Towards a Life Cycle for IoT Applications Development

SUBMISSION DATE / POSTED DATE

04-07-2021 / 08-07-2021

CITATION

Prakash, Deepika; Prakash, Naveen (2021): Towards a Life Cycle for IoT Applications Development. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14906277.v1

DOI

10.36227/techrxiv.14906277.v1

Towards a Life Cycle for IoT Applications Development

Naveen Prakash, Deepika Prakash

Abstract: **An IoT system is specified in terms of sensors/actuators and communication between them. However, we argue for performing upstream activities of the Systems Development Life Cycle during IoT application development. We propose the conceptual design stage followed by conversion to an IoT system and show that we need concepts for autonomy, perception, input processing, changing the external world, maintenance of historical information and communication. To handle these, we use the notion of communicative agents, COMMAGs and develop the Communicative Agent Model. We show conversion of this model into the IoT system to be.**

Keywords: Agent, Communication, Reliability, Autonomy, Conceptual Design, IoT Applications

I. INTRODUCTION

The term IoT system refers to [1] a 'composition of Devices and Services interacting with other Devices, Physical Entities, and Users'. From this perspective, IoT application design is a specification of this composition and interaction. There are two broad approaches to this specification, UML based [2, 3, 4, 5, 6] and SysML based [1, 7, 8] respectively. The former follows the software engineering approach and exploits notions of components, ports etc. of component diagrams of UML. The latter adopts the system engineering approach and uses notions of blocks and block interaction through connectors and ports of SysML.

In assuming the existence of interacting devices and services, an IoT system does not address the issue of where devices/services of the system come from, and what real world application it supports. That is, the larger organizational purpose behind developing an IoT system is not investigated. This purpose lies with domain stakeholders like controllers of power grids, residents of smart homes, health care providers and so on. A conceptual design that meets this purpose leads to the interacting devices/services/users.

N. Prakash is with the Department of Information Technology, IGDTUW, New Delhi 110006 India (e-mail: praknav@hotmail.com).

D. Prakash is with the Department of Computer Science and Engineering, NIIT University, Neemrana, Rajasthan 301705 India (e-mail: dpka.prakash@gmail.com).

Thus, there is a case for performing upstream activities of the Systems Development Life Cycle, SDLC during IoT application development. Accordingly, our long-term research goal is to develop the requirements engineering and conceptual design phases for IoT applications. In this paper, we consider conceptual design and its conversion to an IoT system. We adopt the approach of OMG's Model Driven Architecture, MDA in doing this.

The concepts used to express designs flow from the differences between traditional Information Systems, IS and IoT based ones that we refer to here as ISoT, Information System of Things. The differences are presented in Table I.

The first row of Table I says that the IS receives input from external actors whereas an ISoT perceives input through its sensors. The second row of the table says that compared to a IS where input is relatively reliable, the input to an ISoT is unreliable because sensors may fail or produce false positive/negative data. The third row tells us that in an IS the input is processed rather simply, merely to check input validity. However, considerably more effort is required in an ISoT for data cleaning from redundant sensors to deal with unreliable data.

TABLE I
Differences between and IS and ISoT

Property	IS	ISoT
Input	From external actors	Perceived by system itself
Reliability of Input	Relatively Reliable	Unreliable
Input Processing	Simple	Complex
Output	Message to external world	Change the external world
Behaviour	By invocation	Autonomous
Data	Current	Historical
Interaction	Control flow	Data communication

The fourth row says that the output of IS is a message to external actors (say, an acknowledgement of transaction occurrence) whereas the ISoT changes the external world itself through its actuators, for example, by turning off/on appliances. As shown in the fifth row, features of an IS are invoked i.e., a function when called must execute. In contrast, in an ISoT, there is no request for carrying out an operation. Instead, the data is sent and it is left to the receiver to autonomously decide what it wants to do with it. As shown in the sixth row, data in an IS is primarily current, most recently updated data to reflect the last transaction carried out. In an ISoT however, data may be

historical, say, a month of pictures from a security camera. Finally, an IS relies on interaction that is essentially "flow of control" between objects/entities but the ISoT relies on communicating data using different electronic communication standards like Bluetooth and wi-fi as shown in the seventh row.

Thus, we need concepts for expressing system autonomy, perception of input, input processing, changing the external world, keeping historical data and communication. We take recourse to the notion of agents of agent-oriented software engineering and propose a subclass of agents called COMMuncative Agents, COMMAG.

In the next section we present the COMMAG model with examples to illustrate its concepts. We compare and contrast it with notions found in traditional conceptual models. Thereafter, in section III, we propose an application-oriented model and present an approach to convert a COMMAG schema to the device level. In section IV, we present a case study. In section V, we compare our approach with other approaches and finally conclude our work in section VI.

II. THE COMMAG MODEL

Agent-oriented software engineering recognized the inability of object-orientation to deal with autonomy and perception [9, 10]. Instead, the notion of an agent was put forward. An agent [10] is "situated within or part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." Notice that the property of perception is built into this view. Agents can be classified based on five properties, namely, being communicative, learning ability, mobility (ability to transport itself from one machine to another), flexibility and having a character. Out of these, the property of being communicative is relevant in the IoT domain. Consequently, the notion of a COMMunicative AGent or COMMAG is at the core of the model shown in Fig. 1. The figure is in UML notation.

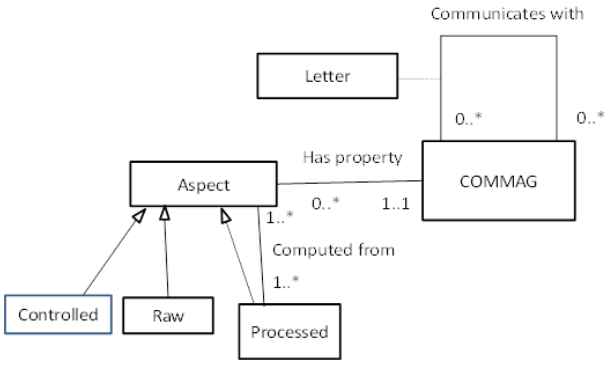

Fig. 1: The COMMAG Meta-model

It is possible to classify COMMAGs in three ways (not shown in Fig. 1 to avoid graphical clutter) as follows:

- Based on Nature: COMMAGS are either physical (e.g., a milk van), that we can feel and touch, or virtual, that can be conceptualized. Thus, a milk van, divided into sections for better temperature control, gives the virtual COMMAG, Section.
- Based on Structure: COMMAGs are one of atomic, complex, or abstract. An atomic COMMAG cannot be decomposed into simpler ones. A complex COMMAG is composed of more than one constituent COMMAGs. Thus, the COMMAG milk van can be viewed as composed of various section COMMAGs. Finally, an abstract COMMAG is a generalization of other COMMAGs.
- Based on Role: A COMMAG may be a nodal processor, a gateway, or a simple COMMAG. The nodal processor collates and processes data from several sources, perhaps from itself as well. For example, the nodal processor Milk-Van collates temperature from several COMMAGs, Section to compute the average temperature of the milk van. A gateway COMMAG is at the boundary of the ISoT with its environment (the cloud or humans). It may double up as a nodal processor as well. A simple COMMAG is neither a nodal processor nor a gateway.

The three classifications proposed above are overlapping. For example, the Refrigeration Unit is a physical, atomic, and simple COMMAG. Notice that classification on "role" is not found in traditional conceptual modeling.

Fig. 1 shows that a COMMAG has zero, one or more aspects. In contrast to traditional conceptual modeling where attributes are properties of entities/objects and attribute values are changed by transactions, aspects here are parameters that a COMMAG perceives and/or acts upon to change what it shall perceive in future. Thus, the COMMAG Milk Van has an aspect, temperature of the van that it senses. Switch position is an aspect that the COMMAG Refrigerator acts upon to change the temperature.

Similar to attributes in conventional conceptual modeling, aspects of a COMMAG may be single/multi-valued. However, the phenomena giving rise to these are specific to IoT conceptual design. Multi-valued aspects here occur because of

- Reliability: Redundant measurement may be introduced to account for sensor failure/malfunction. This yields multiple values for the same aspect and is processed to yield the "right" single value. In IS conceptual modeling, however, all values of a multi-valued attribute are correct and exist separately.
- Multiplicity: A COMMAG may receive values of the same aspect from several sources. For example, the Milk Van receives temperature values from several Milk Van Sections. Thus, the Milk Van has a multi-valued aspect, Section_temperature.

Now, consider raw, controlled, and processed aspects of Fig. 1. Values of raw aspects are directly sensed by

COMMAGs, for example, temperature sensed through a temperature sensor. Controlled aspects are those over which a COMMAG exercises supervision, for example switching of refrigeration. A controlled aspect takes values from the set {Switched, Stepped, Continuous, Timed}. Switched implies ON/OFF, stepped represents discrete steps like in a 3-speed fan, Continuous is variable as in a dimmer, and timed is based on an event occurrence like, time of the clock or a time-out. Values of processed aspects are obtained by some computation carried out by the COMMAG, for example, (a) obtaining the "correct" value from multiple unreliable sources, (b) obtaining a value, say the average value, from multi-valued aspects, (c) unit conversion, (d) computing a new value etc. Notice that (a) is not found in TS conceptual modeling.

Communication is shown in Fig. 1 by the "communicates with" relationship between COMMAGs. As shown, a COMMAG may communicate with zero or more COMMAGs. With zero, the model allows a stand-alone COMMAG like a garden lamp that senses ambient light and lights up or switches off.

The data communicated, Letter in Fig. 1, consists of one or more aspects, raw or processed. For example, the COMMAG, Section of a milk van sends the value of its raw aspect temperature as a letter to the COMMAG, Milk Van, and continues to sense the temperature. Notice the difference between a message of UML and a letter. In UML, a message is either arguments of an operation call or attributes of a signal. On the other hand, a letter contains data, either directly sensed or processed by the sending COMMAG. No operation is invoked or signal sent. The receiving COMMAG autonomously decides what to do with the contents of the letter.

Diagrammatically we represent a COMMAG as a rectangle with three, possibly empty, parts Raw Aspect Part, RAP; Controlled Aspect Part, CAP; and Processed Aspect Part PAP, for holding raw, controlled, and processed aspects respectively. A double headed arrow represents communication between COMMAGs and is labeled with the sent letter. The direction of the arrow identifies the sender and the receiver.

A raw/controlled aspect is expressed by its name and whether it is single valued (the default) or multi-valued. A processed aspect is expressed by its name, the function name that computes it, and function arguments. For example, the Milk Van has the processed aspect, Av_temp Average(Temp_list). Here, Av_temp is the name of the aspect; the function Average computes average from Temp_list, a list of temperatures and returns it in Av_temp.

We illustrate the foregoing by a milk transporting company having a number of milk vans. A milk van is large and there is uncertainty about whether temperature measured at one place holds for the entire van. For this reason, the van is partitioned into a number of regions and the average temperature of the van is computed from the regional temperatures. Since the cold chain cannot be broken, reliability of temperature is important and several measurements are made in each region. A record of the van temperature is kept for post-mortem analysis of the milk van. This is done by a time stamp and is not considered in this paper since it is as usual.

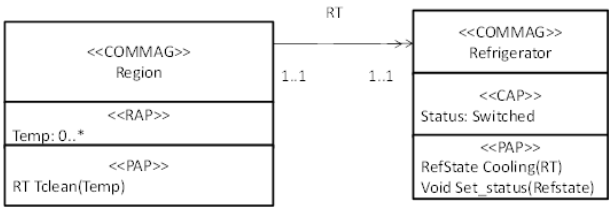

Fig. 2a: The Cooled Unit

We show the design in two parts, the first dealing with Cooled Unit and the second with the Milk Van and communication with the cloud. The Cooled Unit, see Fig. 2a, is a complex COMMAG built over a region and its refrigerator. We do not show this since its representation as an aggregation in UML is well known. The COMMAG Region has a multi-valued aspect, Temp (for temperature) to ensure reliability. This is represented in the RAP part using UML notation. Its PAP consists of aspect RT, Reliable Temperature. This is computed by the function Tclean that produces cleaned up data from Temp. The Refrigerator is another COMMAG. It keeps the status in its CAP. The PAP has an aspect for the state of the refrigerator, Refstate that is computed by the function Cooling having RT as argument. If the Status and Refstate are different then Status is set to the value of Refstate by Set_Status. The cardinality of communication shows that exactly one RT value is communicated and that a refrigerator receives RT value from exactly one Region.

Now, consider Fig. 2b that shows the milk van structurally composed of Cooled Units and receiving RT from its several Cooled Units as shown by the cardinality of the communication link between MilkVan and Cooled Unit. MilkVan is a nodal processor (<<NP>> in the figure) that computes average temperature of the van. For this, a processed aspect, Templist is defined. It is a list of temperatures produced by the Tcollate function that appends received RT to Templist. Further, it computes MVtemp as the average temperature from Templist, stores it in VanTemp, and sends it to the gateway, MilkVanGate. Notice that CAP, RAP and PAP of our gateway are empty.

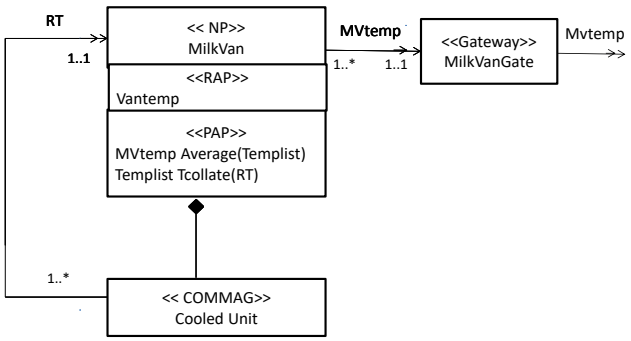

Fig. 2b: Nodal Processor and Gateway

III. THE APPLIANCE LEVEL SPECIFICATION

The foregoing brings out the data, its processing, and communication properties of the ISoT problem. This design is now to be converted into a specification of devices and inter-device communication. We propose to do this in two steps, (a) Identification of sensor types and actuator types of COMMAGs, (b) determination of appliances, device types and their specifications. This is done by building the Appliance Agent schema by instantiating the Appliance Agent Model shown in Fig. 3.

An appliance agent, AAG is a kind of communicative agent. Unlike the COMMAG that is problem specific, an AAG focuses on the types of appliances and is a non-empty aggregate of types of sensors, actuators and apps.

Structurally, an AAG, similar to a COMMAG, may be simple, complex, or abstract. Thus, we can now consider our milk van region as a complex AAG composed of two simpler AAGs, one having the temperature sensor and another with an ON/OFF actuator.

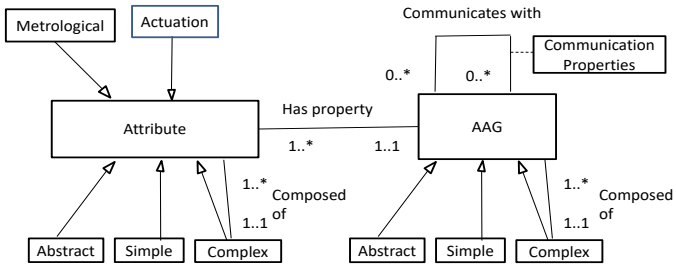

Fig. 3: The Appliance Agent Model

The specification of an AAG is made in terms of its attributes (NOT aspects) and communication properties. AAG attributes, similar to UML attributes, describe properties of an AAG. For sensors, these are metrological properties, features and characteristics that bear on its sensing capabilities like [11] "accuracy, and precision. For actuators these are actuation properties [12] like latency and power consumption.

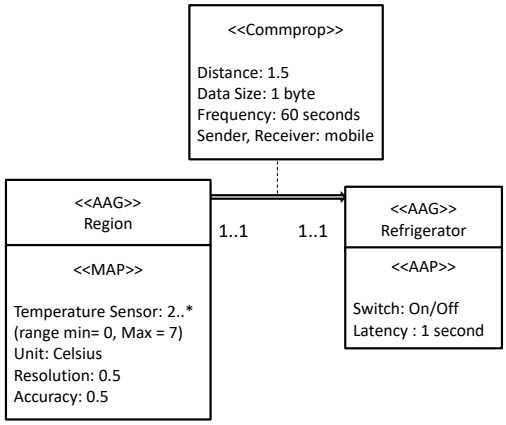

Fig. 4: AAM Schema of the Cooled Unit

The 1: N 'Has property' relationship between an AAG and Attribute expresses AAG properties. Again, from the structural perspective, attributes may be (a) simple, (b) complex, or (c) abstract. As an example of a complex metrological attribute, let the temperature sensor have attributes, range, accuracy, and precision. Of these, range is a complex attribute composed of minimum temperature and maximum temperature.

Fig. 3 also shows that an AAG can communicate with zero or more AAGs. Communication attributes constitute the specification of the required communication, for example, distance, data rates and whether the sender/receiver are mobile or static.

Fig. 4 shows an Appliance Agent schema for the Cooled Unit of Fig. 2a. It uses the same names for AAGs as the COMMAGs. The Metrology Attribute Part, MAP of the Region says that there must be a minimum of 2 Temperature Sensors. The attributes of sensors are determined by the spoiling characteristics of milk. The range of measurement is from 0 to 7 degrees in Celsius. Further, the resolution of the sensors must be half a degree and its accuracy must be half a degree at 4 degrees Celsius.

Communication between the two AAGs is restricted to 1.5 metres which is the distance between the temperature sensor and the switch. Fig. 2a says that RT is to be sent. Its data size is ascertained to be one byte. Further RT is sent every 60 seconds. Finally, communication is mobile since both the region and the refrigerator move along with the milk van. This gives us the communication properties, <<Commprop>>.

A. Conversion to Device Level

As mentioned earlier, conversion is done in two steps. These steps are as follows:

Step 1: COMMAGs and their Sensors/Actuator type

Selecting COMMAGs for conversion to AAG

a. For each ISA hierarchy, pick up every leaf COMMAG.
b. Pick up each component and constituent COMMAG of a complex COMMAG.
c. Pick up each simple COMMAG

Determining Sensor/Actuator/App

This step requires interaction between schema designers and device experts.

i. Define a sensor type in the selected COMMAG for each of its raw aspect.
ii. Define an actuator type in the selected COMMAG for each of its controlled aspect.
iii. For each gateway COMMAG that interacts with humans, postulate an app.

Step 2: Build the AAM schema

This step needs interaction with device and appliance experts to identify the needed AAGs from the device types and apps already determined.

IV. CASE STUDY

The aim is to control cooling, lighting, the irrigation system of the garden and maintain security of a house. The house consists of several bedrooms, bathrooms, kitchen, living room, porch and garden. There are several residents who interact with the house when they are travelling in addition to when they are in residence. The house is as follows:

1. Light in the bedroom and living room is regulated based on ambient luminosity.
2. Bathroom light is on so long as the bathroom is occupied and ambient light is low.
3. Kitchen light is on if the kitchen is occupied and ambient light is low
4. Porch and Garden lights are on when ambient light is low and are switched off at night at a pre-set time.
5. Bedroom and Living room are equipped for cooling during early summer based on air temperature and humidity.
6. Bedroom and Living room are also to be cooled during full summer again depending on humidity and temperature.
7. Any movement at the porch must be captured in pictures
8. Gardens have a drip irrigation system based on soil humidity levels.

The COMMAG schema for the foregoing is shown in Fig. 5. Conceptually, the smart home consists of interiors and exteriors with the former comprising the various rooms and the latter comprising the porch and garden. An abstract COMMAG Room is defined in Fig. 5 that has Ambient light as its RAP and Light control as its CAP. Room is specialized into four COMMAGs which inherit its RAP and CAP. These specializations are (a) Bedroom having moderate and high Temperature, Mod Temp and High Temp as its RAP and humidity control and temperature control, Humidity Control and Temp Control respectively as CAP, (b) Kitchen, having Occupancy as its RAP, (c) Bathroom which has occupancy as its RAP, and (d) Living Room, which is a nodal processor with an empty RAP and it computes average temperature and average humidity as seen in its PAP.

The living room is split into a number of regions represented by the COMMAG Living Room Region which has moderate and high temperature, Mod Temp and High Temp as RAP and humidity control and temperature control Humidity Control and Temp Control respectively as its CAP. The Living Room Region communicates its RAP values to the nodal processor as shown.

The figure shows another abstract COMMAG, Exterior which has Ambient light as its RAP and Light status as CAP. This is specialized into COMMAGs Garden and Porch. Garden consists of COMMAG, Plant Bed that has Soil saturation as its RAP. The COMMAG Tank represents the water reservoir. It has Water Level as its RAP and Level control as its CAP. Plant Bed sends Soil Saturation to Tank that decides to regulate water supply by operating Level control. Finally, the COMMAG Porch has occupancy and Movement as its RAP and Movement control as its CAP.

The smart home has the gateway, Resident for interaction with residents of the house. It sends desired values of ambient light, temperature etc. to the ISoT.

Applying Step 1 of Conversion Process

From step 1a of section III.A, we obtain four COMMAGs for conversion to AAG from the hierarchy of Room; and two from that of Exterior. From step 1b we get two for Living room and Living room region respectively. However, the former is already defined in step 1a and so we get only one additional COMMAG. Similarly, from step 1b we get Plant bed. By step 1c, we get Tank and Resident.

Further, from step 1(i), 1(ii), and 1(iii) we get the various device types as shown in Table II.

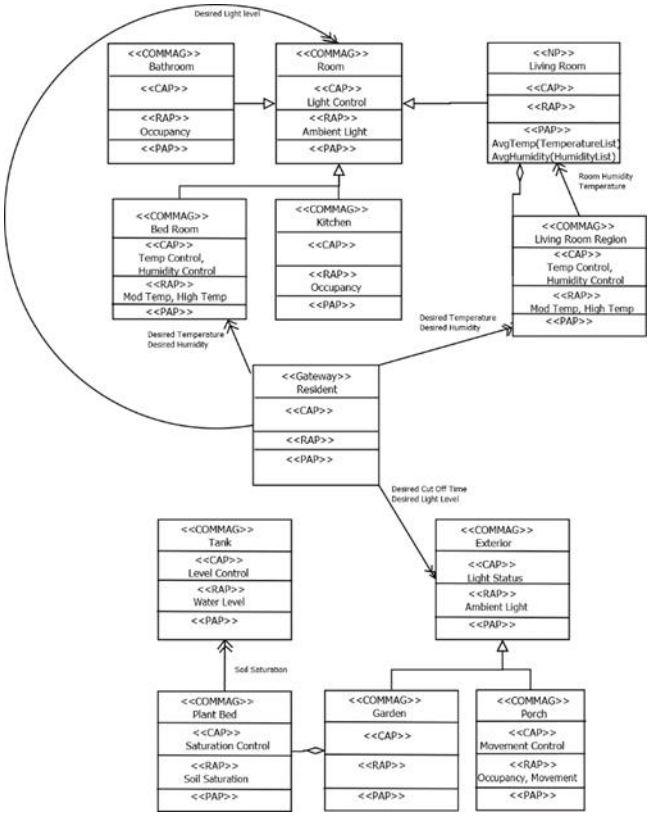

Fig 5: The CAM Schema of the House

Applying Step 2 of Conversion Process

Here we build the AAM schema. Since the schema is diagrammatically large, we are presenting its essential details in Table III. When dealing with humidity and temperature, the appliance and device expert suggests that fans should be used for moderate temperature whereas for high humidity and high temperature air conditioning is recommended. These get reflected in Table III as AAGs, Smart Fan and Smart Air Conditioner respectively.

TABLE II
COMMAGs and their Device Types

AAG	Sensors	Actuators	App
Bathroom	Light Sensor Occupancy Sensor Clock	Light Control	
Bedroom	Light Sensor Moderate Temperature Sensor High Humidity Sensor High Temperature Sensor	Light control Moderate Temperature Control High Humidity Control High Temperature Control	
Kitchen	Light Sensor Motion Sensor	Light Control	
Living Room	Light Sensor	Light control	
Living Room Region	Moderate Temperature Sensor High Humidity Sensor High Temperature Sensor	Moderate Temperature Control High Humidity Control High Temperature Control	
Resident			Home
Porch	Light Sensor Clock Motion Sensor Image Sensor	Light Control	
Garden	Light Sensor Clock	Light Regulator	
Plant Bed	Soil Moisture Sensor		
Tank	Level Sensor	Outlet Control Inlet Control	

For imaging, the appliance to be used is a Smart Camera. This leads the designer to define another AAG with its attributes as shown. Similarly, we have the Plant Bed as an AAG that is for soil moisture measurement as well as for the tank of water with its own level sensor and outlet/inlet actuators. Lastly, there are different kinds of lights as shown in Table III. The timer switches switch on light for a specified duration. Light shutdown switches off lights at a fixed time, 11pm.

Communication is between the gateway, Resident and other AAGs only. The properties of communication are as follows:

Resident: mobile, Distance: 10 Kms, data size: 2 bytes

The data size is enough to hold any data that the gateway wants to send to/ receive from the AAGs of the system.

TABLE III
AAGs, Their Devices and Attributes

AAG	Sensor(Attributes)	Actuator(Attributes)
Smart Fan	Temperature sensor(Range : (min:15 Celsius, max:32 Celsius) Resolution 1 degree	Fan Speed Regulator: 3 step Latency : 1 sec
Smart Air Conditioner	Temperature Sensor(Range: (min:15 Celsius, max : 50 Celsius) Resolution 1degree Humidity Sensor(Range: min:30, Max: 60) Resolution 5 percent	Mode Regulator: 5 modes Temperature Regulator: 5 step Humidity Regulator: 5 step Air Speed Regulator: 3 step Compressor Switch: Boolean
Smart Camera	Image Sensor(Resolution: (length 3264 pixels, breadth 2448 pixels) Number of Pixels: 8MP Crop factor: 0.8)	
Plant Bed	Soil Moisture Sensor(	

	Range: (Min:0%, max: 70%) Accuracy +/- 3%)	
Tank	Level Sensor (min: 0.1 metre max: 1.5 metre)	Outlet switch: Boolean Inlet switch: Boolean
Bedroom light Type	Light Sensor(Intensity:(min: 1 lux Max: 200 lux)	Light Regulator: 5 step
Living Room Light Type	Light Sensor(Intensity:(min: 1 lux Max: 200 lux)	Light Regulator: Dimmer
Bathroom Light Type	Light Sensor(Intensity:(min: 1 lux Max: 200 lux) Motion Sensor (Detection range: 3 metres Detection Field: 360 degree) Timer (Range: min 0; max 5 minutes)	Light Timer Switch: 2 min
Kitchen Light Type	Light Sensor(Intensity:(min: 1 lux Max: 200 lux) Motion Sensor(Detection Range: 4 metres Detection Field: 360 degrees	Light Timer Switch: 10 minutes
Porch Light Type	Light Sensor (Intensity (min: 1 lux, max: 100 klux) Motion Sensor(Detection Range: 2 metres Detection Field: 90 degrees	Light Timer Switch: 1 minute Light shutdown: 11 pm
Garden Light Type	Light Sensor (Intensity (min: 1 lux, max: 100 klux) Motion Sensor(Detection Range: 2 metres Detection Field: 90 degrees	Light Duration Time: 1 minute Light shutdown: 11 pm

*Units of attributes added to make them understandable
Resident not shown since it is an app

Observations

From our case study, we observe that

1. Designer focus was on RAP and Cap follows thereafter. CAP determination may happen once RAP is known. For example, once water level is determined, as a RAP of the water tank, only then is outlet and inlet control seen. Additionally, the RAP of one COMMAG may help in determining the CAP of another. This happens in our Milk Van example: the temperature of a Milk Van region determines that the Refrigerator COMMAG needs to be controlled.
2. A multi-valued attribute was found in our case study but this was on account of multiplicity and not reliability.
3. Communication in our case study was limited to/from gateway to other AAGs.
4. Appliance selection affects ISoT aspects. Consider cooling: fans do not change air humidity; desert coolers increase it; and air conditioners control it. Further, external appliance properties must be considered, for example, power consumption, weight, and size in appliance selection.

We need to apply our technique to other ISoTs where 2 and 3 above do not exist. We also need to see how well our approach scales up.

V. COMPARISON

As against notions of a block of SysML and objects of UML, our proposal uses communicative agents.

- A block represents properties, behavior and constraints of system components. Interaction between blocks is specified by connectors and ports that allow for input/output as well as messaging for service invocation.
- An object is that which has space and shows behaviour. Objects interact with one another through message passing.
- A communicative agent is autonomous, perceives its environment as data, receives/transmits data from/to other agents, and decides what action, if any, is to be taken

In using SysML, Reference [7] treats sensors as blocks. There are methods for monitoring, recording, collecting and pushing sensed data to the gateway. A gateway is a central information hub that connects to a repository of data. It is a block that has methods to receive, do some pre-analysis, and transmit data. This reliance on method invocation does not directly capture that a sensor perceives its environment autonomously. In [7], the representation of actuators is not considered.

[1] use their SySML4IoT profile. Sensors and actuators are blocks. Access points to these are SySML ports. Again, block interaction is by message passing and flow properties of SysML ports. As for [7], the perception of the environment by a sensor is not directly captured. The notion of a gateway is not considered in [1].

Our COMMAG perceives its environment and therefore directly captures sensor measurements. Gateways are agents that lie at the boundary of the ISoT and its external world. Further, we see Nodal Processors, a type of COMMAG, as conceptualizing edge/fog computing. There is no explicit model component in either [7] or [1] for this.

An agent acts on behalf of someone else. Our COMMAG acts on behalf of real-world phenomena, a milk van, an air conditioner etc., and establishes a linkage between these and their raw, controlled and processed information. The linkage ensures that the ISoT is modelled in terms of these phenomena and their inter-communication to meet the purpose of the ISoT. In contrast, both [7] or [1] stay at the device level.

Reliability is seen in a COMMAG as a property of the ISoT to be enforced. This allows multi-valued aspects that in the AAM become multiple instances of the same device type. The PAP of a COMMAG allows for data cleaning and computation locally within the agent, thereby allowing edge and fog computing. When transmitted out of gateways, then computing in the cloud can occur. In both [7] and [1],

reliability is not considered. Further, in [7], pre-analysis is possible in the gateway and processing is left to be done in the cloud, whereas in [1] processing is not considered.

The specifications made in the COMMAG model are converted into application agents as AAGs. Here, device properties and properties of communication are identified. Thus ranges, precision, delays, distances, data sizes etc. are considered. In both [7] and [1], there is no conversion to the device level and device types are treated as given. However, it is possible to specify device properties as attributes of blocks. Communication and its properties, data sizes, distances, are not expressed in either of them. Consequently, determination of communication standards, Zigbee, Bluetooth, etc. does not follow from the design specification.

In [2] a UML based language is proposed. In their language, a thing corresponds to a real-world phenomenon like a Kitchen and consists of three kinds of items, namely, inputs (sensors), outputs (switches, SMS to be sent) and components (software). These are connected together by interfaces.

Being UML based, a thing is an object and follows the message passing approach. In contrast a COMMAG is an agent of a real-world phenomenon like kitchen, milk van etc. and enters into communication. The language of [2] does not elaborate on issues of redundancy due to reliability, edge/fog/cloud computation, gateways and nodal processors. It also does not propose a conversion to the device level where device types and their properties along with attributes for determining communication standards are specified.

VI. CONCLUSION

Our main concern, expressed in the Introduction of this paper, is that an IoT system does not address the issue of where devices/services of the system come from, and what real world application it supports. This is because an IoT system focuses on devices and their interaction. Therefore, we proposed that upstream activities in the SDLC should be considered to lay a firm basis for and IoT system.

The upstream activity we propose is in two parts, a higher level abstraction of the problem at hand and its conversion into a lower level expression of devices and communication as well as their properties.

In our approach the Information System of Things, ISoT, **is a teleological collection of communicating agents**. We have chosen agents because their definition [10] fits well with the nature of an ISoT. Further, communication is essential to an ISoT. Therefore, we postulate a communicative agent, COMMAG as a central concept for an ISoT.

Being an agent, a COMMAG acts on behalf of real-world phenomena like rooms, garden, milk van etc. This encourages an expression of the ISoT in terms that stakeholders are familiar with. The result is a COMMAG schema that reflects the problem at hand and acts as a specification for IoT system designers. It is independent of any implementation platform.

The second step in our approach is to move to the device/communication level. We do this by converting from the COMMAG schema to our AAM schema.

Our future work is to move still more upstream in the SDLC to the requirements engineering level.

REFERENCES

[1] Costa B., Pires P. F., Delicato F. C., , "Modeling IoT Applications with SysML4IoT," 42nd Euromicro Conference on Software Engineering and Advanced Applications, 157-164, 2016

[2] Eterovic T. M., Kaljic E., Donko D., Salihbegovic A., Ribic S., An Internet of Things Visual Domain Specific Modeling Language based on UML, XXV Intl. Conf. on Information, Communication, and Automation Technologies, 1-5, 2015

[3] Fleurey, F., Morin, B., Solberg, A., & Barais, O., MDE to manage communications with and between resource-constrained systems, International Conference on Model Driven Engineering Languages and Systems (pp. 349-363), 2011.

[4] Open Mobile Alliance, Lightweight Machine to machine Requirements, Version 1.1, 2018

[5] Prehofer C., Chiarabini L., From Internet of Things Mashups to Model-Based development, COMPSAC Workshops, 499-504, 2015

[6] Thramboulidis K., and Christoulakis F., UML4IoT- A UML based approach to Exploit IoT in Cyber-physical manufacturing systems, Computers in Industry, Elsevier, 82, 259-272, 2016

[7] Kotronis Ch., Nikolaidou M., Dimitrakopoulos G., Anagnostopoulos D., Amira A., and Bensaali F., "A Model-based Approach for Managing Criticality Requirements in e-Health IoT Systems," in 13th Annual Conference on System of Systems Engineering (SoSE), 60-67, 2018

[8] OMG Systems Modeling Language, OMG, Version 1.6 doc. No. formal/19-11-01, 2019

[9] Shehory O. Sturm A., A Brief Introduction to Agents, in Agent-oriented Software Engineering, Shehory O. Sturm A (eds.), Springer Verlag, 2014

[10] Wooldridge M., and Ciancarini P., Agent-Oriented Software Engineering: The State of the Art, Intl Conference on Agent-Oriented Software Engineering, 1-28, 2000.

[11] Semantic Sensor Network Ontology, 2017, [Online] Available: https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/

[12] Seydoux, N., M. B. Alaya, K. Drira, N. Hernandez, T. Monteil, and O. Haemmerlé. "San (semantic actuator network)." (2016).