Ambi Graph: Modeling Ambient Intelligent System

K.Chandrasekaran,I.R. Ramya, Syama .R
National Institute of Technology Karnataka (NITK), Surathkal.
kch@nitk.ac.in, ir.ramya@gmail.com, syamar@gmail.com

Abstract - **In computing, ambient intelligence (AmI)[2,3] refers to electronic environments that are sensitive and responsive to the presence of people. In an ambient intelligent world, devices work in concert to support people in carrying out their everyday life activities, tasks and rituals in easy, natural way using information and intelligence that is hidden in the network connecting these devices. The ambient intelligence paradigm builds upon ubiquitous computing [4] and human-centric computer interaction design. In this paper, we introduce a notation, called Ambi Graphs, for specifying the work flow in any ambient intelligence system. An ambi graph is elegant when it is designed to adapt to the wide range of users that work on it.**

Keywords-Ambient intelligence, sensor, surrounding, target, sensing, interpretation, action, element diagram, computational diagram, ambi graph.

I. INTRODUCTION

Having a well-defined and expressive notation is important to the process of any kind of development. Standard notations make it possible for describing a computing scenario or developing device architecture and then unambiguously communicate these decisions to others. A notation is a vehicle for capturing the reasoning about the behavior and architecture of a system. It is wholly a very well defined approach which does not involve the use of state diagrams or object diagrams. Characteristics of ambient intelligence environments differ from traditional ones due to various factors, like merging with the surroundings, acting in response to a set of sensed objects, and so on. A generalized approach to represent every ambient intelligence system has to be self-explanatory to every user. We find that the notations reported in literature, such as State diagrams and object diagrams are not sufficient to describe an ambient intelligence system to its fullest. A State diagram [1] is used to show the state space of a system, the events that cause a transition from one state to another and the actions that result from a state change. Here we have extended the State diagrams formally as Computational diagrams, that have the same features of State diagrams and some more features to model the characteristics to give a generalized approach. Object diagrams [1] model the dynamic behavior of an object in its lifetime. Here we have used a notation called Element diagram to specify real-time systems. The notation uses the object diagram's representations but with a much sophisticated approach. In this paper, we introduce a notation, called Ambi Graphs, for specifying the work flow in any ambient intelligence system. Though elements can be thought of as building blocks to model ambi graph, element diagram formalism does not provide mechanisms to model the typical features of ambi graph. Some important ambient intelligence device features are specifying element location, properties, types of sensors used, adaptability, etc. Currently, element diagrams do not have clean mechanisms to specify the above features. In this work, we extend the specification of an element with a describer called tasks. Now each element can independently describe its traits and tasks completely. Using ambi graph, users can design ambient intelligent devices by understanding how exactly they want the device to function.

What is a notation and why it is used

A standard notation, that unambiguously expresses different aspects of a system, is important to the process of software development. The expressiveness of a standard notation helps analysts and developers to describe a computing scenario or to formulate software architecture and then to communicate these decisions unambiguously to other team members. An expressive notation eliminates much of tedium of checking the consistencies and correctness of the architectural decisions of the developer by using automated tools.

What is ambient intelligence?

Ambient Intelligence is a vision of a world where we are surrounded by a huge amount of intelligent and small devices, which are seamlessly embedded in the user's environment. It is a vision on the future of consumer electronics, telecommunications and computing that was originally developed in the late 1990s for the time frame 2010–2020. A key to the process of realizing an ambient intelligent system is defined using the following stages:

- Sensing
- Interpretation
- Action

II. AMBI GRAPH

The following three models are used to define an ambient intelligence system. Element graph- one that describes all the elements present in the model; Computational graph- describes various computational techniques used and the sequence in which the system proceeds with various inputs; Ambi graph- diagram that shows the actual flow of data element graph and the computational graph to describe the system as a whole.

Element Graph

In an ambient intelligence system (device), there has to be a classification of different elements which constitute not only the device but also the surroundings. Here, surroundings play a vital role because most of the existing ways of representation has description of only how the device works. By describing surroundings, we ensure that the changes that are sensed are clearly monitored. In the element graph we describe the various elements which constitute any ambient intelligence device. The major elements are as follows:

- Environment
- Sensor
- Target

To name each element, we use a semi-circular block. This is placed over two other rectangular blocks. One of which consists of the traits of the element and the other, the tasks. Traits and tasks are used to describe the two features of the elements- its characteristics and its mode of operation related to it.

We connect the elements with arrows to show the work flow. This is a trivial work-flow model. Here we enhance on the traits and tasks rather than the links from/to various elements. Since this approach is dynamic, we express the traits and tasks within each element as shown in Fig.1. Modification of data is possible at any stage of development. It has to be reflected in all the dependent models. The following diagram is a generalized view of all the elements mentioned above.

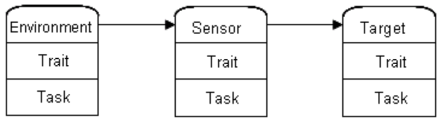


Figure1. Element Graph

Let us consider the very famous home intelligence systems. When somebody enters the hallway, the lights automatically get turned on. When they sit on the couch, the coffee maker starts making coffee, etc.

Here our surrounding is the room. The possible traits of this room could be the presence or not of anybody around. Based on this, the task is the turning on or off of the lights. If we consider the count of people as the trait, the task could be how many people to prepare coffee for.

Similarly, for sensor and target also we can have traits and tasks. These are based on the computation involved with respect to the entity that is sensed. These shall be discussed later so as to provide a clear idea because it follows the computations involved in each step of sensing.

Computational Graph

Any intelligent system is known so for it's decision making ability. We have a particular form of representing our computations with details such as what events and corresponding actions are performed, the current state of the device, the conditions which act like a split-road. So we represent all these details in a very sophisticated form in Fig.2.

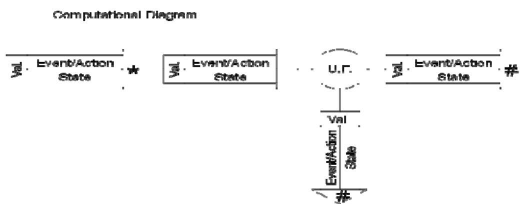


Figure 2:Computational diagram

Each state is represented in the form of block arrows. The upper half describes the event and action. The lower half specifies the state name. Both, event and action are separated by a '/'. Event is the occurrence of a particular process. Action is the process of doing something that can lead us to the next state. Sometimes an ambiguity occurs when we have to decide which the next state has to be based on a specific condition. In such cases we use utility functions that maps each possible state of a system or object to a real scalar value on a common scale. Utility function is specified in the circle. The value returned by the utility function is written in a box at the beginning of the next state. The start state and final state are marked with an asterisk (*) and a hash (#) respectively.

We use a policy based approach that separates the policy from the implementation of a system and therefore permit the policy modified without changing the systems underlying implementation. In the model being described we will have two policies a low level policy and an efficiency level policy. A low level policy basically describes the computational model at a very abstract level. At the efficiency level, the computational model described is very specific. This approach helps maintain the hierarchy. As ambient intelligence is a technique used widely in the armed forces there should be check on what is available to whom. This kind of a policy based approach ensures that critical information does not fall in the wrong hands.

Getting back to the home intelligence system, if somebody sitting on the couch is the event and making coffee is the action. If we had to choice between making coffee and juice, we need a condition. This condition could be sensing the temperature. In on a clod day, the coffee maker can serve us and on a hot day, nice cold juice could be heaven!

Ambi Graph

The ambi graph is the core of any ambient intelligent system. It shows the actual step by step process that takes place. By making use of the element chart as well as the computational chart the ambi graph gives a very precise model of ambient intelligence. It is very important to have such a model as the elements and the computations alone cannot lead to the understanding of a system. The ambi graph acts as a container which uses both these models to give a complete description. The model may roughly be divided into two parts:

- Elements
- Computations

Figure 3. Ambi Graph

In the above diagram environment, sensor and target are the element graphs specified by rectangular boxes. The element diagram is specified inside the rectangle. The computational model is specified inside the rhombus. The arrows show the flow of data between the element graphs and the computational graph. Data received from elements or as a result of certain computation is written on the arrows. This allows the programmer to have a clear picture of the flow of data along with all the processes and transitions taking place in the system. The programmer gets a better view of the whole scenario and the things he has to take care of at any given point of execution of the system.

The ambi graph has two forms a private form and a public form. In the private form all the details are described and the total workflow is represented. Whereas in the public form only the outer picture or abstract picture is represented.

In the case of our intelligent home, the turning on or off of lights depends on certain data such as presence or absence of people in the room. Similarly the amount of coffee that has to be made depends on the count of people. This data is represented on the arrow headed lines in the above mentioned model. This data is processed by the computational model which again leads to a new piece of data leading to another element.

III. CASE STUDY

WORKING OF A TREADMILL

As the famous saying by Albert Einstein goes, “Example isn't another way to teach, it is the ONLY way to teach." Treadmill is a device used to exercise. It consists of a continuously moving belt. The treadmill is programmable regarding simulated course, distance, initial speed, and user weight, and records approximate caloric burnt. Sensors strategically positioned below the belt upper surface regulate belt speed to keep a user toward the center of the treadmill.

The treadmill works as follows. The upper surface of the revolving belt upon which a user performs is equipped, on the underside, with three sensors. The sensor nearest the front of the belt is an acceleration sensor. The acceleration sensor causes the speed of the rotating belt to increase. Conversely, the deceleration sensor slows the belt speed. The target speed sensor maintains belt speed. Through these mechanisms, the present invention automatically adjusts to the momentary speed of the person, through any course. No impact on the sensors stops the treadmill.

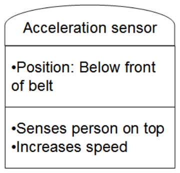
Element diagram

In a treadmill, we have identified the following elements with respect to our modeling concepts. They are as follows:

- Acceleration sensor
- Deceleration sensor
- Target speed sensor
- User
- Belt
- Drive motor

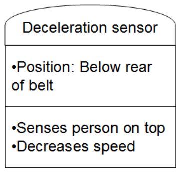
Acceleration sensor

It is placed below the belt. The acceleration sensor causes the speed of the rotating belt to increase when the belt is compressed against it by a user's foot. An increase in speed of the belt thereby carries a user back toward the midpoint of the belt upper surface, the most desirable position for a user to be in.



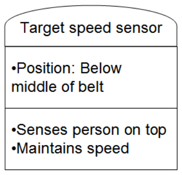
Deceleration Sensor

The deceleration sensor is disposed below the belt proximal to the rear of the drive assembly of the treadmill. When the belt is caused to be compressed against the rear sensor by a user's foot strike, the belt communicates that information to the CPU which in turn slows the belt speed, providing for the user to return to the midpoint of the rotating belt.



Target Speed Sensor

The target speed sensor is disposed midpoint below the belt upper surface. The target speed sensor maintains belt speed, as the user is ideally positioned when in the center. Through these mechanisms, the present invention automatically adjusts to the momentary speed of the runner or walker, through any course. No impact on the sensors stops the treadmill.

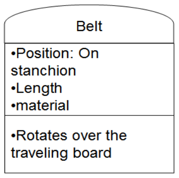


Drive Motor

The driver/controller is itself controlled through programmable circuitry having a display and keyboard with user input options. The controller receives signals from a speed sensing device attached to the AC induction motor to maintain the rotational speed of the AC motor within pre-selected limits. The AC induction motor is attached to one or more flywheels and directly engages, through a drive roller, a walking belt. The matched combination of an AC motor driver/controller with appropriately sized flywheels allows utilization of a variable speed AC induction motor for the direct drive of the treadmill belt.

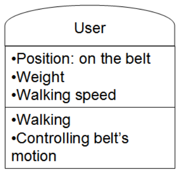
Belt

The treadmill has an endless belt entrained around a drive roller and an idler roller. The belt has a first or upper reach and a second or lower reach extending between the rollers. The belt has an outer surface and an inner surface. The endless belt encircles a support deck so that a user positioned on the outer surface of the upper reach is supported by the top side of the deck. The inner surface of the upper reach is in a sliding relationship with the deck.



User

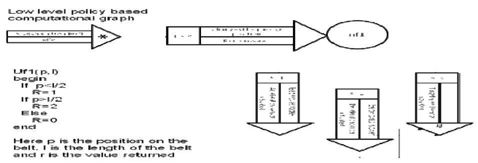
Each user has different properties. For example, the weight of each user differs and so does the walking speed. Certain treadmills give an option to have different profiles for each user. The amount of calories burnt is also kept a record of. Depending on the user's position on the belt, the speed of rotation of the belt varies.



Computational Graph

In the computational graph of the treadmill we describe the various computations and decision makings taking place in the treadmill.

First let us see the low level policy based computational graph which gives an overview of the working without going into the intricate details. In Fig.4 we describe where each sensor comes into picture without giving an actual description of the internal working and the use of motors.

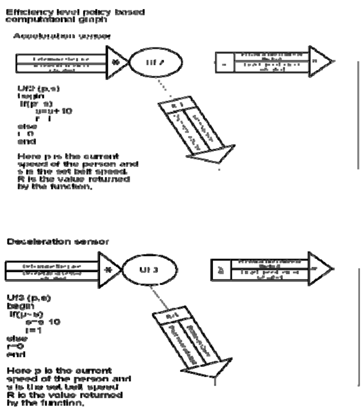


```
Uf1( p, l)
begin
  If  p<l/2
     R=1
  If p>l/2
     R=2
  Else
     R=0
end

Here p is the position on the
belt. l is the length of the belt
and r is the value returned
```

Figure 4. Low level computational graph

In Fig.5, the efficiency diagrams, we look at each sensor separately and show the actual computations and interactions with the driver motor and the belt. The acceleration sensor increases the speed of the driver motor if the pace at which the user is moving is greater than the pace of the belt. This is done so as to make the user move towards the centre of the belt where his target speed is calculated. Similarly the deceleration motor slows down the driver motor if the user is walking too slowly.



```
Uf2 (p,s)
begin
 If(p  s)
     u=u+10
     r  1
else
r  0
end

Here p is the current
speed of the person and
s is the set belt speed.
R is the value returned
by the function.
```

```
Uf3 (p,s)
begin
 If(p<s)
     s=s 10
     r=1
else
r=0
end

Here p is the current
speed of the person and
s is the set belt speed
R is the value returned
by the function.
```

Figure 3. Efficiency level computational graph

Ambi Graph

Treadmills are used by different category of people. Depending on the class and understanding required by them, we propose two types of ambi charts. For the high level a private one and for a comparatively lower level a public one. Below is the complete description of the two types.

Private Ambi Graph

We make use of a private ambi chart in order to make every process visible to the user. This solely depends on the kind of user. This level is preferred for the advanced level users.

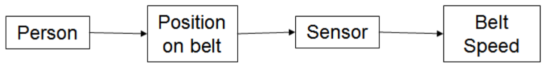


Figure 3. Private ambi graph

Public Ambi Graph

The public level is used for the reference of a normal user. At this level only the basic knowledge of how a treadmill works is required.

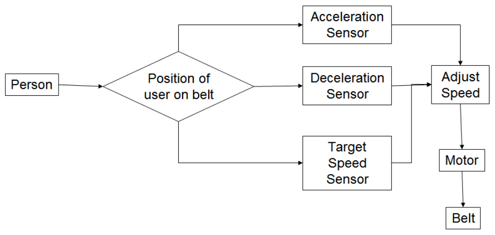


Figure 3. Public ambi graph

CONCLUSION

We have tried to give a notation for ambient intelligence systems above. Such a system is very useful in the analysis and planning stages. It provides a very clear understanding of the system describing all the functionalities and properties of it. It is also a form of documentation for the system. Using such a notation makes it easier for the programmer to design the system.

REFERENCES

[1] Satyajit Acharya Hrushikesha Mohanty and R.K. Shyamasundar “Mobicharts: A Notation to Specify Mobile Computing Applications”
[2] Orit Zuckerman “Interactive Portraiture: Designing Intimate Interactive Experiences”
[3] Paolo Remagnino Gian Luca Foresti “Ambient Intelligence: A New Multidisciplinary Paradigm”
[4] Anand Raghunathan, Srivaths Ravi, Sunil Hattangady, and Jean-Jacques Quisquater “Securing Mobile Appliances: New Challenges for the System Designer