paper is organized as follows. Section 2 discusses the standards and protocols that define the interface between various IoT/robotics devices to the Internet. Section 3 summarizes the papers selected for this special issue and discusses their contributions to IoT and autonomous decentralized systems technologies. Section 4 presents ASU VIPLE, the Visual IoT/Robotics Programming Language Environment, which makes IoT/robotics device programming as easy as drawing and connecting functional boxes. Section 5 concludes the paper.

2. IoT standards and protocols

Internet of Things (IoT) is a general concept that can be interpreted in different contexts. Different protocols and standards can be used for the communication between the devices and the Internet. As shown in Fig. 1, IoT connects to Internet through Internet protocols, such as HTTP, TCP and IP. The data received from IoT is represented as Web data in forms such as HTML, JSON, XML, and URI. The data can be further organized into ontology presented in RDF, RDFS, and OWL, for stor-

ing, analyzing, and reasoning. The data is typically processed in service-oriented and Web-based computing environment. If the data amount is big, it can be processed by cloud computing through big data analysis. At this end, the IoT and its data are fully integrated into the Web and the virtual world. All the technologies and applications developed can be applied to process IoT data and control the physical world connected to IoT on the other side. A convenient way of programming the IoT devices is critical to the success of the current organization stack, which allows the massive proliferation of applications.

On the other side, IoT is connected to devices and physical world through different device protocols and standards. For example, ADS uses a content-oriented protocol to broadcast data to a device bus, and the devices take the data based on a content code, instead of destination address [6,7]. In the railway application, when a train is delayed, it sends the delayed data to the content-oriented bus with a content code. No destination address is needed. Any other devices that need the delay information will pick up the data by the content code and apply the delay information.

Devices Profile for Web Services (DPWS) applies Web service standards in device interface and communication. It defines the implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices between Web services and devices [29]. DPWS specification is built on the following core Web Services standards: WSDL 1.1, XML Schema, SOAP 1.2, and WS-Addressing. Initially published in 2004, DPWS 1.1 was approved as an OASIS Standard together with WS-Discovery 1.1 and SOAP-over-UDP 1.1 2009. Microsoft .Net Framework Class Library has defined classes for supporting DPWS device programming. Devices that implement DPWS are already on the market. For example, Netduino Plus is an interface board that wraps a device with a Web service interface, so that the device can communicate with a virtual thing (a Web service) without requiring device driver code. The device is available in major stores, such as Amazon.com.

Robot as a Service (RaaS) applies Web service standards at the device level, just like DPWS. However, RaaS extends DPWS to include the service broker and service clients [3]. The service broker allowed the devices (robots) to register their IP address once they are online. The registration is necessary, as many of the devices are connected to the Internet via WI-FI, and their IP addresses are dynamically assigned. Once registered, the devices can be discovered and connected by looking up the device broker. As robots can be powerful devices, RaaS allows the robots to install multiple clients, and the robots can execute a different client to perform a different set of functions.

Defined by the US Nation Institute of Standard and Technology (NIST), Industrial control system (ICS) is a general term that encompasses several types of control systems used in different industry sectors. ICSs are typically used in industries such as electrical, water and wastewater, oil and natural gas, chemical, transportation, pharmaceutical, pulp and paper, food and beverage, and discrete manufacturing systems [30]. ICSs including

- Supervisory Control and Data Acquisition (SCADA) systems. They are used in large systems such as water distribution and wastewater collection systems, oil and natural gas pipelines, electrical power grids, and railway transportation systems. They are highly distributed and are used to control geographically dispersed assets, often scattered over thousands of square kilometers.
- Distributed Control Systems (DCS). They are often used as the nodes of a SCADA system, which control industrial processes such as electric power generation, oil refineries, water and wastewater treatment, and chemical, food, and automotive production.
- Programmable Logic Controllers (PLCs) are computer-based solid-state devices that control industrial equipment and processes. They are control system components used throughout SCADA and DCS systems.
- Other control system configurations, such as skid-mounted Programmable Logic Controllers (PLC) often found in the industrial sectors and critical infrastructures.

The Industrial Internet is a term initially proposed Frost & Sullivan consulting firm in 2000, which refers to the integration of complex physical machinery with networked sensors and software [31]. The Industrial Internet draws together fields such as machine learning, big data, the Internet of things, machine-to-machine communication and Cyber-physical systems. Industrial Internet is designed to extract data from devices, analyze it (often in real-time), and use it to adjust operations. In 2014, the Industrial Internet Consortium (IIC) was founded by AT&T, Cisco, General Electric, IBM, and Intel to bring together industry players to accelerate the development, adoption and wide-spread use of Industrial Internet technologies. Over hundred members from industry, academia, and governments are contributing to the consortium.

3. Topics of this special issue

This special issue is dedicated to the current efforts in IoT and the related technologies in ADS and sensor networks, as well as their applications in and impact to the rapidly changing society.

3.1. Natural disaster resilience

The paper by Harrison and Williams presents a systems approach to the natural disaster resilience [32]. It considers how the human mitigation of and the adaptation to the risks of natural disasters in cities and regions can be strengthened through the application of Systems Science perspectives and Systems Engineering methods building on the Internet of Things, autonomous decentralized systems, analytical techniques from Data Science., and general developments in Information and Communications Technology (ICT). As the populations in urban areas increase and the society and the daily life

of human relay more and more on the technologies, the destruction of the infrastructure will have much greater impact to human life in today than ever before.

The disasters cannot be avoided. Instead, they come more and more frequently. However, we can study and apply the resilience techniques to recover from the disasters more quickly. Resilience is the ability of a system, community or society exposed to hazards to resist, absorb, accommodate to and recover from the effects of a hazard in a timely and efficient manner, including through the preservation and restoration of its essential basic structures and functions. The resilience in a modern city environment refers to the capacity of individuals, communities, institutions, business, and systems within the city to survive, adapt, and grow no matter what kinds of chronic stresses and acute shocks they experience. Current technologies are not sufficient in implementing the city resilience. General software tools, such as the Global Earthquake Model (GEM) and the CLIMADA for probabilistic damage calculation and economics of climate adaptation, to support certain types of systems-of-systems analysis that we are advocating, but they are not a coherent system for resilience application. Failure-tree analysis has a long-history, but ignores inter-system and spatial dependencies. GIS systems are excellent at identifying spatial interactions, but they do not of themselves consider the system's functionality. This kind of such analysis is routine is in global supply-chain management, which studies the end to end performance under spatially-distributed failure conditions such as storms, strikes, production failures, and so forth, and this is still performed by human beings. No commercial tool considers how two or more systems might interfere during normal operation. Great efforts are needed to achieve the goal of resilience. ADS and IoT technologies are promising in providing integration and implementation of the city resilience, through their connection to many other technologies.

3.2. Wareable sensors and networks

The latest development in IoT and ADS is in the wearable sensors and their networks. The key issues to address in such systems are security and energy efficiency, in addition to the other sensor IoT and ADS features. This special issue selected two papers in this area.

Zhao et al. studied different physiological-signal-based key negotiation protocols for body sensor network (BSN) in medical applications [33]. BSN typically consists of dozens of biosensor nodes distributed on or in the human body. They are autonomous nodes and form a wireless network to measure physiological signals and execute intelligent treatment. BSNs have been widely applied for intelligent healthcare. Because physiological signals measured and processed by BSNs involve patient privacy, security mechanisms must be developed to secure BSNs, and therefore the adoption of available key negotiation protocols is fundamental. Due to stringently limited operation resources, BSNs require these protocols to be highly energy efficient. Recent development has discovered that certain physiological signals can be used for efficiently negotiating common keys among biosensor nodes. These signals and fuzzy technology are used for designing lightweight key negotiation protocols, and many solutions have been proposed. Zhao's paper explores and categorizes the available solutions, analyzes their merits and drawbacks, evaluates their performance, and presents open research issues in the domain.

As an application of the wearable body sensors, Zheng et al. studied the positioning system in the absence of GPS that establishing indoor directional guidance and localization [34]. Such a system can be applied, for example, in tracking a firefighter in action, a child or an elderly person in a shopping mall, and finding a car in a large parking garage. The main idea is to use Inertial Measuring Units (IMUs) for detecting the movement of a pedestrian in the pace and direction. In this research, a three-dimensional (3D) indoor positioning system is developed using foot mounted low cost Micro-Electro-Mechanical System (MEMS) sensors, which can locate the position and attitude of a person in 3D view, by plotting the path travelled by the person. The sensors include accelerometers, gyroscopes, and a barometer. The pedestrian's motion information is collected by the accelerometers and gyroscopes to achieve Pedestrian Dead-Reckoning (PDR), which is used to estimate the pedestrian's rough position. A zero velocity update (ZUPT) algorithm is developed to detect the standing still moment. A Kalman filter is combined with the ZUPT to eliminate non-linear errors in order to obtain the accurate positioning information of a pedestrian. The information collected by the barometer is integrated with the accelerometer data to detect the altitude changes and to obtain the accurate height information. The proposed system with our 3D model and algorithm has been tested in a simulation environment and in repeated physical experiments using different persons wearing the system. The results show that distance errors are around 1% and the positioning errors are less than 1% of the total travelled distance. These results are better than the other similar systems using the low-cost IMUs.

3.3. Big data analysis for IoT and sensor networks

As IoT, ADS, senor networks, and their applications are ubiquitously deployed, big amount of data are generated. Fast and real-time processing of the data is critical in many of applications to react to the arising situations. Two papers are selected in this area.

Lai et al. developed an anomaly detection model in an industrial autonomous decentralized system based on the real-time data tracks [35]. The data traffic model uses structural time series to analyze the behaviors for a chemical system application. The study focuses on monitoring the states of operation stations, engineer stations, and the industrial Ethernet. Aimed at avoiding the interference of impulse background traffic, the study proposes an approach to analyze and detect malicious traffic based on structural time series model. The model decomposes a complex time series into a sequence of components with actual content, instead of accurately depicting data generation process, which makes it easier and more

precise to express the variation features of sequences. According to the periodicity and stability of industrial traffic and a series of experimental results, a high-accuracy state-space method is developed to identify the required parameters of the model.

On the IoT, ADS, and sensor networks, big data analysis can create various intelligent applications, and many of them are developed and implemented as mobile services. Wei and Zhao developed a multiple granularity fused algorithm for mobile forensics applications, such as human tracking, vehicle tracking, and event detection [36]. The multiple granularity approach applies an abstraction technique that presents different level of details and allows the proposed algorithm to implement facial recognition in two steps: from a coarse level to a fine level. The study combines two kinds of multiple granularity and feature-based person recognition algorithms. It fuses biometric feature and face recognition algorithms into two kinds of strategies. A bionic pattern recognition based face recognition algorithm is also proposed. Experimental results show that the proposed new algorithm has a satisfied recognition precision and low time complexity in many forensics scenarios, particularly in noisy and under illuminated conditions, which allows the forensics algorithm to be implemented on IoT and mobile devices, where computing capacity is limited.

3.4. Aviation system data analysis

This section continues on the topic of IoT, ADS, and sensor network data analysis. It focuses on the aviation system data analysis.

With the rapid increase in air traffic demands, the more accurate and reliable tracking system for aircraft surveillance is required to improve the capacity, safety and efficiency of Air Traffic Control services. As a considered part of the surveillance infrastructure, the Secondary Surveillance Radar Mode S has been widely used in the aircraft tracking systems. Moreover, in the Mode S Enhanced Surveillance, the Downlink Aircraft Parameters (DAPs) are available for obtaining updated and detailed information from aircrafts. However, this information of aircraft parameters has not been used systematically to improve the performance of current tracking systems. Lu and Koga studied IoT and ADS application in air traffic applications [37]. They proposed a high-assurance aircraft tracking system by applying all the available Downlink Aircraft Parameters. The system with a high-accuracy estimator and a high-reliability checker enable not only tracking aircraft with changing flight modes in real time but also identifying potential faults of these parameters by pre-to-post data verification. Both simulation and physical experiments are conducted. The simulation results show that the proposed system significantly reduces the prediction error through maneuvers compared with the existing Interacting Multiple Model estimator. The results of physical experiments show that the proposed system has a satisfactory and stable performance of during tests. There are no significant disturbances due to false DAPs measurements, and the outputs can be observed by the pre-to-post validation check processes. Therefore, the proposed system not only obtains the better performance during the maneuvering periods but also has a lower computation cost and configuration complexity.

In our next paper, Morales discussed the situational awareness (SA) and its related metrics for rating the performance of flight crews, especially for flight safety purposes [38]. He developed a model and a simulation environment for the aircrew information management using Dynamic Bayesian Networks. The paper focuses on the work performed to validate different data discretization methods. The developed simulation environment is used to conduct experiments that monitor pilot activities, with special attention on information management. The outcome of simulated flights are datasets that contain relevant data, including control actions of the pilot, information queries, navigation information and environment and flight parameters. A large amount of data is generated during the experiments, driven by the complex and powerful flight data relationships provided by the System Wide Information Management standards. Thus, dynamic Bayesian networks are especially applicable for this research due to their suitability to learn probabilistic dependencies from data.

4. Visual IoT/Robotics Programming

This section adds as a new topic to the special issue. It presents IoT programming and education using VIPLE, a Visual IoT/Robotics Programming Language Environment.

4.1. Backend programming model

Different techniques can be applied to program IoT and robotics devices. VIPLE uses the same front end and backend programming model that is used in Microsoft VPL.

On the front end, the IoT and robotics devices will support at least the programs that read and process sensor data, interpret commands from the back end, and control the actuators. A device can be implemented as a server that offers a set of services. RaaS (Robot as a Service) implements this style [3]. The way it works is to have a service broker implemented as a Web service. When the device is powered up, it logs its dynamic IP address into the service broker [39]. The following operations are offered at this service broker: AddData(), DeleteDocs(), GetAddress(), GetData(), and SetAddress.

At the backend, three methods are implemented to connect and access the front device. One way is to look up the broker to find the device's IP address.

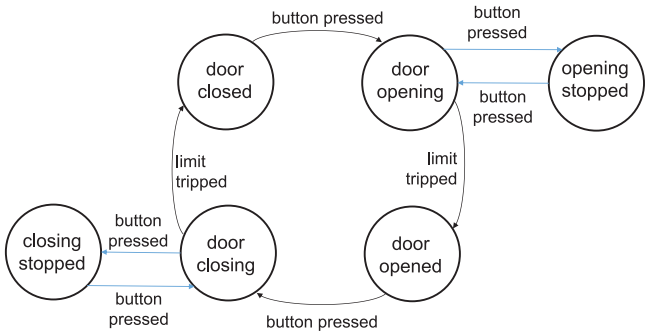

Fig. 2. Finite state machine for a garage door opener with a single button remote controller

The second way of implementing device as a service is to use the ad hoc communication mode, where the device has a hard-code virtual IP address, and the backend computer can connect to the address in ad hoc mode. The connect mode works when both device and the backend computer are in the direction connection distance.

The third way of establishing the connection is to implement the IoT/Robotics device as a client. Both the backend computer and the front end device connect to a standard router, and then the program at the backend connects to the device via a TCP or Web socket.

One of the advantages of programming IoT devices at the backend is the ability to run a powerful programming environment. This can make IoT programming much easier than directly programming on the device. ASU VIPLE (Visual IoT/Robotics Programming Language Environment) is such a programming environment, and it allows novice programmers to program complex IoT and robotics devices.

4.2. Finite state machine model

VIPLE is general-purpose programming language and is Turing complete in its capacity. It can be used for any kind of computation tasks. In VIPLE, a program is represented as a diagram of workflow, and the components are defined as activities or Web services in the diagram.

Although VIPLE can be used as general-purpose programming language, its strength is in event-driven programming that can respond to a sequence of events. The event-driven applications are best described by finite state machines consisting of states and transitions between the states. The transitions are triggered by events. As an example, Fig. 2 shows the finite state machine for a garage door opener. The control system consists of a single button remote controller and a limit sensor. The button on the remote controller controls the door open, stop, and close. The limit sensor is a build-in sensor in the motor. When the door reaches the limit, the sensor will generate a notification and stops the motor.

The VIPLE program that implements the finite state machine is given in Fig. 3. The remote controller button is simulated by the keyboard “Ctrl” key. As shown in Fig. 3, six states are coded in the If-activity with six conditions. When the Ctrl key is pressed, the If-activity checks the current state, transits into the next state, and prints the result in the console using the Print Line service. The limit sensor is simulated by the “m” key. When the m-key is pressed while the door is opening or is closing, the state will change and the result will be displayed.

4.3. VIPLE devics and interface definition

ASU VIPLE supports an open interface to other IoT and robotics platforms. Any device that can be programmed to support the same interface and can interpret the commands from ASU VIPLE program can work with ASU VIPLE. ASU VIPLE program communicates with the robot using the JSON object shown in Fig. 4, which defines the types of devices and their inputs to the robot from the ASU VIPLE program and the outputs from the robot to the ASU VIPLE program.

The ASU VIPLE environment encodes the control information into this JSON object. The robot just needs to interpret the script and perform the actions defined. On the other hand, the robot encodes the feedback in the same JSON format, sends the object back to the ASU VIPLE program. Then, the ASU VIPLE program will extract and use the information to generate the next actions.

ASU VIPLE supports all types of IoT and robotics devices by diving them into two types: The closed architecture devices and open architecture devices. The closed architecture devices do not allow application developers to deploy programs to the devices to process the JSON objects. In this case, we write specific services on VIPLE side to access these device’s open APIs. For each of such devices, we need to implement a set of specific services for the device. Currently, we have implemented specific services for LEGO EV3 robots in this way, as EV3 does not have an open architecture, and we cannot deploy code to EV3 to interpret the JSON object. Instead, we follow EV3 open APIs, and we write specific services on VIPLE side to call EV3 APIs to implement the communication and command interpretation. On the other hand, the open architecture devices

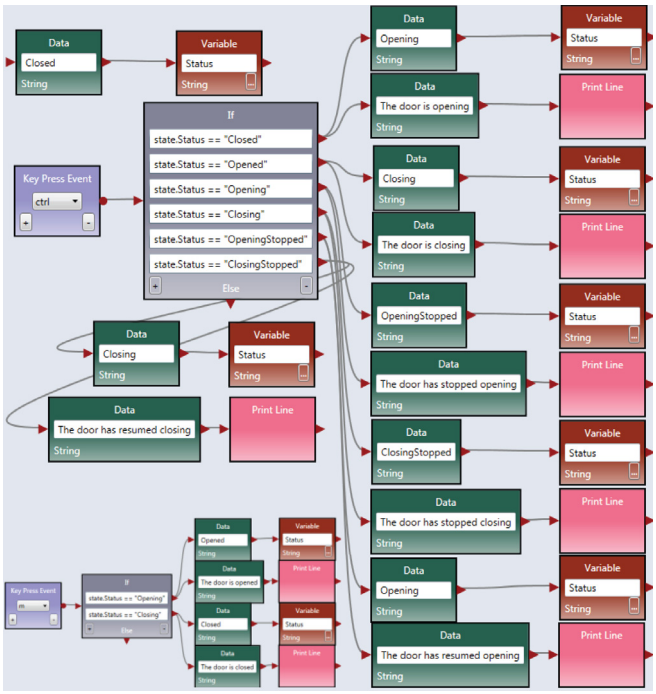

Fig. 3. VIPLE diagram implementing a garage door opener

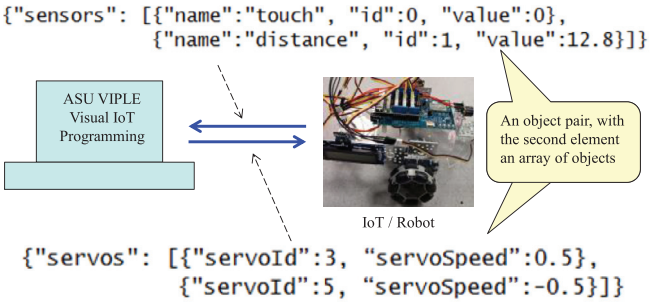

Fig. 4. JSON object for the backend and front end communication

allow application developers to deploy programs to the devices that can process the JSON objects. We just need implement one set of services on ASU VIPLE side to read and control all open architecture devices.

In the current VIPLE version, three sets of services are implemented. The first set of services is a list of general services, including the Simple Dialog, Key Press Event, Text To Speech, Print Line, Timer, and RESTful services. This set of general services can be used not only for robotics applications, but also for general purpose applications. The RESTful service allows VIPLE to call any RESTful services available online, which allow application developers to extend VIPL functionalities by implementing their functions in RESTful services. The second set is for LEGO EV3 robots, including EV3 brick, different EV3 sensors and motors. The addition of EV3 services allows the Microsoft VPL developers who used NXT robots to use the new third generation EV3 robots. The third set is used for connecting to open architecture robots, sensors, and motors, which allows the developers to use general purpose processors, such as Intel and ARM processors to build robots for VIPLE applications.

As an example, we show the VIPLE code and the configuration for communicating with different sensors on the devices. Fig. 5 shows the VIPLE code to test generic sensors and motors connected to an open architecture robot. The device consists of a robot (My Robot 0), a touch sensor, a distance sensor, a color sensor, a single-input motor and a two-motor combo drive device.

In order for the robot, the sensors, and the motors to communicate with the ASU VIPLE properly, we need to configure the partnership between the robot and its devices, the IP address, and ports. Fig. 6 shows the configuration of the three devices: robot, drive, distance sensor, and touch sensor. Notice that the numbers may differ from different robot configuration.

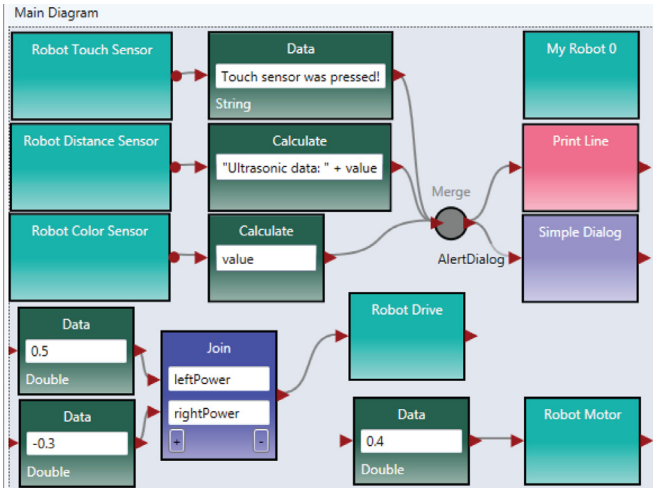

Fig. 5. VIPL code testing generic sensors and motors

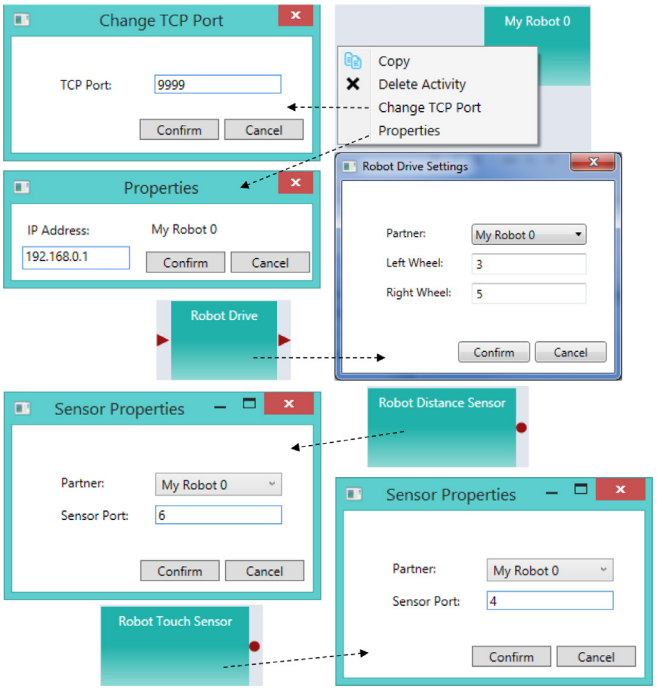

Fig. 6. Configuration of IoT device ports

ASU VIPLE has been pilot tested multiple times at Arizona State University and several other universities. The full documents for ASU VIPLE development, open architecture device middle ware for communication between VIPLE and device, program download, videos, and PPT presentations can be found at VIPLE site at: http://venus.eas.asu.edu/WSRepository/VIPLE/

5. Conclusions

This guest editorial paper introduced the topics and contributions of the selected papers in this special issue. These papers cover the latest research on IoT and ADS design, implementation, and analysis presented in ISADS 2015. As an addition to these selected topics, this paper also presented a visual programming environment for IoT and robotics application development. This environment is a new educational tool for high school students and college freshman students to understand and learn IoT and robotics programming.

Acknowledgement

The guest editor would like to thank the steering committee chair of ISADS, who have started ADS technology and the ISADS conference series, the authors who have submitted their papers to this special issue, and the reviewers who provided valuable and constructive comments and suggestions to the authors. The special issue has received 17 submissions, and we selected seven papers that are of high quality and are relevant to the topics of this special issue. The guest editor also would like to thank Gennaro De Luca and the VIPLE development team at Arizona State University. VIPLE provides an excellent topic that extends this special issue to IoT programming and education.

References

[1] IoT, http://en.wikipedia.org/wiki/Internet_of_Things
[2] Stephan Haller, http://services.future-internet.eu/images/1/16/A4_Things_Haller.pdf, May 2009.
[3] Yinong Chen, Hualiang Hu, Internet of Intelligent Things and Robot as a Service, Simulation Modelling Practice and Theory 34 (May 2013) 159–171.
[4] GM Ton Steenman, Intel Intelligent Systems Group, Accelerating the Transition to Intelligent Systems, Intel Embedded Research and Education Summit (February 2012) http://embedded.communities.intel.com/servlet/JiveServlet/downloadBody/7148-102-1-2394/Accelerating-the-Transition-to-Intelligent-Systems.pdf .
[5] F. Akyildiz, I.H. Kasimoglu, Wireless Sensor and Actor Networks: Research Challenges, Ad Hoc Networks 2 (4) (Oct. 2004) 351–367.
[6] Kinji Mori, Autonomous Decentralized System and Its Strategic Approach for Research and Development, in: Invited Paper, Special Issue on Autonomous Decentralized Systems Theories and Application Deployments, IEICE Transactions on Information and Systems, E-91-D, Sept. 2008, pp. 2227–2232.
[7] Autonomous Decentralized Systems (ads): https://en.wikipedia.org/wiki/Autonomous_decentralized_system
[8] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, John Stankovic, Cyber Physical Systems: The Next Computing Revolution, in: 47th Design Automation Conference (DAC 2010), CPS Demystified Session, Anaheim, CA, June 17, 2010.
[9] Wikipia, Cyber-physical system, http://en.wikipedia.org/wiki/Cyber-physical_system
[10] Jian Huang, Farokh Bastani, I-Ling Yen, Wenke Zhang, A Framework for Efficient Service Composition in Cyber-Physical Systems, in: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), Nanjing, June 2010.
[11] NSF solicitation, “Cyber-Physical Systems”, 2008, www.nsf.gov/pubs/2008/nsf08611/nsf08611.pdf
[12] Yoshiaki Kakuda, Assurance networks: concepts, technologies, and case studies, in: Proc. Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC 2010), Xi’an, China, October 2010, pp. 311–315.
[13] M. Malek, Responsive Systems: A Marriage Between Real Time and Fault Tolerance, Keynote Address, in: Proceedings of the Fifth International Conference on Fault-Tolerant Computing Systems, Nuernberg, Germany, Springer-Verlag, Informatik-Fachberichte, September 1991 283, 1-17.
[14] Chris Gill, Jeanna M. Gossett, David Corman, Joseph P. Loyall, Richard E. Schantz, Michael Atighetchi, Douglas C. Schmidt, Integrated Adaptive QoS Management in Middleware: An Empirical Case Study, in: submitted to the 24th IEEE International Conference on Distributed Computing Systems (ICDCS), Tokyo, Japan, 2004.
[15] DOC, Distributed Object Computing Group for Distributed Real-time and Embedded (DRE) Systems, http://www.dre.vanderbilt.edu/
[16] Hiroshi Nakagawa, Kazuyuki Nakamaru, Tomoyuki Ohta, Yoshiaki Kakuda, A hierarchical routing scheme with location information on autonomous clustering for mobile ad hoc networks, ISADS (2009) 269–274.
[17] Yinong Chen, W.T. Tsai, Towards dependable service-orientated computing systems, Simulation Modelling Practice and Theory 17 (8) (September 2009) 1361–1366.
[18] Ioannis A. Moschakis, Helen D. Karatza, “Evaluation of gang scheduling performance and cost in a cloud computing system, the Journal of Supercomputing 59 (2) (2012) 975–992.
[19] Zafeirios C. Papazachos, Helen D. Karatza, Performance evaluation of bag of gangs scheduling in a heterogeneous distributed system, Journal of Systems and Software 83 (8) (2010) 1346–1354.
[20] Georgios L. Stavrinides, Helen D. Karatza, Scheduling multiple task graphs in heterogeneous distributed real-time systems by exploiting schedule holes with bin packing techniques, Simulation Modelling Practice and Theory 19 (1) (2011) 540–552.
[21] Yinong Chen, Zhihui Du, M. Marcos Garcia-Acosta, Robot as a Service in Cloud Computing, in: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), Nanjing, 2010, pp. 151–158.
[22] W.T. Tsai Yinong Chen, Service-Oriented Computing and Web Software Integration, 5th Edition, Kendall Hunt Publishing, 2015.
[23] Yinong Chen, Zhizheng Zhou, Robot as a Service in Computing Curriculum, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 156–161.
[24] App Inventor, http://ai2.appinventor.mit.edu/
[25] Alice, http://www.alice.org/
[26] Microsoft Robotics Developer Studio, https://msdn.microsoft.com/en-us/library/bb648760.aspx
[27] Wikipedia https://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio
[28] ASU eRobotics programming environment, http://venus.eas.asu.edu/WSRepository/erobotic/
[29] DPWS, https://en.wikipedia.org/wiki/Devices_Profile_for_Web_Services
[30] ICS, https://en.wikipedia.org/wiki/Industrial_control_system
[31] Industrial Internet, http://www.industrialinternet.us/revive/
[32] Colin Harrison, Natural Disaster Resilience, in: Forum speech at the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015 p. xxxii.
[33] Huawei Zhao, Ruzhi Xu, Minglei Shu, Jiankun Hu, Physiological-Signal-Based Key Negotiation Protocols for Body Sensor Networks: A Survey, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 63–69.
[34] Lingxiang Zeng, et al., A Foo-Mounted Sensor Based 3D Indoor Positioning Approach, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 145–150.
[35] Ruikang Zhou, et al., Study on authentication protocol of SDN trusted domain, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 145–150.
[36] Zhihua Wei, Rui Zhao, A Hybrid Feature Based Mobile Forensics System, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 151–155.
[37] Xiaodong Lu, Tadashi Koga, Real-time Oriented System Wide Information Management for Service Assurance, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 175–180.
[38] Carlos Morales, Serafin Moral, Discretization of Simulated Flight Parameters for Estimation of Situational Awareness Using Dynamic Bayesian Network, in: the 12th International Symposium on Autonomous Decentralized Systems, Taichung, March 2015, pp. 196–201.
[39] Service broker for IP address registration: http://venus.eas.asu.edu/WSRepository/RaaS/RaaS_Broker/Service1.asmx