



# A Goal-Oriented Requirements Engineering Approach for IoT Applications

Deepika Prakash<sup>1</sup><sup>a</sup> and Naveen Prakash<sup>2</sup><sup>b</sup>

<sup>1</sup>Department of Computer Engineering, JK Lakshmipat University, Jaipur, India

<sup>2</sup>ICLC, 21/8 S. Bhagat Singh Marg, New Delhi 110001, India

**Keywords:** Functional Objective, Communication Objective, Sender, Receiver, Goal Reduction.

**Abstract:** Requirements Engineering of IoT systems has the twin objectives of specifying functionality as well as communication objectives of the system. Existing goal-oriented and use case approaches were not developed to bring out communication objectives of systems. Consequently, when these techniques are applied then communication remains subordinate to functionality. We integrate both objectives in the notion of a GOT, GOal of Things. The GOT model represents the structure of GOTs and an instance of this model is the requirements specification of an IoT. The accompanying GOT Process provides three ways of GOT reduction. We illustrate its application to an Accident Reporting System. The GOT proposal is compared with a use-case oriented approach and a goal oriented approach.

## 1 INTRODUCTION


With the rapid growth of IoT applications like for homes and power grids (Stankovic, 2014), dynamic cargo monitoring (Focker, 2015), people with special needs (Ferati, 2016), connected cars (Mikusz, 2017) etc., it was found necessary to formulate ways of representing an IoT system. The term IoT system refers to (Costa, 2016) a 'composition of Devices and Services interacting with other Devices, Physical Entities, and Users'. A number of techniques were developed to specify this composition (Costa, 2016, Eterovic, 2015, Fleurey, 2011, Kotnis, 2018, Prehofer, 2015).


A conceptual modeling layer was placed (Prakash, 2021, Prakash, 2022) on top of the IoT system. This highlighted the data of interest as well as data communication needs, in an IoT application. The salient feature of this conceptual model is its reliance on COMMunication AGents, COMMAGs that have aspects. Agents represent real world entities that are participants in IoT applications. Aspects capture the measurable quantities like temperature and pressure that are sensed by agents, possibly processed, and communicated to other agents.

Moving further upstream, proposals have also been made for requirements engineering of IoT applications. Both Functional and Non-functional Requirements, FR and NFR respectively, have been addressed. The approach of (Costa, 2017) determines four parameters, namely, Location, Sensed Variable, Sensing Rate, and Sending Rate. For (Costa, 2017) these constitute FRs. NFRs are quality attributes of the system to-be, for example, reliability. Requirements are expressed as propositions in natural language in Requirements Diagrams of SySML.

Kotnis et al. (Kotnis, 2018) concentrate on NFRs. They start with the assumption that devices are already known in their Health application. Devices are represented in Block Definition Diagrams, BDD of SySML. Interaction among devices is expressed as Internal Block Diagram, IBD of SySML. Thereafter Kotnis et al. associate BDD blocks with the requirements they must meet. For this, three categories of requirements are defined for health systems, namely, non-critical, safety-critical, and mission-critical requirements. Requirements falling in each category are identified.

Use cases/Scenarios and Goals techniques originally developed for determining functionality of Software Engineering/Information Systems, have

<sup>a</sup> <https://orcid.org/0000-0001-8404-3128>

<sup>b</sup> <https://orcid.org/0000-0003-1644-5613>

also been used. Whereas FRs can be determined using these, special provision must be made for determining the sensing/actuating capacity of things as well as communication among devices.

(Meacham, 2016) use a combination of use case diagrams for system functionality and Requirements Diagrams of SysML for things and thing-interaction. There are two levels of analysis, a high level and low level. Each high-level use case is expressed as blocks in RD and the action they perform is expressed in text in these RD blocks. Low-level use case diagrams are for representing low-level functions comprising the high-level functions. Again, the action carried out by these is expressed as text in structural blocks of RD. Movement to system design starts off from the high-level RD by postulating system blocks for each RD block. However, the manner in which this is to be done is not elaborated in (Meacham, 2016).

(Mezghani, 2017) organize their requirements engineering approach in two parts, requirements identification and requirements formalization. In the former the elicited FRs are expressed as use cases. The latter is organized in a number of design patterns in multiple levels. This proposal aims to assist system architects rather than system developers per se.

Reggio (Reggio 2018) builds a goal hierarchy of strategic and operational goals. Goals are then expressed in UML. Operational goals are associated to technological goals that express communication and device needs. However, the technological aspect is not elaborated in (Reggio, 2018) who state that it is "preliminary".

The approach of (Takai, 2019) is to use the translate GQM<sup>+</sup> approach. Business goals become requirement nodes, questions become text of the requirement node and metrics become stereotyped requirement nodes, stereotyped as performance requirements. These are then converted into Requirements Diagram of SysML. The manner in which things and thing interaction is represented is not articulated in (Takai, 2019).

From the foregoing, we conclude that there are two parts to Requirements Engineering in the IoT domain (a) determining function is to be performed or what is to be done by a thing and (b) what is to be sensed and interaction among things. The former is de-emphasized in (Costa 2017, Kotnis, 2018) whereas the latter is of subordinate concern in (Meacham, 2016, Mezghani, 2017, Reggio, 2018). Yet, we believe that thing functionality, its sensing/actuating capacity and thing interaction form one integrated whole. By separating these, either thing functionality is not articulated as in (Costa

2017, Kotnis, 2018) or the sensing/interaction aspect is not fully elaborated.

It is with a view to obviate this, that we present our integrated approach to IoT requirements Engineering. Broadly speaking, our proposal is as follows. We base our proposal in goal-orientation and refer to GOals of IoT as GOT. We say that a GOT is achieved when the functional intention behind it is **fulfilled and communicated**. Goal fulfilment is for the function/processing to be performed by a thing and goal communication is for the sensing/actuating and thing interaction part. Goal reduction results in the usual AND/OR hierarchy.

The layout of this paper is as follows. In the next section, we elaborate on the notion of a requirement in the IoT domain and define our notion of a goal. In section 3, the GOT model is described. The GOT requirements engineering process and the various drives are presented in section 4. In section 5, we apply our ideas to the car accident IoT. Thereafter, in section 6, we compare the proposals of (Meacham, 2016) and (Reggio, 2018) with ours. Section 7 concludes the paper.

## 2 GOALS IN IoT

A requirement has been variously defined and we adopt the definition as in IEEE standard 730-2014 as "A condition or capability that must be met or possessed by a system or system element to satisfy an agreement, standard, specification, or other formally imposed documents." When applied to the IoT domain, a thing is the system component of interest and a collection of communicating things delivers the service required by an IoT application. The "condition or capability" of a thing is that it may obtain data, optionally process it, and communicate it to another thing. Data may be obtained by sensing the environment or as a communication from another thing. There are terminator things like the cloud or users, that consume data that they receive. Thus, we can **define a requirement in the IoT domain as a statement about processing and communicating data**. For example, consider the requirement, "the life of the battery must be monitored and low-life batteries must be replaced before they die." The processing is "Estimate battery life" and the communication need is to send the message, "Low battery" to the user.

Now, let us consider how to represent an IoT requirement as a GOal Of Things, GOT. A goal has been variously defined. (Lamsweerde, 2000) defines a goal as an objective that the system should achieve

through cooperation of agents in the software-to-be and in the environment. Anton (Anton, 1994) states that goals are high-level objectives of the business, organization or system. Dardenne (Dardenne, 1993) defines a goal as a non-operational objective to be achieved by the composite system.

It can be seen that, central to the notion of a goal is that of an objective. In software engineering/information systems, an objective is the identification of the functions that the system should perform. In the IoT domain, however, the mere computation of the remaining battery life is not enough but the objective of the system is fully achieved only when this is also communicated to the user. Thus, for us, GOT is a pair

$$\text{GOT} = \langle \text{FO}, \text{CO} \rangle$$

where FO is the functional objective and CO is the communication objective. Thus, our battery requirement above is specified as a GOT as follows: Maintain Battery =  $\langle$ Estimate Battery Life, Inform battery status $\rangle$

Now, we turn to the communication aspect of a GOT. We represent a communication objective as a triple,

$$\langle \text{Sender}, \text{Receiver}, \text{Message} \rangle.$$

The message may contain the following:

- a) Sensed data. As an example, consider the temperature sensed by a temperature sensor
- b) Processed data produced as a result of FO, e.g., cleaned temperature data after taking false positives/negatives into account.

Senders and receivers may be physical or virtual. Consider the goal to keep the milk in a milk van at a temperature at 4°C. Here, the physical milk van or the cooling unit is a sender/receiver. If the milk van is divided into regions, then each region could be treated as a sender/receiver. Senders/receivers may be the cloud or a user.

Now, following goal-orientation, we build AND/OR goal hierarchies. In (Dardenne, 1993), we get the basis for this as follows:

“Reduction (G, g) iff achieving goal g possibly with other subgoals is among the alternative ways of achieving goal G.”

We adopt this for GOT as well thereby allowing G and g to be GOTs. Again, due to the two objectives of a GOT, it is possible to reduce an IoT goal along FO or CO. We illustrate these two forms of reduction through the milk transportation example having the requirement, Maintain temperature at 4 degrees centigrade. The corresponding GOT is as follows:

$$\text{Maintain\_temp} = \langle \text{Compute Avtemp}, \text{Inform} \rangle$$

Let computation of average temperature be in two parts, clean sensed data AND the actual calculation, Calculate Average. Notice, however, that in this example, the decomposition of the FO serves to clarify the manner of its achievement and that cleaned data obtained from the first sub-goal is not to be communicated. The rules governing this are given in section 3.

Now, consider reduction along CO. Average temperature is required to be sent to two receivers, (a) the cooling unit, and (b) the cloud. Thus, we can do goal reduction and get sub-goals

$$\begin{aligned} &\langle \text{Compute Avtemp}, \text{Inform Cooling Unit} \rangle \\ &\text{AND} \\ &\langle \text{Compute Avtemp}, \text{Inform Cloud} \rangle \end{aligned}$$

From the foregoing, we see that the approach of goal-orientation formulated in Software engineering/Information Systems is **extendable to goal-orientation in IoT with goal reduction along FOs and COs.**

It is to be noted that we do not consider inter-GOT relationships other than AND/OR here. Thus, issues of GOTs supporting each other and conflicting GOTs are not dealt with in this paper. Similarly, we are not dealing with non-functional aspects in the IoT domain here. A treatment of these issues is left for another paper.

### 3 THE GOT MODEL

The foregoing can be represented in a model-driven manner in the GOT model shown in Fig. 1. The model says that a GOT is an aggregate of FO and CO. The former may specify a computation or unprocessed, raw data. From the cardinality, it can be seen that a GOT has exactly one FO and one CO respectively. Further, an FO or a CO can participate in more than one GOT. The structure of the CO in Fig. 1 says that a CO is an aggregate of Sender, Receiver, and Message. The cardinality shows that a CO has exactly one Sender, one Receiver, and one Message. In the reverse direction, a Sender, Receiver, or Message may participate in more than one CO. It may be that the Sender sends the message to itself. For this, consider an IoT system with a single thing, a garden light, that turns ON/OFF after sensing the ambient light. Here, the GOT pair will have CO as:

$$\langle \text{garden light}, \text{garden light}, \text{ambientLight} \rangle.$$

The model proposes three types of GOTs, simple GOTs that cannot be decomposed into simpler ones; complex GOTs that are composed of other simpler

GOTs; and abstract GOTs or those that enter into a generalization/specialization hierarchy.

A complex GOT corresponds to the AND of goal-orientation considered in the previous section. Specialization/generalization allows a high level GOT to have specialized GOTs and corresponds to the OR of goal-orientation. Notice, that the model allows components/specializations that are themselves simple, complex or abstract.

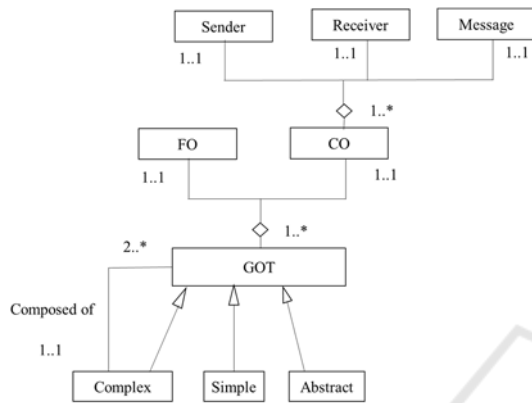


Figure 1: The GOT Model.

Following the classification of a GOT, the **FO/CO may be simple, complex, or abstract** (not shown in the figure to prevent graphical clutter). These terms have the same meaning as for a GOT but are now applicable to functional and communication objectives. Again, the classification of CO implies that Sender and Receiver may be simple, complex, or abstract. Thus, the model makes it explicit that the requirements engineer should consider reduction of all three of these. We shall use this as a basis of our requirements engineering process discussed in the next section.

Interaction among FO and CO is specified by the rules as follows.

**Rule 1:** This rule considers the case when the sender or receiver is complex.

- If the Sender is complex then ONLY the complex Sender sends out the message to the receiver but its component senders DO NOT.
- If the Receiver is complex, then ALL component receivers get the message.

Let us consider an example. Let there be a **simple FO** that computes Average temperature, Avtemp. Let our milk van be a **complex sender** composed of a number of **regions**, each of which calculates its own average temperature and sends it to the milk van. The milk van sends out the message, Avtemp. Let there be a **complex receiver, Operations**, having

components, **Cooling Unit and Recording Unit**. Applying Rule 1 to the GOT, we get the pair

<Calculate\_Av\_temp, (Milk\_van, Operations, Avtemp)>

We see that the Milk-van sends its average temperature to Operations and all components of the latter receive it. The average temperatures computed at the regions are not sent to Operations but only to milk-van. In other words, a valid reduction of the GOT is

<Calculate\_Av\_temp, (Milk\_van, Cooling Unit, Avtemp)>

AND

<Calculate\_Av\_temp, (Milk\_van, Recording Unit, Avtemp)>

**Rule 2:** This rule considers Communication with abstract Sender or Receiver. The rule for communication is as follows

- The sender is specialized: ANY one or more, possibly, all the specialized senders can send a message
- The receiver is specialized: ANY one or more, possibly all the specialized receivers can receive the message.

Now, consider a simple FO, Calculate average temperature. Let us consider the milk van as a region specialized into two regions, front region and back region. Each region has its own cooling unit and we have the receiver, cooling unit specialized into front cooling unit and back cooling unit. Consider the GOT

<Calculate\_Av\_temp, (Region, Cooling Unit, Avtemp)>

Applying the rule for specialization, any or both of our back and front regions can be senders and any or both the colling units can be receivers. Thus, a reduction will be:

<Calculate\_Av\_temp, (Back Region, Back Cooling Unit, Avtemp)>

OR

<Calculate\_Av\_temp, (Front Region, Front Cooling Unit, Avtemp)>

To show communication to more than specialized receiver, let us change the example. Now, there is the milk van without regions but two cooling units as before. Then the message is sent from the milk van to both units. The GOT to be reduced is as follows

<Calculate\_Av\_temp, (Milk Van, Cooling Unit, Avtemp)>

The specialization rule allows the reduction as follows:



<Calculate\_Av\_temp, (Region, Back Cooling Unit, Avtemp)>

AND

<Calculate\_Av\_temp, (Region, Front Cooling Unit, Avtemp)>

**Rule 3:** This rule considers the case of communication when an FO is complex. **Only the data resulting from the complex FO is available for communication.** Any data from the component FOs that needs to be transmitted must be subsumed by it.

This justifies the example at the end of section 2 above. The FO, Clean Data is a component of the complex FO, Compute Avtemp. Cleaned data generated by it is not available for transmission but only average temperature is. If cleaned data is to be communicated then it is the responsibility of Compute Avtemp to do so, in addition to average temperature.

Rule 1 and Rule 2 apply for communication among different kinds of senders and receivers of complex data.

**Rule 4: If an FO is abstract and has specialized FOs, then data available from only one of the latter may participate in a GOT.** That is, a separate GOT is required for each specialized FO and these are organized in the GOT reduction hierarchy. Rule 2 and Rule 3 apply in this case as well.

## 4 THE REQUIREMENTS ENGINEERING PROCESS

The requirements specification is a GOT hierarchy. The requirements engineering process presented in Fig. 2 considers how the hierarchy is built. We refer to this process as the GOT Process.

The initial step, Formulate GOT in the GOT Process is elicitation and formulation of a high-level GOT. As part of this step, the requirements engineer, after first eliciting the FO or the CO, proceeds to complete the GOT. This completion corresponds to a specification of the various elements comprising the GOT. For example, it is possible that the FO is discovered first, as Calculate Average Temperature. In Formulate GOT, the missing CO part is elicited, say Inform Cooling Unit. The sender, receiver, and message of this CO are now determined. This yields the high-level GOT

<Compute Avtemp, (Milk van, Cooling Unit, Temperature value)>

With this, Formulate GOT is complete.

Now, as mentioned in section 3, the three aspects of a GOT that form the basis of GOT reduction are the FO, Sender, and Receiver. Each of these gives rise to its own sub-process. We refer to these as FD for FO Drive, SD for Sender Drive and RD for Receiver Drive respectively. In the FD, reduction of FO is done; in the SD reduction of Sender is done and in the RD, reduction of the Receiver is done. Any of these can be picked up as the next step in the GOT Process as shown in Fig. 2.

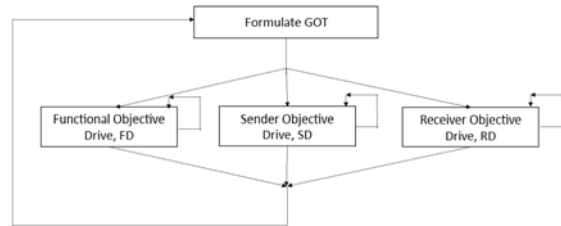


Figure 2: The GOT Process.

In carrying out these drives, the next level of reduced FOs, Senders or Receivers respectively are determined along with their nature, simple, complex or abstract. It is possible to follow these drives to any depth as shown by the self-loops in the figure. Upon exit from the drives, Formulate GOT is carried out and the reduced GOTs are completely defined. The reduction process takes into account the four rules of FO-CO interaction of section 3. It can be seen that GOT reduction is multi-dimensional: FD yields reduction in one dimension, RD in the second, and SD in the third. We will indicate the level in each dimension by the superscript of the drive, for example, 3<sup>RD</sup> and 2<sup>SD</sup> refer to the third level in the RD dimension and second level in the SD dimension. The default dimension is FD.

The GOT Process is applied to each GOT as it is formulated. The process terminates when simple GOTs are reached and no further reduction is possible.

## 5 THE ACCIDENT EXAMPLE

In this section, we apply the GOT requirements engineering process to reporting a car accident. In doing so, we proceed level-wise and complete each level before proceeding to the next.

**The main requirement is to alert help providers in case of a motor car accident.** As stated, this requirement provides both the FO, **Accident Alert**, and the CO, **Inform Help Providers**. This latter is further defined in Formulate

GOT to determine the Sender, Accident Estimator that sends the message, SOS to the receiver, Helper. Accident Alert is predicated on detecting that an accident has occurred. The first row of Table I shows the GOT in two parts in the two columns Functional Objective and Communication Objective respectively. The root level of the GOT hierarchy (level 1 in the table) has been determined and GOT reduction can now be carried out.

Table 1: **FO-Reduced** GOT <Accident Alert, Inform Help Providers>.

Level	Functional Objective	Communication Objective
1	Accident Alert	Accident Estimator, Helper, "SOS"
2	Airbag Release Alert	Air Bag Mechanism, Accident Estimator, "Air Bag Released"
2	Seat Belt Tension Alert	Seat Belt Mechanism, Accident Estimator, Tension Value
2	Impact Alert	Impact Assessor, Accident Estimator, Impact Value

Out of the three drives of Fig 2, let the requirements engineer choose to apply FD to reduce Accident Alert. It is determined that **Accident Alert is a complex FO** built from multiple inputs. The second and subsequent rows in Table I show these FOs. Formulate GOT is applied to each of these FOs and the COs are determined for each.

Table 2: **RD** of Root GOT.

Level	Functional Objective	Communication Objective
1	Accident Alert	Accident Estimator, Helper, "SOS"
2 <sup>RD</sup>	Accident Alert	Accident Estimator, Police, "SOS"
2 <sup>RD</sup>	Accident Alert	Accident Estimator, Health Service, "SOS"
2 <sup>RD</sup>	Accident Alert	Accident Estimator, Relative, "SOS"

Now let us apply SD of Fig 2 to the root goal. Senders of the reduced GOTs have already been determined as the result of applying the FD and the subsequent Formulate GOT step. Now, the requirements engineer determines that there is an OR relationship between the second level GOTs since all are alternative ways of estimating accident occurrence. Consequently, Accident Estimator is an

abstract sender, with the second level senders as its specialization.

The requirements engineer now proceeds to apply RD to the root goal. In fact, the receiver, Helper is complex as shown in Table II since all must be informed about the accident. This change in dimension is shown by the superscript.

Now, the requirements engineer applies the reduction process to each of the second level GOTs in Table I. FOs for *Airbag Release Alert* and *Seat Belt Tension Alert* (second and third rows of Table I) are non-reducible. Attention shifts to **Impact Alert** and the result is shown in Table III. There are four ways in which the car can be impacted making it a complex FO. Notice that these are at the third level of the GOT hierarchy.

SD is applied to the GOT for Impact Alert. Again, the reduced senders have already been determined at the third level GOTs. The question is only about their relationships. Any one or more third level senders can send a message to the second level GOT making **Impact Assessor an abstract sender with third level senders as its specializations**. Application of RD to Impact Alert shows that the receiver Accident Estimator structure has already been considered. Thus, RD does not contribute any further GOTs.

Table 3: Reduction of Impact Alert.

Level	Functional Objective	Communication Objective
2	Impact Alert	Impact Assessor, Accident Estimator, Impact Value
3	Frontal Impact Alert	Front Acceleration Assessor, Impact Assessor, Acceleration Value
3	Rear Impact Alert	Back Acceleration Assessor, Impact Assessor, Acceleration Value
3	Right Impact Alert	Right Acceleration Assessor, Impact Assessor, Acceleration Value
3	Left Impact Alert	Left Acceleration Assessor, Impact Assessor, Acceleration Value

During application of FO to each of the newly discovered GOTs, the requirements engineer determines that these are all simple. Since GOTs of Table III are all irreducible, the GOT Process comes to an end.

## 5.1 The Starting Objective

According to the GOT process, it is possible to deploy any drive in any order. In the accident example, we gave precedence to FD. Instead, the requirements engineer can start with reducing the communication objective first. In the accident example, this means that the starting requirement is, “Inform all helpers that a resident has had an accident” leading to the communication objective (Resident, Helper, “SOS”). During the Formulate GOT, the functional objective, Accident Alert would be identified and the GOT formulated. Thereafter, instead of FD, the requirements engineer could follow the RD and determine the complex structure of Helper. Alternatively, the SD could be followed.

## 6 COMPARISON

In this section we compare our proposal with the goal oriented one of (Reggio, 2018) and use case based approach of (Meacham, 2016). This allows a comparison of the GOT approach with existing two major approaches to requirements engineering.

### 6.1 The Goal-Oriented Genoa Science Festival

The major differences between GOT and (Reggio, 2018) are as follows:

- Reggio proposes an explicit domain modelling stage that yields participants and objects whereas the GOT approach determines these as part of the requirements engineering process.
- Sequence and activity diagrams of (Reggio, 2018) identify respectively, the data needed for realizing operative goals and the sequence of operations to be carried out. In GOT, the issue of how irreducible functional objectives are operationalized is not considered. Rather, it is left to be considered in a subsequent step that deals with conversion of the GOT model to the IoT conceptual model (Prakash 2022). This aspect is the subject of another paper.
- The communication aspect is brought into (Reggio 2018) by associating technological goals like “Communicate with Visitors Using Email” with operative goals like ‘Ask for Booking Confirmation’. How this association is carried out is not clarified but evidently, it is not part of the reduction process. We believe that this reflects the functional objective-

communication objective dichotomy found in the IoT domain. This dichotomy led us to integrate both in the notion of a GOT.

- The well defined-ness of technological goals of (Reggio 2018) is not formulated. In the GOT model, the communication objective is specified when the sender, receiver, and message are all specified.
- Technological goals of (Reggio, 2018) reflect a choice of communication and device technologies likely to deliver operative goals. There is no proposal here to consider alternatives. In contrast, through SD and RD, we do AND/OR reduction of a GOT.

### 6.2 The Use Case View of Fall Detection

Let us now turn our attention to the Fall detection system of (Meacham, 2016):

- Use cases are not used in (Meacham, 2016) to specify data flows in and out of the system but only to define SysML blocks so as to express objectives in textual form.
- Low-level use cases correspond to GOT reduction following FD. However, in (Meacham 2016), we have only a high and low level. This inhibits recursive reduction.
- SysML RDs do not express the message and its sender/receiver.
- The use of free text does not provide any structure for the notion of an objective. In contrast, the GOT is model driven. This provides a systematic GOT Process with clear expressive power and guidance.

## 7 CONCLUSION

An IoT application has its own specific requirements consisting of communication among connected things and processing carried out at things. We have defined a GOT that integrates the notions of functional and communication objectives. The structure of a GOT permits specification of a variety of communication needs like sending messages to multiple receivers or selected senders sending messages to selected receivers.

The issue of non-functional requirements surfaces in two ways in an IoT application. There are traditionally recognized issues like security, reliability etc. Additionally, an IoT has its own non-functional requirements. These are for example, the

cost; battery life, remaining battery charge; convenience of humans wearing devices and so on. We believe that these require detailed investigation in the future.

## REFERENCES

- Anton A., McCracken W., Potts C (1994). Goal Decomposition and Scenario Analysis in Business Process Reengineering. Proc. 6th Conference On Advanced Information Systems Engineering (CAiSE'94), Utrecht, Holland, June 1994
- Costa B., Pires P.F., Delicato F. C. (2017), Specifying Functional Requirements and QoS Parameters for IoT Systems, 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 407-414.
- Costa B., Pires P.F., Delicato F. C. (2016), Modeling IoT Applications with SysML4IoT, 42<sup>nd</sup> Euromicro Conference on Software Engineering and Advanced Applications, 157-164.
- Dardenne, A.; Lamsweerde, A.V.; Ficas, S. (1993). Goal Directed Acquisition. *Science of Computer Programming*. 20:(1-2)
- Eterovic, T., et al., (2015), An Internet of Things visual domain specific modeling language based on UML." *Information, Communication and Automation Technologies (ICAT), 2015 XXV International Conference on. IEEE, 2015 (3)*
- Ferati M., Kurti A., Vodel B., Raufi B., (2016), Augmenting Requirements Gathering for People with Special Needs using IoT: A Position Paper, 9<sup>th</sup> Intl. Workshop on Cooperative and Human Aspects of Software Engineering, ACM, 48 -51
- Fleurey, F., Morin, B., Solberg, A., & Barais, O., (2011), MDE to manage communications with and between resource-constrained systems. In *International Conference on Model Driven Engineering Languages and Systems*, 349-363
- Föcker F., Neubauer A., Metzger A., Gröner G., Pohl K. (2015), Real-time Cargo Volume Recognition using Internet-connected 3D Scanners. ENASE, 323-330.
- Kotronis C., Nikolaidou M., Dimitrakopoulos G., Anagnostopoulos D., Amira A., Bensaali F. (2018), 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.
- Lamsweerde van A. (2000), Requirements Engineering in the Year 00: A Research Perspective, Keynote Paper for ICSE'2000 - 22nd International Conference on Software Engineering, Limerick, ACM Press
- Meacham S., Phalp K., (2016) Requirements Engineering Methods for an Internet of Things applications: Fall Detection for Ambient Assisted Living, BCS SQM/Inspire Conference
- Mezghani E., Exposito E., Drira K., (2017), A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare, in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, 224-234
- Mikusz M., Schafer T., Taraba T., Jud C. (2017), Transforming the Connected Car into a Business Model Innovation, 19<sup>th</sup> IEEE Conference on Business Informatics, 247 – 256.
- Prakash D., Prakash N., (2021), Towards a Life Cycle for IoT Applications Development, TechRxiv. Preprint.<https://doi.org/10.36227/techrxiv.14906277.v1>
- Prakash N., Prakash D.(2022), Concepts for Conceptual Modelling of an IoT Application. ENASE, 494-501
- Prehofer C., Chiarabini L. (2015), From Internet of Things Mashups to Model-Based Development. *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference, COMPSAC '15*, vol. 3, 499-504
- Reggio G., (2018), A UML based Proposal for IoT Requirements Specification, 10<sup>th</sup> International Workshop on Modelling in Software Engineering, 9 - 16.
- Stankovic J.A.,(2014), Research Directions for the Internet of Things, *IEEE Internet of Things Journal*, 1, 1, 3 -9.
- Takai T., Shintani K., Andoh H., and Washizaki H. (2019), Case Study Applying GQM+Strategies with SysML for IoT Application System Development," 8<sup>th</sup> International Congress on Advanced Applied Informatics (IIAI-AAI), 914-919.