Designing for End-User Development in the Internet of Things

Barbara Rita Barricelli and Stefano Valtolina(✉)

Department of Computer Science, Università degli Studi di Milano, Milano, Italy
{barricelli,valtolin}@di.unimi.it

Abstract. With the widespread of Internet of Things' devices, sensors, and applications the quantity of collected data grows enormously and the need of extracting, merging, analyzing, visualizing, and sharing it paves the way for new research challenges. This ongoing revolution of how personal devices are used and how they are becoming more and more wearable has important influences on the most well established definitions of end user and end-user development. The paper presents an analysis of the most diffused applications that allow end users to aggregate quantified-self data, originated by several sensors and devices, and to use it in personalized ways. From the outcomes of the analysis, we present a classification model for Internet of Things and new EUD paradigm and language that extends the ones existing in the current state of the art Internet of Things.

Keywords: Internet of Things · End-User Development · Quantified self · Lifelogging · Pervasive computing · Mobile devices · Unwitting developers · End users

1 Introduction

The Internet of Things (IoT) concept was coined in 1999/2000 by Kevin Ashton and his team at MIT's Auto-ID Center [1] and rapidly spread around the world thanks to the evolution of sensor technology and its use that is becoming more and more mobile and pervasive [2]. To connect uniquely identified everyday objects in a network allows to send and receive data and at the same time to influence the behavior of the objects in two ways: automatic, on the basis of the collected data, and semi-automatic/manual, according to users' needs and/or preferences. Today, IoT is successfully adopted in several application domains and it is estimated that in 2015 the number of objects connected will be around 12 billion, while in 2020 it will be 50 billion [3][4].

Recent studies [5][6] show that the coming of IoT changed the way people use the Internet, and mobile and sensor-based devices. This tendency is more relevant in domains that present pervasive characteristics where the integration of data could help in improving quality of life and in offering an even richer and satisfying experience of use of everyday objects. This type of integration is what characterizes the so-called *lifelogging*: keeping track of the collected data through all the everyday or occasional

P. Díaz et al. (Eds.): IS-EUD 2015, LNCS 9083, pp. 9–24, 2015.
DOI: 10.1007/978-3-319-18425-8_2

activities that may influence people's quality of life. Lifelogging, initially conceived in the 70s as a 24/7 broadcasting of self-videos, has become today a wide spreading phenomenon, called quantified-self movement, that allows people to keep track of their habits, health conditions, physiological data, and behavior, and to monitor conditions and quality of the environments in which they work and live. Today, a continuously increasing number of lifelogging devices are on the market and become more and more affordable to the masses.

In our research, we mainly study applications of lifelogging in three domains: health, wellness, and domotics. In the health domain, people can collect data gathered through several devices for monitoring, among all, blood pressure, heart beat rate, glucose level, and coagulation factor. Lifelogging in the wellness domain allows to keep track for example of weight, sport/fitness activity, calories intake, and sleep quality. As to domotics, IoT helps in having better awareness about energy consumption, use of entertainment or work appliances, and even care of gardens/plants. Some of the most advanced IoT devices offer solutions based on artificial intelligence and expert systems for avoiding to prompt users too often and risking to bother them with too many questions. The idea to make objects and environments able to take decisions on behalf of the users aims at not disturbing and overwhelming people in their everyday lives. Although these automatic suggestions avoid to bother users by helping them in managing objects more easily, we believe that the user control over connected objects is a crucial element for IoT success. In fact, newly created Web, mobile, wearable, and pervasive applications are today designed in a more user-centered manner and particular attention is made in taking care of the user experience.

More than 20 years ago, Cypher [7] defined the end user as a "user of an application program", someone who is not a computer programmer and who "uses a computer as part of daily life or daily work, but is not interested in computers per se". In the IoT era, this concept evolves because now machines are becoming part of the social tissue and their use is common in almost every cultural context: with the growing diffusion of mobile devices, like smartphones and tablets, pervasive computing is spreading [8]. IoT allows the end users to manage physical devices, interactive systems, and quantified-self data by deciding how to create new usage scenarios and this empowers them more than ever, making them evolve, as explained later in the paper, to become end-user developers [9].

In Section 2, we describe how digital devices have become not only tools to satisfy the need of getting jobs done but also the key for taking care of social relationships (real or virtual) and to manage several aspects of personal life (e.g. financial, wellness, entertainment). Under this perspective, we describe IoT as an ecosystem of objects and services that aim at supporting the end users in extracting, merging, analyzing, visualizing, and sharing data enabling them to unwittingly transforming the data into information, information into knowledge, and knowledge into wisdom. This scenario leads towards an innovative point of view on technology and mobility, focusing diversity and agency as central aspects of a socially responsible approach to mobile computing [10]. According to this consideration, we then discuss the most consolidated definitions of end user and End-User Development (EUD) with respect to the IoT domain. Even though the EUD definitions given in scientific literature remain valid, we claim how the perspective has deeply changed. EUD in IoT is now focused on how users interact with an ecosystem of elements and how they are able to affect the way data is collected and aggregated. According to this new perspective, in Section 3 we present the current state of the art of EUD in IoT and in particular we present applications that enable the users to arrange data coming from IoT devices/sensors and to aggregate it via Social Media, Mobile and Web apps. Finally, in Section 4 we present the definition of a new EUD paradigm and language in IoT domain. Specifically, we propose a sensor-based rule language able to support the end user in aggregating and combining data originated by several sensors/devices and in creating personalized use of the quantified self-data. This language aims at enabling end user for unwittingly developing personalized IoT environments according to specific temporal, spatial, and fuzzy conditions that may affect the elements in the IoT environment.

2 End-User Development in the Internet of Things Era

The "old computing" as claimed by Shneiderman [11] is focused on what computers can do for the user, while the "new computing" regards people activity and what people can do by using computers. Users of digital devices and interactive systems are increasingly evolving from passive consumers of data and computer tools into active producers of information and software [12][13]. The potentials offered by network and connectibility of the objects does not only enrich the person's personal sphere but also offers the possibility of sharing data with other people who can be family members, friends, colleagues, or others. Data sharing contributes to the creation of a large quantity of data especially in the long term, calling for the integration of recommendation, intelligent, and distributed systems in order to help in their aggregation and exploitation. In this scenario, the end users finds themselves at the center of a complex ecosystem that they need to manage in efficient, effective, satisfactory, and aware manner. EUD represents the ideal approach for empowering the end users and make them becoming unwitting developers in their own IoT environment [14][15][16].

2.1 The IoT Ecosystem

In designing for the IoT, the attention is not focused on the development of a unique interactive system but of an ecosystem of elements (hardware and software) that exchange data through the Internet and act and react in a semi-automatic or automatic way according to events, and/or users' preferences, rules, or decisions. At the center of this ecosystem stands the end user, the one who generates (or contributes to) the data, manages the IoT elements in the ecosystem, and unwittingly develops in the IoT environment defining the interactions among the elements and the elements' behavior. The elements of the ecosystem (depicted in Figure 1) can be categorized into five groups:

Sensors. The IoT sensors are typically built-in components in electronic devices aimed at collecting data of various nature. Examples of sensors are those present in devices for weather stations, activity tracking armbands, or Wi-Fi body scales. IoT devices can be portable – meant to follow the end user everywhere (e.g. activity trackers) – or unmovable – designed for being placed in a specific place and not moved around (e.g. weather stations). Sensors and their devices can autonomously send the data they collect or wait for the end user to collect the data when they need to. They typically come with applications for enable the end user to access them (both settings and data) but can also present an embedded stand-alone interface.

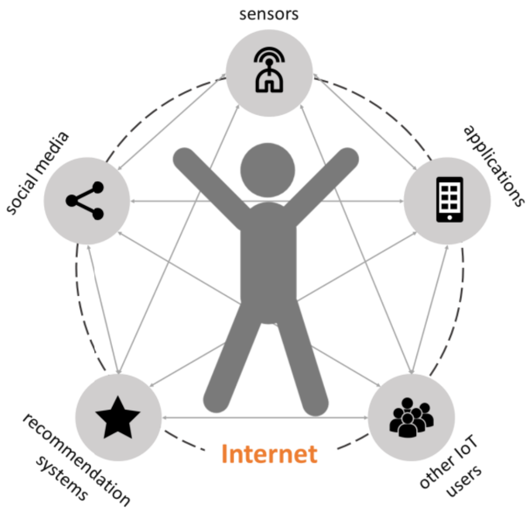

Fig. 1. The IoT ecosystems. The end user plays the central role deciding what elements to connect, how to interact with them, and how they should interact with each other. (Icons made by Freepik from www.flaticon.com are licensed by CC BY 3.0).

Applications. Through them, the end user is able to access the IoT devices. They typically are of two types: bundled with specific devices or compatible with several devices. Applications are usually designed to be mobile-compliant giving the end user the chance of interacting with the devices on remote setting.

Social Media. Those applications built on Web 2.0 principles and technology that allow the end user to share content on the Internet. The relationship with IoT is that most of its devices and applications are equipped with social features that allow the end user to share the collected data with virtual communities (friends, family, or colleagues in real life or people they just know on the Web).

Recommendation Systems (RS). In the context of IoT, recommendation services aim at suggesting proper aggregation, integration and distribution strategies of data coming from sensors or other informative resources and at tailoring them according to the context of use, the users profile, and goals. However, the use of automatic suggestions might be not appreciated by the end user that may feel frustrated in using the RS features, whenever the recommendations appear to be inappropriate. To deal with this problem, some RSs offer solutions able at exploiting end user's social relationships for improving the service quality.

Other IoT users. Those people who belong to the virtual communities mentioned before. They typically share with the end user some particular interests, life choices, or other aspects. Is the end user who chooses the people to be connected with on the basis of personal searches or suggestions made by the applications (thanks to RSs).

The quantity of different types of IoT devices that are today on the market is continuously growing and their variety leads to a higher and higher level of complexity in the IoT ecosystem. To support the integration of new IoT devices and related applications into the ecosystem and understand how to better empower the end user in becoming unwitting developer of their own IoT environment, we propose a 3-dimensional model of classification (Figure 2) that is based on three peculiar aspects in IoT: space, time, and social dimension.

Space. This dimension goes from “settled” to “mobile”. Elements in the “settled” area of the 3D model are typically constrained to a fixed position (e.g., home, office) and are not supposed to be used on the move. Examples of such category are devices for ambient surveillance, weather stations, energy consumption monitoring, water leak sensors. On the other hand, “mobile” elements are designed to be used in different places while accompanying the user during their movements. An example of this category of elements are wearable devices that are used to track activity, calories burning, and physiological data (e.g., fitness armbands, smartwatches).

Time. Along this axis, an element can be categorized as asynchronous or synchronous. Asynchronous elements are typically those that collect data only when the user decides to, while synchronous elements collect and analyze data on the fly when they are generated without the need of having users directly involved. Especially earlier IoT devices were not equipped with Bluetooth/Wi-Fi connectivity and a mechanical action by the users was required to connect the device with a smartphone, a tablet or a desktop PC in order to collect the data generated by the device’s use. Today, most of the IoT devices are designed to be standalone and directly connected to the Internet so that the users’ intervention can be very limited.

Social Dimension. An element may be designed for individual use, if it is supposed to be used by a user only, or for collective use, if the element’s data are meant to be accessed by many users and not only by the element’s owner. Choosing between sharing and keeping private the data collected via IoT devices can be driven by personal motivations or by default characteristics of the devices themselves; sometimes in fact, IoT devices are meant to be used individually only and the sharing of data can be achieved only using third-party applications.

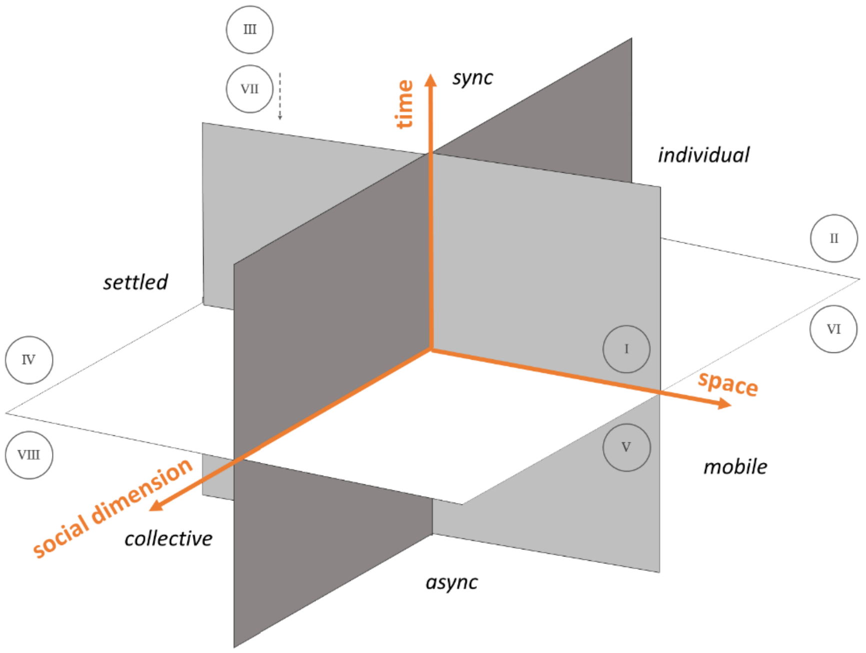

Fig. 2. The 3D model for classifying IoT devices according to space, time and social dimension

Table 1. The octants resulting from the 3D space depicted in Figure 2

Octant	Social dimension	Space	Time	Signs
I	Collective	Mobile	Synchronous	+ + +
II	Individual	Mobile	Synchronous	- + +
III	Individual	Settled	Synchronous	- - +
IV	Collective	Settled	Synchronous	+ - +
V	Collective	Mobile	Asynchronous	+ + -
VI	Individual	Mobile	Asynchronous	- + -
VII	Individual	Settled	Asynchronous	- - -
VIII	Collective	Settled	Asynchronous	+ - -

Such representation allows to identify the position of the elements in terms of octants of the 3D space. Table 1 presents all the octants in the model. The most social, connected, flexible, and mobile elements are those located in octant 1 (+ + +), while the elements that present less degree of flexibility and personalization and do not follow the users in their social and real life are those octant VII (- - -). Given the flexibility of some IoT devices, it is important to keep into account that the position of the elements in the 3D space model of classification may change in time because their state can dynamically change according to users' behavior and preferences. It can be used both as a tool for analyze an existing IoT ecosystem and to explore and better understand its potentials, or as a classification to inform IoT ecosystems design that applies EUD techniques. In particular, the classification can be of help for identifying the peculiarities of the context of use of the IoT ecosystem that is under design. According to the values of the three dimensions – space, time, and social – the designer is able to decide what EUD features are to be provided to the user, what EUD activities, and to what extent EUD can be applied without overwhelming the user. The values of space, time, and social dimensions may also influence the interaction style design that has to be adopted in a specific context of use with specific devices (e.g. mobile or desktop, touchscreen or not).

Beyond the practical uses that one can do of this model, in this paper it is used to highlight that the high complexity of IoT ecosystems may inevitably cause a shift in the traditional and more or less consolidated definitions of *end user* and *EUD*, as explained in what follows.

2.2 EUD and IoT: Evolution of Definitions and Principles

The definition of end user has experienced deeply changes in the last decade. However, there are some seminal works in the consolidated EUD scientific literature that still hold and are those that see the end user as someone interested in using digital devices just for the sake of it and not with the idea of becoming expert in the technology itself (e.g. [13], [17]). Also the definition of EUD given in [9] still sounds valid to describe the phenomenon: "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact". In this scenario, end users are increasingly evolving from passive consumers of data and computer tools into active producers of information and software [18][12]. From an organizational point of view, end users are not necessarily experts in computer science, but in the domains they work in. In [19], Åsand and Mørch consider end users those persons who are skilled with computers, while Nardi and Miller [20] gave a classification of end user according to their computing skill level. At some point, it is possible to recognize some similarities between the two classifications: Åsand and Mørch describe regular users as those workers who are not developers and not interested in tailoring but want to use computers to perform they daily work; Nardi and Miller define the non-programmer users as workers who could also some have programming skills. On the contrary, Ye and Fischer [21] did not focus on users' classifications but on how the distinction between users and developers is going to disappear with the time's passing. There are however, some other definitions that do not reflect anymore the current scenario of IoT and this can be linked to a shift in the definition of time, space, and social dimension that are the dimensions of the 3D model of classification that we presented in Section 2.1. Nardi's definition given in [22] states that the end user is "the person who does not want to turn a task into a programming problem, who would rather follow a lengthy but well-known set of procedures to get the job done". If we consider that today the majority of digital devices are general purpose, the term job is too vague and too work-related to be used when speaking of modern technology. With the large diffusion of portable and mobile digital, pervasive systems are becoming the most diffused computing paradigm in which infrastructure and services are seamlessly available anywhere, anytime, and in any format [8]. Dourish, Anderson, & Nafus proposed in [10] an innovative point of view on technology and mobility, focusing diversity and agency as central aspects of a socially responsible approach to mobile computing. This work also connects current research in HCI, ubiquitous computing and human and social geography suggesting new perspective for design that should help in reflecting the current idea of *space*. This broadening of the space dimension in the use of digital devices leads to a revision of all those definitions of end users that consider the context of use as fundamental. Brancheau and Brown [23] confined the end users to a space that is somewhere "outside the information system department". Confining the end users at the end of design and development processes and putting distance, as suggested by Cypher [7], between them, designers, and developers is an approach that does not reflect the current society and its real expectations. Another problem with this definition is that the notion of time in today's life and the way in which we manage it have deeply changed: digital devices continuously become faster and faster allowing their users to obtain feedbacks and results very quickly. Moreover, with the computational performance, also our speed in performing actions and take decisions has increased and our expectations in terms of time saving have become very high. This has led us to reconsider the concept of *time* and to change the way in which we deal with its flow [24]. Every process becomes more and more faster and any time spent waiting for a response from a machine is seen as an unbearable waist. So, forcing the user to perform a "lengthy" set of procedures does not appear to be the right design choice. Moreover, when dealing with sensors and temporal data, there is the need to make a distinction between valid time and transaction time. The first refers to the instant in which an event actually occurs, while the second is linked to the instant in which the event has been registered in the system. Another aspect that changed in the last decade is the concept we have of the *social dimension* in which we live: the digital devices have become not only tools to satisfy the need of getting jobs done but also the key for taking care of social relationships (real or virtual).

3 EUD Activities in IoT

As widely reported in literature, EUD can be enabled by offering the end users tools that allow them to develop without having specific programming skills and knowledge about programming languages. In this Section, we first resume the most consolidated literature on EUD activities, and then we critically review the current practice of EUD by presenting and discussing the most used applications for IoT that implements EUD activities.

3.1 Literature Review

EUD covers a large area of interests, i.e. customization of applications by parameters setting, control of a complex device like a home-based heating system, script of interactive Web sites [25]. EUD allows users to configure, adapt, and evolve their software by themselves [26] and such tailoring activities, together with personalization, extension, and customization are defined in literature in different ways, sometimes referring the same concepts and sometimes referring different ones [27]. Trigg et al. [28] define a system as adaptable if it "enables user-customizable behavior". They also state that a system can be adaptable in four different ways: i) flexible systems provide generic objects and behaviors that can be interpreted and used in different ways from different users to carry out different tasks; ii) parameterized systems offer many alternative behaviors among whom the users can choose; iii) integratable systems can be interfaced to and integrated with other facilities being part of the environment or connected to remote facilities; iv) tailorable systems allow users to modify the system by building accelerators, specializing behavior, or adding functionalities. In [27], Mørch defines the tailoring activity as a way to bridge the gap between the objects that compose the interface (simple widgets such as menu items, icons, buttons or composite widgets such as menus, dialog boxes) and the underlying implementation code that defines the functionality (written in a general-purpose programming language). Furthermore, he presents three levels of end-user tailoring: by customization, by integration, and by extension. With customization users can modify the appearance of presentation objects (the ones that compose the interface) or can edit their attribute values by choosing among a set of predefined configurations. Integration allows users to add existing functionalities to an existing application. Extension permits to add new functionalities to an existing application. A further definition of tailoring is given by [29]: if the modifications that are being made on a system are on the subject matter of the tool then there is a use activity, otherwise if the modifications are made on the tool itself that can be called activity tailoring. A further classification of EUD activities has been introduced in [30]. The authors list five characteristics in the functional design of tailorable technologies by adapting what Baldwin and Clark presented in [31]: 1) splitting by reducing a single module to smaller components; 2) substituting by replacing components or parts of them; 3) augmenting by adding new modules; 4) excluding by deleting modules; 5) porting by adding a component made for another technology. In [32] another classification of users' activities is presented: the different activities are classified in two distinct classes that includes respectively those activities that allow users to choose among different behaviors by setting some parameters and those activities that imply some programming for software artifact creation or modification. Furthermore, the authors provide examples of activities belonging to the two different classes, Class 1 and Class 2. Class 1 groups together activities that support the user in setting parameters in order to choose among various behaviors available in the application. Two examples of activities that belong to this class are parameterization and annotation. Class 2 is constituted by those activities that allow the user to create or modify a software artifact, by programming in any programming paradigm. To meet the users' need of not becoming developers, programming by demonstration, programming by examples, visual programming, and macro generation are used. Examples of activities that belong to this class are modeling from the data, programming by demonstration, formula languages, incremental programming. All these classifications still apply to EUD for IoT but an important observation has to be made: in IoT, the target of EUD activities is not (only) the interface and the behavior of an interactive system, but is the whole IoT ecosystem with its elements. Therefore, we need to distinguish between those activities that can be made at three different levels: hardware, software, and data. EUD activities on hardware are those made on the devices via their bundled applications. They typically are configuration, personalization, and customization by setting parameters and choosing among existing behaviors. The activities on software targets the applications that allow to control more than one sensor/device (even of different brands) and include tailoring by integration of existing and/or new functionalities, macros, visual programming, and programming by examples. The EUD activities that can be made on data can be resumed in aggregation, filtering, and porting. In what follows, we will use the classifications presented so far for discussing the state of the art of applications that can manage data originated by more than one sensor/device and shared on Social Media, and that enable the end user to unwittingly develop their own IoT environment.

3.2 The Current State of the Art

We analyzed the most diffused applications for IoT that exploit EUD principles and we identified two main types that differ in terms of activities and interaction style. A first type of applications allow users to define sets of desired behaviors in response to specific events. This is made mainly through rules definition-wizards that rely on the states of sensors/devices. Rules can be typically chosen among existing presents or can tweaked through customization. These EUD activities can be clearly seen as belonging to Class 1 and put in place a task automation layer across all sensors/devices in the IoT environment. Such strategy is adopted by those applications that use automated rules-based engines like Atooma (http://www.atooma.com/) and IFTTT (https://ifttt.com) – by using the programming statement *IF this DO that,* and by Wewiredweb (https://wewiredweb.com/) with the statement *WHEN trigger THEN action.* With a more advanced use of these applications, the end user can exploit EUD activities that belong to Class 2 and make use of RSs (as part of the IoT ecosystem). These activities are supported by the RSs that by reading end user's pattern of use for a device can suggest compelling examples of statements that the end user can adapt to their needs. A second type of applications stems from the outstanding work done with Yahoo's Pipes (https://pipes.yahoo.com/pipes/) and can be classified to Class 2 as they typically use formula languages and/or visual programming. Applications like Bipio (https://bip.io/) and DERI pipes (http://pipes.deri.org/) offer engine and graphical environment for data transformation and mashup. They are based on the idea of providing a visual pipeline generator for letting the end user creating aggregation, filtering, and porting of data originated by sources. An advanced use of such visual paradigm is offered by WebHooks (https://developer.github.com/webhooks/) that allows the end users to even write their personal API for enabling connections with new sources of data.

Both presented typologies of EUD strategies, adoptable in the context of the IoT applications, offer a solution able to gather information from across the net and trigger specific actions when certain things happen. The first type of applications offers a very simple and easy to learn solution based on the definition of ad hoc rules that can notify the end users when something happens – e.g. when their favorite sites are updated, when they check-in in some places or their friends do, or warn them when specific weather conditions are going to take place. However, the adoption of the IF-THIS-DO-THAT/WHEN-TRIGGER-THEN-ACTION patterns are not enough to deal with more sophisticated rules based on time, space, and fuzzy conditions. On the other hand, the second type of applications offers a too complex solution for supporting the end user in expressing their preferences. Pretending that the end users are able to deal with APIs of several sensors/devices put at risk the success of the EUD approach. Another problem with the current state of the art regards the fact that in the most diffused applications the social dimension is commonly taken care of, while time and space dimensions are almost never considered. To face these problems, in the next Section we propose an extension of the IF-THIS-THEN-THAT paradigm by presenting a sensor-based rule language able to support the end user in defining rules in a more articulated way but keeping the complexity at an acceptable and accessible level. This idea is to keep the simplicity of the IF-THIS-THEN-THAT paradigm pairing it with the use of formula languages. Moreover, time and space dimension will be exploited and fuzzy conditions are adopted for expressing more loose rules in the statements.

4 A New EUD Paradigm and Language for IoT

In the most common programming languages, a control structure is a block of instructions that on the basis of specific variables and parameters chooses a direction (flow control) to follow. The flow control determines how a computer will respond when certain given conditions are in place and specific parameters are set. In the same way, in IoT domain, the end user needs to state that if a condition (e.g. the weather station says that it is going to rain in the next 12 hours) and an action (e.g. tweet this news on the end user's Twitter personal account using hashtags #weather #rain). As described before, this solution is adopted in many applications for supporting end users in creating rules for their IoT environment. The IF-THIS-THEN-THAT paradigm seems work well when end users need to be warned or notified on a specific event, but uses a very simple language that has a quite low expressive power. We propose to extend this paradigm by giving end users the possibility of setting triggers that do not depend just on one event but also on other conditions. Such conditions rely on a language with higher expressive power that draws from database management rule languages. Moreover, the paradigm allows end users to design triggers that depend also on time and space and not only on social media content, like most of the applications in the current state of the art. The introduction of time dimension allows end users to set triggers that can be fired at some specific time, delayed in case of certain conditions are verified, and may be repeated until some event happens. The space dimension gives end users the chance of linking triggers to the place/area where they currently are, where they will possibly be in the future, where they are moving into, or where some events are taking place. In literature [33][34] there is a nearly unanimous agreement on an extension of "classical" trigger languages by including time dimension.

The proposal in this field can be summarized as follows: (1) Rules should be triggered by the occurrence of time events, (2) Enabling periodically repeated triggering of the same reaction, where the period is specified by an expression returning a time duration. (3) Delaying reaction execution to some later point in time relative to the triggering event of a rule. In [35] the authors propose a set of functionalities to be implemented with triggers written in SQL:1999 standard that cover three types of temporal categories – absolute, periodic, and relative event specifications – and allow to base delay or periodic repetition on valid time or transaction time events, respectively. According to this proposal of functionalities, we can provide users with a new set of rules composition-strategies able to go beyond the simple use of an IF-THIS-THEN-THAT statement. Up to now, rules can be triggered by call events only, and reactions are always executed one time. We identified four types of rule:

1. Space Events: rules that need to be triggered if the data stream refer to a specific geographical place/area. An example: IN *"homeplace"* IF *"my sleep-controller detects that I did not sleep at least 7 hours in the last 3 nights"* THEN *"the alarm clock on my smartphone should ring at 11 PM for suggesting me to go to bed earlier"*.
2. Time Events: Rules triggered on certain absolute time events are the most common feature of time-triggers. An example: AT *"summer time"* IF *"my sleep-controller detects that I did not sleep well the night before"* THEN *"my activity tracker device should suggest me to take a walk before going to bed"*.
3. Delayed Reaction Execution: Reaction execution can be delayed by combining a call event with a temporal offset. This offset is a time-valued attribute of the related environment, thus generating "relative events". For instance, to check three months after my last blood test if I need another test, a possible rule could be: AT *"The date of my last blood test + 3 months"* IF *"the person scale says that I lost more than 10 kilograms"* THEN *"my smart watch should show a message suggesting to book a medical exam"*.
4. Repeated Reaction Execution: Repeating execution of a particular action regularly after a fixed period of time has passed. In this case the keyword EVERY could be combined with an expression of type PERIOD, e.g. EVERY MONTH, or EVERY 3 HOURS, or with even more sophisticated specifications, such as EVERY MONDAY, EVERY 2nd MONDAY IN A MONTH, or EVERYDAY EXCEPT SATURDAY.

The EUD paradigm we propose in this paper aims supporting the end user in composing such space/time-based rules for extending the well-established but not powerful IF-THIS-THEN-THAT paradigm. Our Sensor-based Rule Language follows syntax, semantics, and grammar of a Policy Rule Language proposed in [36], and is based on the ECA (Event, Condition, Action) paradigm [34]. Our language allows to specify rules stating policies for triggering actions (one or a set). The general format of a rule is the following (square brackets denote optional components):

```
RuleName: "MY RULE"
ON Sensor[s]
  [WHENEVER "Condition"]
Action: "Some Actions"
  [VALIDITY: Validity_Place-Interval]
```

A rule consists of several components. The `RuleName` component represents the rule identifier. Users can retrieve rules by means of such identifier for visualizing, sharing, dropping, or modifying them. `Sensor[s]` represents the sensor or set of sensors upon which data the rule is triggered. Each sensor exposes a set of parameters which can be used for expressing the conditions. `Condition` is an optional conditional expression. `Action` is an expression that states what happens when the condition is verified. `Validity_Place-Interval` is a special spatial and/or temporal condition also expressed by means of the condition language we developed, representing the space and time period during which the rule is enabled. For example, if the interval [EVERYDAY EXCEPT SATURDAY] is specified we know that a rule is enabled every day of the week but not on Saturday. But if `Validity_Place-Interval` is not specified, we know that the rule is always enabled. By means of `Validity_Place-Interval` it is possible to state that certain rules are not always enabled; rather, they are enabled only if an event happens in a specific place or during specific temporal intervals. Such a feature is not provided by conventional apps for IoT.

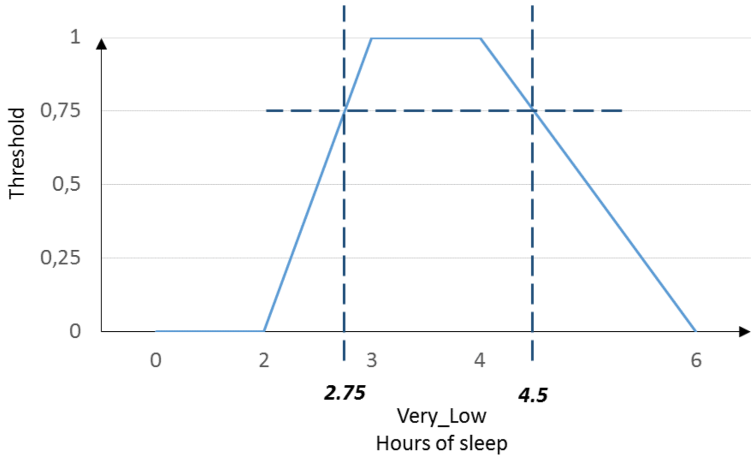

Fig. 3. Example to illustrate hours of sleep distribution

Another extension of the IF-THIS-THEN-THAT paradigm is to provide end users with a more flexible way for expressing the condition statement by incorporating fuzziness into the condition on which the events need to be triggered. This extension focuses on the linguistic values of the fuzzy condition [37], in which the fuzziness concerning the linguistic concepts are interpreted in an application context. A linguistic variable is represented by a quintuple of <v,T,X,g,m> where v is name of the linguistic variable, T is set of linguistic terms applicable to variable v, X is the universal set of values, g is the grammar for generating the linguistic term, m is the semantic rule that assigns to each term t∊T, a fuzzy set on X. To illustrate our approach, we us as example an IoT environment that has a sleep monitor among its elements. Let represent a linguistic variable with a graphical distribution based on four parameters as depicted in Figure 3. Very_Low for hours of sleep is represented using trapezoidal function as Very_Low (2 hours, 3 hours, 4 hours, and 6 hours). Current IoT applications use simple statements such as "Hours of sleep <= 3 hours" to indicate when the value is very low. Using our language, users can use the statement "Quantity = $Very_Low": a set of values are related to "$Very_Low" in this comparison, rather than one single value. Fulfillment threshold is allowed to specify the condition with a degree value in the range of [0, 1]. For example, in Figure 3, we used 0.75 as the threshold to indicate that the value of hours of sleep is very low with the degree of 0.75. As a result, a value in the range [2.75, 4.5] indicates that "the number of hours of sleep is very high with a threshold of 0.75". By using our Rule Language the user can express a fuzzy condition is this way:

```
RuleName: "Quality of sleep Monitor"
ON sleep-controller AND Thermometer
  WHENEVER "the number of hours of sleep is $Very_Low"
AND "the temperature is not $Very_High"
Action: "The activity tracking device suggests me to take
a walk before going to sleep"
  VALIDITY: IN "Milan" and AT "August"
```

5 Discussion and Conclusion

The main contribution of this paper regards the analysis of the current state of EUD in the light of the development of IoT research and practice. The comprehensive overview that we provided, helps in underlying the nature of the challenges that arise today. The analysis of existing IoT ecosystem, as well as the under-design ones, according to elements and dimensions (time, space, social) may be of great help in assessing potentials and issues that may arise. From a study of the most diffused applications for IoT that provide the user with EUD tools, we identified and discuss some open problems and propose in this paper a new EUD paradigm and language to solve them. The language presented in the previous section is currently under implementation in an IoT application that is aimed at dealing with an ecosystem in the wellness domain. It consists in the design and development of an interactive visual system aimed at implementing the paradigm and language proposed and at testing its validity.

References

1. Ashton, K.: That ‘Internet of Things’ Thing. RFID Journal, June (2009). http://www.rfidjournal.com/articles/view?4986 (accessed on January 9th, 2015)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Comput. Netw. **54**(15), 2787–2805 (2010)
3. Connections Counter: The Internet of Everything in Motion. http://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342 (accessed on January 9th, 2015)
4. Evans, D.: The Internet of Things. How the Next Evolution of the Internet is Changing Everything. Cisco Internet Business Solutions Group – White Paper (2011). http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (accessed on January 9th, 2015)
5. Meder, J.: Human Empowerment in a Semantic Web of Things: Concept of a semantic platform for connected devices. Master Thesis at Uppsala University, Department of Information Technology (2014)
6. Munjin, D.: User Empowerment in the Internet of Things. Ph.D dissertation at University of Geneve, Department of Economy and Management (2013)
7. Cypher, A.: Watch What I Do: Programming by Demonstration. The MIT Press (1993)
8. Barricelli, B.R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Padula, M., Scala, P.L.: Designing pervasive and multimodal interactive systems: an approach built on the field. In: Ubiquitous and Pervasive Computing: Concepts, Methodologies, Tools, and Applications, pp. 212–233. IGI Global (2010)
9. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End-User Development, pp. 1–8. Springer (2006)
10. Dourish, P., Anderson, K., Nafus, D.: Cultural mobilities: diversity and agency in urban computing. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4663, pp. 100–113. Springer, Heidelberg (2007)
11. Shneiderman, B.: Leonardo’s Laptop: Human Needs and the New Computing Technologies. MIT Press (2002)
12. Fischer, G.: Beyond ‘Couch Potatoes’: From Consumers to Designers and Active Contributors (2002). http://firstmonday.org/issues/issue7_12/fischer/ (accessed on January 9th, 2015)
13. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE TSMCA **37**(6), 1029–1046 (2007)
14. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: End users as unwitting software developers. In: Proc. of WEUSE 2008, pp. 6–10. ACM (2008)
15. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Advanced Visual Systems Supporting Unwitting EUD. In: Proc. of AVI 2008, pp. 313–316. ACM (2008)
16. Barricelli, B.R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Valtolina, S., Fresta. G.: BANCO: a Web Architecture Supporting Unwitting End-User Development. IxD&A, 5-6, pp. 23–30 (2009)
17. Petre, M., Blackwell, A.F.: Children as Unwitting End-User Programmers. Proc. of VL/HCC **2007**, 239–242 (2007)
18. Costabile, M.F., Fogli, D., Lanzilotti, R., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Advancing end-user development through meta-design. In: End User Computing Challenges Technologies: Emerging Tools and Applications, pp. 143–167. Information Science Reference (2007)

19. Åsand, H., Mørch, A.: Super Users and Local Developers: the Organization of End User Development in accounting company. JOEUC **18**(4), 1–21 (2006)
20. Nardi, B.A., Miller, J.R.: An ethnographic study of distributed problem solving in spreadsheet development. In: Proc. of CSCW 1990, pp. 197–208). ACM Press (1990)
21. Ye, Y., Fischer, G.: Designing for Participation in Socio-Technical Software Systems. In: Proc. of UAHCI, pp. 312–321. Springer (2007)
22. Nardi, B.: A Small Matter of Programming. MIT Press (1993)
23. Brancheau, J.C., Brown, C.V.: The Management of end-user computing: status and direction. ACM Computing Surveys **25**(5), 437–482 (1993)
24. Lee, H., Liebenau, J.: Time and the Internet at the Turn of the Millennium. Time & Society **9**, 43–56 (2000)
25. Sutcliffe, A., Mehandjiev, N.: Introduction of Special Issue on End User Development. CACM **47**(9), 31–32 (2004)
26. Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V.: Introduction. In: Proc. of IS-EUD 2009, pp. V–VI. Springer (2009)
27. Mørch, A.: Three levels of end-user tailoring: customization, integration, and extension. In: Computers and Design in Context, pp. 51–76. MIT Press (1997)
28. Trigg, R.H., Moran, T.P., Halasz, F.G.: Adaptibility and Tailorability in NoteCards. In: Proc. of INTERACT 1987, pp. 723–728. Elsevier Science Publishers (1987)
29. Henderson, A., Kyng, M.: There's no place like home: continuing design in use. In: Design at Work: Cooperative Design of Computer Systems, pp. 219–240. Lawrence Erlbaum Associates (1991)
30. Germonprez, M., Hovorka, D., Collopy, F.: A Theory of Tailorable Technology Design. Journal of the Association for Information Systems **8**(6), 315–367 (2007)
31. Baldwin, C.Y., Clark, K.B.: Design Rules: The Power of Modularity. MIT Press (2000)
32. Costabile, M. F., Fogli, D., Mussio, P., Piccinno, A.: End-user development: the software shaping workshop approach. In: End-User Development, pp. 183–205. Springer (2006)
33. Ceri, S., Cochrane, R., Widom, J.: Practical applications of triggers and constraints: success and lingering issues. In: Proc. of VLDB 2000, pp. 254–262 (2000)
34. Widom, J., Ceri, S.: Active Database Systems. Morgan Kaufmann Publisher (1996)
35. Behrend, A., Dorau, C., Manthey, R., Grundspenkis, J., Morzy, T., Vossen, G. (eds.): SQL Triggers Reacting on Time Events: An Extension Proposal Advances in Databases and Information Systems. Springer (2009)
36. Bertino, E., Cochinwala, M., Mesiti, M.: UCS-Router: a policy engine for enforcing message routing rules in a universal communication system. In: Proc. of Mobile Data Management 2002, pp. 8–16 (2002)
37. Jin, Y., Bhavsar, T.: Incorporating fuzziness into timer-triggers for temporal event handling. In: Proc. of IRI 2008, pp. 325–329 (2008)