An Internet-of-Things System Architecture based on Services and Events

Shiddartha Raj Bhandari [#1], Neil W. Bergmann [#2]

[#] *School of ITEE, University of Queensland*
St Lucia Q 4072, Australia

[1] siddhartha.raj.bhandari@gmail.com

[2] n.bergmann@itee.uq.edu.au

***Abstract*—Most modern building automation systems use closed, proprietary networking protocols and control systems. They are difficult to program, do not respond well to component failures, and are not easily expandable. This paper proposes an open system architecture based on modern networking middleware and service Oriented Architectures. An Event-Condition-Action framework for describing system behaviour is also proposed and explored.**

I. INTRODUCTION

Considerable work has been done over the past decade to transform the Internet from a way to access specific documents on specific machines into a system which provides services independently from the physical location of those services. Service-oriented Architectures (SOA) are now a standard Internet design methodology, and have spawned a large number of Internet standards [1]. Laskey and Laskey list some of the features of well –known features of an SOA [1]:

> *"Services are made visible to potential users, interact with users through a series of information exchanges, and produce real-world effects. Invocation and information exchange rely on standard languages and protocols that facilitate interoperability. From a user's perspective, invoking a service to produce an intended effect is experienced as a single, atomic operation. The detailed sequence of actions carried out by the service may involve any number of operations such as database queries, data transformations, execution of models, and formatting of displays. These operations may themselves be composed of lower level operations. The data and/or software modules required to perform the operations may reside at different physical locations and be controlled by different organizations. The details of composing the sequence of actions that produces the real world effect are transparent to the user interacting with the service. SOA frees users to concentrate on their business problem, leaving the details of the solution to the service."*

The Internet pervasively links global information sources and information access devices (computers, smart phones, etc). The recently popularized term "Internet of Things" (IoT) is used to describe evolution of the Internet to include more than just information and services. Internet connectivity will be extended to many of the physical objects associated with daily life – people, buildings, appliances, vehicles, and tools.

"Internet of Things" was a term first used around 1999 by developers of Radio Frequency IDentification (RFID) systems that were intended as bar-code replacements. By placing unique tags on everyday items, which could be interrogated by appropriately networked readers, the location and condition of individual "Things" could be monitored to improve business processes. The use of the term has now been extended to encompass a broader range of networking technologies (not just RFID) and a broader range of applications (not just location and condition monitoring).

IoT is now an area of significant international research interest. In 2010, the EU Framework 7 program has funded significant IoT projects. The Flagship IOT-A project [2] aims to develop a reference architecture for a future Internet of Things, and is funded at 18.7 million Euro (25 million AUD) over 3 years from 2010. The European Research Cluster on the Internet of Things [3] provides collaboration and coordination between 42 separate EU funded projects within the broad area of IoT, with cumulative research funding of over 100 million Euros. In 2010, China set up its first "Internet of Things" research centre in Shanghai, with funding of more than US $100 million. 2008 saw the first international IoT conference, and the 2010 IoT conference had sponsorship from major industry players such as IBM, NTT, SAP and Cisco.

A key application of IoT is in smart environments, such as smart buildings [4]. The arguments for using a service-oriented architecture as an architectural framework for IoT are the same as the reason for using a service-oriented architecture for enterprise-level activities. An SOA separates the services which the user wishes to implement (in this case switching on lights, maintaining a comfortable room temperature, minimising energy use, maintaining physical security) from the devices, servers, interfaces and networks which are required to implement those services.

SOA standards and packages for enterprise-level systems are readily available, relatively mature, and well-understood. However, it is a non-trivial task to transfer these existing technologies to the IoT smart building environment where devices are very heterogeneous, often small and low-cost, with constrained computing power, communication bandwidth and limited energy resources. Additionally, the number of devices may number in the hundreds or thousands, so system

setup and management is further complicated. One goal of this paper is to explore this problem of how to map existing SOA techniques to the IoT domain.

Most of the application development approaches used involving resource constrained objects in general and building environment proposed so far are proprietary. However, using open standards based approaches from conventional Service Oriented Architectures (SOA) has the potential to allow heterogeneous systems to be integrated and deployed.

In the past few years, IoT applications based on Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) based web services been proposed [5]. However, most of the applications realized based on REST or SOAP approaches are operator initiated and require manual control, such as smart home application proposed in [6] requires users to control devices in building environment manually.

Building automation systems are expected to be automatic and intelligent. "Automatic" means that application should be able to perform operations without intervention of the user, and "intelligent" means that based on the current circumstances, the system should be able to behave accordingly even if responses to every possible combination of situations have not been explicitly specified.

In a smart building environment, most of the network objects are resource constrained, and so they are not capable of processing and providing useful information on their own. For instance, a sensor cannot perform complex processing as they have limited processing, memory, storage and power resources. However, such resource constrained devices are capable of initiating some action by a more complex system when some changes occur in the environment. For example, when room temperature goes below threshold level, a temperature sensor can generate an alert by sending a relevant message to the control system which can further take relevant action.

Therefore, this paper proposes a new building automation framework which leveraging SoA integrates IoT devices within an Event-Condition-Action (ECA) framework[7]. Leveraging functionalities of underlying service abstraction layer, we propose necessary components on top of which home automation applications can be realized.

Upon occurrence of an event, this system evaluates which possible semantic conditions are true and triggers suitable actions. The proposed framework divides its operations basically into three stages: event detection, condition evaluation and action execution.

A rule based approach is used in condition evaluation that is used to implement the goals of intelligence and automation. The rule based system consists of set of rules that get executed automatically when all the facts required to fire that particular rule become true. When an event occurs and certain facts become true, corresponding rules get executed and actions get performed in the building environment. Rules are stored in a rule base and a rule engine scans the rule based automatically when any facts become true.

In an IoT building environment where large numbers of devices are resource constrained, this ECA based approach combines resource constrained sensors and actuators with resource-rich devices like home gateways that can also take the role of the rule-based controller.

II. BACKGROUND

An ECA pattern based approach is one of the event driven approaches that can be used in order to model some applications and services related to control and management [7]. It can model any domain where operations to be performed depends on occurrence of some events and certain conditions need to be fulfilled, such as event-based business process composition proposed for the case of enterprise environments in [8]. Similarly most of the application scenarios in building environment can be modelled following an ECA pattern. A very simple example considers the case of managing building security. If an authorised person appears at the building, the automatic door opening system should allow the person to enter, otherwise the system should alert the respective operator about the arrival. This simple operation can easily be modelled as an ECA pattern based approach: user's arrival (including user identification) is the event part, checking whether the person is authorized or not is the condition part and either opening the door or alerting the security officer is the action part. In general, ECA pattern based modelling can be represented as in figure 1, in which interaction among events, conditions and actions are shown; detailed descriptions are provided in section III .

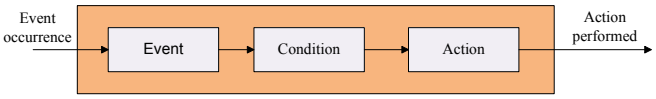

Fig: 1 ECA based approach

In [9], authors have proposed an event-driven sensor virtualization approach for IoT, however the paper does not provide a clear notion of events, how they are subscribed to and finally utilized. Our framework aims to clearly define events. Thanks to ECA components, applications can subscribe to relevant events following an event driven SoA without requiring direct low level interaction with devices. Similarly, in [10], authors have proposed a conceptual modelling approach of processing primitive events occurring in IoT environment to generate complex events. However, events are restricted to RFID technology and a business expert with domain knowledge is expected to be responsible for complex modelling. Our framework however aims to incorporate heterogeneous devices that are able to be connected into the network and provide standard and well understood SoA based interfaces so that a normal application developer can incorporate low level device functionality as with simple web services.

As consideration of events generated by resource constrained devices like sensors in real time are increasingly important in various scenarios in building environments, our proposed

ECA based framework provides the necessary components to allow an easy and standard way of utilizing them.

III. Proposed Framework

In this section we explain the architecture of the proposed framework which consists of various layers with different components as shown in figure 2.

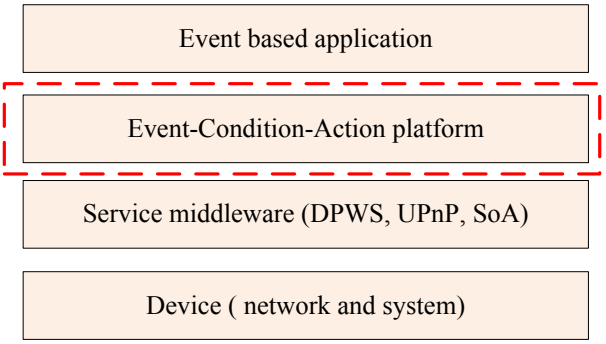

Fig: 2 ECA pattern based framework

A. Device Layer

This layer deals with the various devices that are present in building environment. Devices could be of any type; however we have focused more on resource constrained objects specifically sensors, actuators, and controllers that can be used in building management, security, heating, and lighting. Sensors can detect changes that occur in the building environment. A temperature sensor can detect temperature increase or decrease, an activity sensor can detect the presence of people in an area, or a light switch may detect an occupant's desire for lighting. Sensors are the source of external events in this framework.

Actuators perform actions such as switching on lights, heaters or opening doors or windows. Objects can be categorized into two groups based on their ability to be incorporated into the web directly or not. Sensors, actuators or any devices supporting a TCP/IP layer can independently be connected with the Internet. Small sensors with 6LowPAN connectivity are in this first group and are the focus of our research. However, provided with a suitable gateway, proprietary devices can also become part of such a framework. At the device layer, whenever devices are brought into the network, they will be able to communicate with other devices, either directly or with the help of a gateway.

B. Service Layer

In the Service Layer, the functionalities of devices are represented as web services. There are two possible service description approaches that can be selected in this layer: WSDL (Web Services Description Language) and WADL (Web Services Application Language) [11]. WSDL follows the SOAP standard and is the description language for the DPWS specification. WADL is used in REST based approaches, and is more lightweight than WSDL. However, WADL has not yet been standardized and is not used by many developers. Hence, we have selected WSDL based description and DPWS based approach as components of our framework which also supports automatic discovery of services compared to manual search of WADL in case of REST.

C. ECA platform

The ECA layer is the main focus of this work, and it consists of components that enable the event based automatic and intelligent actions required for building facility management applications. Although targeted at smart buildings, this approach is useful in general scenarios where operations are performed based on an ECA framework. The ECA based approach divides its operation basically into three stages namely, event processing, condition evaluation and action execution. The event part is further divided into event detection, subscription and processing to detect, subscribe and process events. The condition part deals with condition evaluation and it is based on a rule based system consisting of a fact base and a rule base. The action part consists of an action engine that is responsible for executing actions that are triggered after condition evaluation. Execution of an action is to execute a web service that abstracts device functionality such as, when temperature goes above allowed limit, executing the cooling functionality of air conditioner is to execute corresponding web service representing that particular operation. It could be specifically a particular cooling device or an automatic action that satisfies the relevant condition triggering action (cooling environment).

1) Event generation

An event is defined as a sensed change in the building environment that needs further processing. In general there are two types of events that may occur: internal and external. Internal event occurs within the application logic inside a program, such as a timer that expires. An external event is generated by an external sensor, e.g., a sensor detects that the temperature inside the building is lower than the set point and it transmits an event message. A notify/subscribe event networking model is the most suitable framework for signalling events.

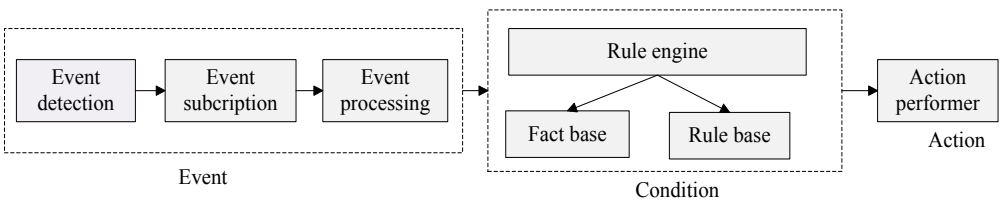

Fig: 3 Event-condition-action based platform

2) Condition Evaluation – Fact Base

Condition evaluation deals with evaluation of the event based on facts that become true when an event occurs. The current state of all facts is stored in a fact base.

For example, in a building environment, when a user arrival event occurs, the facts based on that arrival are updated. Such evaluation occurs based on the facts that become true when an event occurs. Depending on the type of sensor, the facts could range from an arrival of one or more people, to a complex event such as the arrival of a person recognised as a building resident by biometric evaluation.

3) Condition Evaluation - Rule Engine

A rule based approach is one of the techniques that can be used in realizing an expert system. Rules define how the system should operate and are described in some rule description languages, such as C based CLIPS and Java based JESS. Depending upon the domain in which the rule based approach is used; rules are different. In a business computing domain, a rule can encapsulate business processes, such as the flow of expenditure approvals. In our framework, as we are dealing with building related facility management, rules incorporate building automation related processes.

The condition evaluation system consists of a rule engine (rule scanner), fact base and rule base as shown in figure 3. Facts represent situations that may occur in the building environment. For example, "Bob is in the Kitchen" is an example of a fact and "On a person entering the bathroom, if it is dark, switch on the bathroom light" is an example of a rule. Rules consist of an ECA based pattern: *on <event> if <condition> then <action>*. Whenever an event occurs, the rule scanner checks all relevant rules to determine any consequent actions that need to be initiated.

4) Interaction of event, condition and action modules

Figure 4 shows the steps that the proposed framework follows from when an event occurs to the execution of a consequential action, and these steps are described as follows.

When a device generates an event, the event processing component of our framework receives the corresponding event notification. The format of event message will depend on the complexity of the device. If the device supports direct web services, the detected event may consist of either XML or JSON (Java Script Object Notation) formatted messages, as commonly used in the web service domain. In the case of Bluetooth or Zigbee devices, however, the message varies according to its corresponding notification implementation. The Event Processing step maps device level events into higher abstract level events that in turn update the fact base of the system. When a person enters a bathroom, the activity sensor changes its state from "off" to "on" and sends a message to the framework informing a change of the state in its relevant message format. When the device reports changes it can be further be mapped into an abstract event such as a particular person or sensor being detected in a particular location. This mapped abstract event can be used in order to notify any application that has previously subscribed to it as well as updating new facts.

Each possible event is capable of generating new facts in the system. In this particular event, new facts could be "bathroom is occupied" and "someone is in home". It is the responsibility of the fact update module to enable all related facts of any particular event. It is the fact update module that is responsible for filtering repetitive events such as the same person was detected a minute ago at entrance and the fact "someone is in home" fact was updated.

After the fact update module updates the fact base, the rule scanning module gets triggered.

When the rule engine is alerted about changes, it starts scanning its rule base against facts that are just enabled, In our example the new facts are "someone is in home", and "bath room is occupied". A rule basically consists of an expression with LHS (Left Hand Side) and RHS (Right Hand Side). LHS defines possible facts and its combinations while RHS consists of actions. a very simple example in our case could be *<if bathroom is occupied => front_panel_display (occupied) >* that displays occupied notice outside the bathroom. The RHS side of the rule can be either execution of some functionality as mentioned before or it could be also some expression that further generates events or update facts. Such as if the person inside the bathroom is a child, start a timer in case if child doesn't exit in a reasonable time. If the timer is expires that enables facts "child is inside bathroom" "allocated timer expired" and then a new rule gets fired alerting parents that "child is overtime inside bathroom".

.

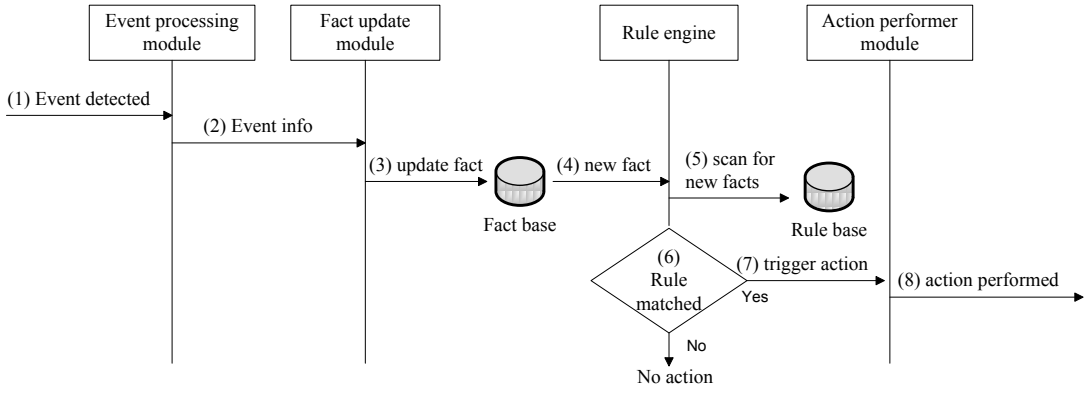

Fig: 4. ECA pattern based operation

The action performer module is responsible for performing actions that are in the RHS side of rules. Actions include displaying notifications, starting or stopping devices; generating events that eventually enable more facts or accessing external web services

5) Semantic Mapping

Another key issue is the domain in which rules are constructed. If a system is to operate in an automatic, intelligent and fault tolerant fashion, then it will be necessary to write rules in a semantic domain [12]. There will then need to be a semantic mapping between meanings in rules, and the actual capabilities and characteristics of sensors and actuators. This will be a key responsibility of the service-level mapping on sensors and actuators.

IV. DISCUSSION

To date, our proposed framework is still in its early phases, and this paper does not yet present any implementation details. We have proposed the required components and how they fit into a general SoA based IoT architecture and how the system could achieve automation and a certain level of intelligence exploiting ECA patterns. Another contribution of this paper is to describe the problems and design decisions that must be tackled to implement such a system. Following is a list of such questions.

What communication protocols are used by sensors and actuators? – Existing sensors and actuators for smart buildings employ a wide range of different protocols in network and application layer. Hence, such a system must be able to cope with a wide variety of such protocols both at the network layer (WiFi, Zigbee, IEEE 802.15.4, Ethernet, BACnet, ..). At the application layer there are several suitable communication protocols. CoAP (Constrained object Application Protocol) has recently been proposed for the case of resource constrained objects like sensors and actuators and is in the process of becoming a standardized application layer protocol by the IETF. HTTP (Hyper Text Transfer Protocol) is considered to be the default protocol of the WWW (World Wide Web). CoAP is a possible replacement to HTTP, however it is still to be standardized and almost all currently available services are based on HTTP requiring a proxy to interoperate with CoAP. Also, as UDP is the underlying transport protocol, broadcasting is possible in CoAP compared to HTTP that operates on top of TCP which has serious performance issues in wireless communication.

Another important factor to be considered is the request and response format to be used in communications. Probable options are simple text based messages, XML messages or JSON formatted messages. XML and JSON are structured message formats widely used in SoA, however they are liable to incur further overheads, especially XML which needs to incorporate more structural information even to represent fairly small data, such as temperature changes on a sensor. In such a case normal text based formatting may be better, however this requires a particular text processing module making the solution less flexible. Tradeoffs between flexibility and performance gain needs to be evaluated. It is our belief that these disparate protocols are best accessed through a uniform gateway, which converts individual protocols into a single format of system messages.

How are services accessed? The success of service-oriented architectures in enterprise systems suggests that sensors and actuators may be usefully accessed as services, either directly from the sensor or often via the gateway. Development of 6LoWPAN and a small foot print HTTP server on a sensor itself allows a sensor to be accessed just as another web application using a web client like a browser. Though it is technically possible, from a resource point of view it may drain scarce power of a sensor as it requires to run applications all the time. Running a sensor behind a gateway however, allows caching sensor data in the gateway for some period of time so that the sensor itself does not need to handle each and every request. Also, most of the resource intensive operations can be offloaded to the gateway preserving sensor resources further.. If sensors are capable of supporting IP based connectivity directly, our framework can directly access them as client-server based services. In order to support automation and intelligence in a building environment, sensors and actuators will be usefully accessed as event-based services. Real-world events such as a person entering a room can best be modelled by an event-based networking model, such as publish/subscribe.

How are services described? At some point in the system, syntactic events (PIR sensor with ID 12345678 switched from off to on) need to be converted to semantic events (person has entered the bathroom). Mapping techniques that classify real events into generic abstract events are being explored. This mapping allows applications to subscribe to events in an abstract manner rather than being specific to a particular event generated by some particular device. If the device is replaced or removed just the semantic mapping needs to be changed making the application resilient to any underlying hardware changes. This improves abstraction and adds modularity into the system which are very important in the case of resource constrained devices with limited resources where device failure is likely to occur.

How are facts and rules described? This is perhaps the key question that will drive much of the rest of the architecture of this system. Rules should be as generic or specific as needed. For example some rules might be specific to a person and a location (if Bob enters the living room then …), others to a class of people or places (if a family member approaches an entrance, then ….), and others to all people and places (if a person enters a dark room then ….). Leong et al [13], for example, present one possible ECA rule framework based on work from active databases. Substantial work is still needed to develop a suitable rule framework for building automation that can be quickly and easily customised and deployed.

What are the costs and benefits of an ECA framework? There is substantial cost in developing any generic framework, and many such frameworks have ended up with low levels of uptake. For example, UPnP and Jini have had very limited market penetration despite having benefits for home automation. The costs of any framework need to balanced by demonstrated benefits in areas such as heterogeneity, intelligence, automation and resilience.

V. CONCLUSIONS

ECA pattern-based frameworks have found significant application in domains like database management. The fact that a building automation system produces changes in the building operation based on changes in the building environment and by changes in user behaviour suggests that an ECA-based system may produce a suitable framework for both describing the desired system operation, and also linking to an event-based networking structure for communicating with resource-limited sensors and actuators.

A service-oriented architecture provides a possible mechanism for separating out the event-signalling functions of sensors from the details of the underlying hardware, protocols and message encodings that provide those event notifications. Significant work still needs to be done to design both the heterogeneous networking infrastructure and the rule-based expert-system controllers.

So far, this work is mostly at the proposal stage. Substantial work still remains to implement the ideas. One of the most significant challenges is identifying exactly what language constructs are needed to adequately, intuitively and concisely describe the desired behaviour of a typical smart house. The next step in this research is to implement a simple case study application.

REFERENCES

[1] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol 1, 2009. pp. 101–105.

[2] (2012) Internet of Things – Architecture Website, [Online]. Available: http:// www.iot-a.eu

[3] (2012) IERC - European Research Cluster on the Internet of Things, Website, [Online]. Available: http://www.internet-of-things-research.eu/

[4] A.P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, M. Zorzi, "Architecture and protocols for the Internet of Things: A case study," *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshop,* pp.678-683.

[5] N. B. Priyantha, A. Kansal, M. Goraczko, and Feng Zhao. 2008. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08).* pp. 253-266.

[6] A. Kamilaris, V. Trifa, and A. Pitsillides, "HomeWeb: An Application Framework for Web-based Smart Homes." *Proceedings of the 18th International Conference on Telecommunications, Ayia Napa, Cyprus*, May 2011

[7] J. Bailey, A. Poulovassilis, and P. T. Wood. "An event-condition-action language for XML". *In Proceedings of the 11th International Conference on World Wide Web (WWW '02).* pp. 486-495.

[8] Zakir Laliwala; Rahul Khosla; Pritha Majumdar; Sanjay Chaudhary; , "Semantic and Rules Based Event-Driven Dynamic Web Services Composition for Automation of Business Processes," *Services Computing Workshops, 2006. SCW '06. IEEE* , vol., no., pp.175-182, Sept. 2006

[9] Alam, S.; Chowdhury, M.M.R.; Noll, J.; , "SenaaS: An event-driven sensor virtualization approach for Internet of Things cloud," *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on* , vol., no., pp.1-6, 25-26 Nov. 2010

[10] Seel, C.; Schimmelpfennig, J.; Mayer, D.; Walter, P.; , "Conceptual modeling of complex events of the Internet of Things," *eChallenges, 2010* , vol., no., pp.1-8, 27-29 Oct. 2010

[11] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated Discovery, Interaction and Composition of Semantic Web Services," Journal of Web Semantics, 2003.

[12] D. Valtchev, I. Frankov, , "Service gateway architecture for a smart home," *IEEE Communications Magazine*, vol.40, no.4, pp.126-132, Apr 2002

[13] Chui Leong; A.R.; Ramli, T. Perumal, , "A rule-based framework for heterogeneous subsystems management in smart home environment," *IEEE Transactions on Consumer Electronics*, vol.55, no.3, pp.1208-1213.