End-User Development for Personalizing Applications, Things, and Robots

Fabio Paternò & Carmen Santoro
CNR-ISTI, HIIS Laboratory
Pisa, Italy
{fabio.paterno, carmen.santoro}@isti.cnr.it

ABSTRACT
The pervasiveness of ICT technologies has led to a growing need to empower people to obtain applications that meet their specific requirements. End-User Development (EUD) is a growing research field aiming to provide people without programming experience with concepts, methods and tools to allow them to create or modify their applications. Recent mainstream technological trends related to the Internet of Things (IoT) and the availability of robots have further stimulated interest in this approach. In the paper, we discuss the historical evolution of EUD, then we analyse the main current challenges with respect to recent technological trends (IoT and social robots) through the use of some conceptual dimensions, and conclude with a discussion of a possible research agenda for the field.

Author Keywords
End User Development; Internet of Things; Robots; Metaphors; Programming styles.

INTRODUCTION

The goal of End-User Development (EUD) is to allow people without programming experience to create or modify their applications (Lieberman et al., 1986). In EUD the focus moves from easy-to-use to easy-to-develop. Interest in EUD arose soon after the introduction of personal computing in order to empower people without particular programming experience (such as teachers, scientists, health care workers, salesmen, and administrative assistants) to be directly involved in the creation of their applications. Often such people work on tasks that rapidly vary, and thus their software needs are diverse, complex, and frequently changing. Professional software developers cannot directly meet all of these needs due to their limited domain knowledge and because their development processes are too slow.

Historically, first End-User Programming (EUP) (Nardi, 1993) was proposed but this concept is more limited than EUD, since EUD methods, techniques, and tools span the entire software development lifecycle, including modifying and extending software, not just the creation phase. Burnett and Scaffidi (2013) consider EUP as the subset of EUD that is the most mature, describing EUP as a set of techniques that empower end users to write programs by adopting special-purpose programming languages, such as those included in spreadsheets or web authoring tools. EUP also includes techniques such as programming by demonstration, visual programming, and high-level scripting languages.

EUD aims at empowering end users to develop and adapt systems at a level of complexity that is adequate to their practices, background and skills (Lieberman et al., 2006). Therefore, it focuses on system flexibility and modifiability, and it encompasses domain-specific environments for software creation. The goal of EUD is thus to make users able to participate in their own software artefacts design and development, not only at design time, but also during use, which also distinguishes EUD from Participatory Design, which foresees users' participation at design time only.

Some authors (Ko et al., 2011) proposed the End-User Software Engineering (EUSE) concept as well, with the aim of finding ways to incorporate general software engineering activities into users' existing workflow, with an emphasis on the quality of the software end users create, modify, or extend. EUSE takes a different perspective compared to EUP and EUD because it focuses on systematic and disciplined activities carried out throughout the system lifecycle to guarantee the quality of the code created by end users. In particular, EUSE proposes techniques derived from traditional software engineering, which are aimed at fostering reliability, efficiency, reuse, debugging support, maintainability, and version control (Burnett, 2009).

The EUD approach has shown to benefit from the increasing intertwining of the design and use phases that characterise modern applications, because it is easier for users to think about how to change the application after having actually used it. Thus, it obviates some limitations of user-centred design in which users are mainly involved in the identification of requirements for the professional developers (Fisher and Giaccardi, 2006) so that the designers extract information from the users (through interviews, focus groups, questionnaires), observe them at work and retrieve their feedback. However, the inverse process does not happen, i.e. the users are not directly involved in the world of software design and development, and their contribution in this phase is not contemplated. Participatory design foresees a more active user contribution, even in the design phase. EUD tends to further strengthen this active involvement by allowing, to some extent, the users to even autonomously carry out design of at least some parts of their applications. Such an activity can be performed at various times, not only at the initial design phase, but also after actual use. However, to make this possible there is a need for meta-design methods and tools (Fisher and Giaccardi, 2006) that provide open environments to enable design changes of interactive applications continuously intertwined with their actual use without requiring people to substantially change the nature of their work or their priorities.

In terms of application domains, a recent systematic literature review of EUD, EUP, and EUSE over the last twenty years (Barricelli et al., 2019) has pointed out that the application domains that have mostly been considered are: business and data management, web applications and mashups, and smart objects and smart environments. Other application domains that have received attention are games and entertainment, education and teaching, healthcare and wellness, mobile applications, interaction design, and robotics.

In this paper, after a description of the historical evolution of the field, we focus on the current generation addressing EUD for IoT and/or robot applications. Thus, we introduce its characterising elements, and describe a design space that facilitates the discussion of the various approaches proposed. Then, we compare a representative set of approaches through the identified design dimensions, and lastly we provide indications for promising research topics and draw some conclusions.

HISTORICAL EVOLUTION

Historically, it is possible to identify various generations of EUD approaches. In the evolution of this research area, the start of a new generation has been characterised by the advent of some mainstream ICT technology, which raised the need for new application types or allowed users to exploit new platforms to obtain their applications.

The first approaches relevant to EUD were put forward in the late 80s, soon after the advent of graphical desktop systems, which made it possible to support the first emerging end-user development needs. Over time several technologies had an effect on the possibilities for EUD, including language technologies and social systems. Amongst them, the rapid increase in the use of the Web, with its open interfaces (as opposed to offline desktop applications) to support computational work, made possible the design of a

number of interactive tools to support EUD. The third generation corresponded to the success of touch-based mobile devices, characterised by sufficient interaction and computation resources to directly support development activities. The last generation aims to empower users in order to exploit the continuously increasing availability of smart things and robots, two types of technologies that share the use of various types of sensors and actuators.

Figure 1 provides a summary view of the evolution of the field. For the sake of clarity, we did not include in the diagram all the contributions cited in the paper. In the late 80s and early 90s the first contributions, which mainly addressed how to improve end user programming, (e.g. Myers and Buxton 1987, Nardi, 1993, Cypher 1993) were put forward. Then, on the European side there was the Network of Excellence on EUD[1] , which was useful to create a community with an open view of how the user can be empowered in the development cycle. In parallel, the NSF started a project (EUSES[2]) on end-user software engineering that stimulated a more systematic view on how to approach the various software engineering phases from the user side. Such research efforts stimulated an agreed definition of EUD (Lieberman et al., 2006) and various research initiatives, for example, the manifesto for meta-design (Fisher et al., 2004). Regarding the exploitation of the Web technologies in this perspective, a book (Cypher et al., 2010), which was a follow up of a CHI workshop, provides a good overview of various approaches in this area. The first attempts to exploit mobile technologies in this area were put forward around 2008 (e.g. Carmien and Fisher 2008, AppInventor 2010). An overview of initial approaches in the area of EUD for IoT applications appeared in the dedicated TOCHI special issue (Markopulos et al., 2017).

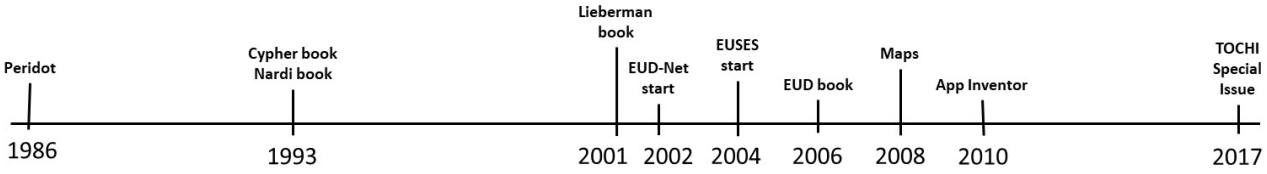

Figure 1 – Summary overview of the field evolution.

In the first generation, corresponding to the wide availability of graphical desktop systems, the contributions mainly focused on two types of approaches: those based on spreadsheets and those based on visual languages. A long standing example of domain expert development of interactive applications is the spreadsheet application: laymen as well as professionals can develop or adapt computational models useful for accounts, planning, etc. with a minimal overhead of learning programming concepts and conventions. To reach such a broad audience the EUD should be almost transparent to users, e.g., spreadsheet users may not even be conscious of the fact they are programming when processing data. At least, EUD should present a very low threshold to get started, while letting users progressing far in the value and even complexity of the software they create. In the case of spreadsheets, the interest started with VisiCalc, then continued with Lotus 1-2-3 and Excel, and the focus has been on facilitating the definition of expressions to apply to multiple cells, and then testing their results, and the analysis of the relationships amongst the values of the cells involved. One goal can be to simplify the tracking of successful and failing inputs incrementally, providing feedback about software quality as the user edits the spreadsheet program.

In the case of visual languages one often used approach is based on icons associated with high-level functionalities, which are connected through arrows to indicate the data that can flow across them. For

[1] http://hiis.isti.cnr.it/projects/eud-net.htm

[2] http://eusesconsortium.org/

example, LabVIEW is an environment for creating circuit simulations and other analysis programs. Each box represents a computational component, while lines indicate the flow of data. Usually, in this type of approach there is some support to check whether the output from one functionality is actually compatible with the input of another one. One further visual approach that soon raised interest is that based on the jigsaw metaphor. The idea is to visually represent the jigsaw puzzle pieces so that each element corresponds to a function or a programming element, and their shapes provide hints about how many connections they can manage both for receiving input and producing output. Scratch (Resnick et al., 2009) was one of the first environments to adopt this type of representation. In visual languages one common design aspect has been how abstractions are used to hide the implementation details (Paternò, 2013). In some cases, the main visual elements have been associated with high-level functionalities developed by programmers, so that end users need only to compose them without knowing how they were implemented. In other cases, the visual elements correspond to low-level programming constructs, and the end users should specify how the desired interactive program should behave at a more detailed level. In this phase another approach to EUD that emerged is programming-by-example, in which the user demonstrates an example of what the program should do, from which the programming environment infers a more general specification supporting the desired behaviour. Peridot (Myers and Buxton, 1986) applied this approach for creating interactive components.

Over time the Web has become the most common user interface because it can be accessed through most devices and provides an open interface (the Document Object Model[3]), which can be exploited to interactively manipulate Web applications, even by people other than the original developers. The programming by example approach has been implemented in Web environments through different mechanisms. Nichols and Lau (2008) describe a system that allows users to create a mobile version of a Web site through a combination of navigating through the desired portion of the site and explicitly selecting content. Macias and Paternò (2008) take a similar approach, in which users directly modify the Web page source code. These modifications are used as a specification of preferences, which are then generalized and applied to other pages on the same site through the support of model-based user interface description languages. CoScripter (Leshed et al., 2008) is a system that allows user to record, share, and automate tasks to perform in the Web, and provides a repository where the scripts created are shared. In CoScripter scripts are recorded as natural language commands that can be modified by the user without having to understand a programming language. In this area one approach often considered is the mashup approach characterised by the possibility of creating new applications by interactively composing components from existing applications. NaturalMash (Aghaee and Pautasso, 2014) is a Web-based environment that allows non-programmers to exploit existing Web resources by combining their input/output. NaturalMash users start defining a mashup by picking ingredients from a toolbar that includes services/contents available through Web APIs. Then, users specify how to bind components together through a natural language subset. However, the mashup components associated to textual expressions are predefined and require pre-processing by expert programmers. PEUDOM (Matera et al., 2013) is another Web-based platform that allows end users to compose components associated with registered Web services into a mashup. Components are defined by professional developers, and can subsequently be connected by means of drag-and-drop actions and by selecting the binding properties from some dropdown menus. Ghiani et al. (2016) put forward a graphical environment in which users create new mashups by directly selecting interaction elements, content and functionalities from existing Web applications without requiring the intervention of expert developers. Then, users just need to exploit

[3] https://www.w3.org/TR/WD-DOM/introduction.html

a copy-paste metaphor to indicate how to compose the selected interactive content and functionalities in the new mashup.

A different perspective in exploiting Web technologies that can be relevant for EUD has been introduced by Webstrates (Klokmose and others, 2015). It is an environment for creating shareable dynamic media that blurs the distinction between documents and applications, showing how software can become reprogrammable and extensible in a collaborative fashion. Webstrates augment web technology with real-time sharing. They turn web pages into substrates, i.e. software artefacts that embody content, computation and interaction, blurring the distinction between documents and applications, as substrates can evolve over time and shift roles, acting as what are traditionally considered documents in one context and applications in another, or a mix of the two.

The advent in the mass market of touch-based mobile devices has enabled new opportunities for EUD. On the one hand, the mobile device can be the platform through which users can create or customize their applications, and on the other hand applications have to be able to consider that the surrounding context of use is no longer fixed, and so they have to adapt to changes that can dynamically occur in the environment. Carmien and Fisher (2008) describe a framework for customizing mobile applications to help people with cognitive disabilities. A graphic editor, intended to be used by the caretakers, facilitates the management of the task-support scripts for helping the disabled. Ghiani et al. (2009) have developed an environment that allows customization of mobile solutions for museum guides, and it also allows the generation of application versions for stationary systems with large screens. Puzzle (Danado and Paternò, 2014) supports editing on a touch-based smartphone by using the jigsaw metaphor to convey the concepts of connecting high-level functionalities, and a solution, inspired by the work by (Cuccurullo et al., 2011) using the colours to indicate the associated data types, thus providing intuitive cues to help the users to correctly connect the outputs and inputs of the jigsaw pieces. App Inventor (2010) addresses the application development at a more detailed granularity, asking the end-user developers to use jigsaw pieces representing low-level programming constructs and specify what should be done when low-level events occur.

The Internet of Things is the network of objects of our daily life (such as lights, refrigerators, car components, medical devices, dog collars, etc.) that can send or receive information with various devices on the network. These objects include sensors and actuators of various kinds and can interact with each other, with human beings and with the environment to exchange data, reacting to real-world events, triggering actions and activating services. They are increasingly used in any sector: home, retail, industry, agriculture, ... We use our applications more and more in dynamic contexts in terms of services, devices, objects and people where many events can occur. Since 2004 there are more connected devices than people in the world and the number of connected objects is steadily increasing. According to a recent report of the World Economic forum, it is one of the largest enablers for responsible digital transformation and it will add $14 trillion of economic value to the global economy by 2030 (WEF, 2018). These technological trends provide great opportunities, new possibilities, but also risks and new problems. Indeed, our interactions with such objects can be monitored by unauthorized parties, their inappropriate use can generate unwanted effects, there can be intelligent services based on them that eventually generate effects that do not match the real needs of end users. Thus, one fundamental research question has become how to provide tools that allow users to control and customize the way they use the available connected objects.

Atzori et al. (2010) survey the Internet of Things area mainly from a technical perspective (e.g., by discussing the pros and cons of enabling technologies such as RFID and TCP), but also mention the benefits of combining sensors and actuators with personalization techniques: managing home appliances based on user preferences and dynamic contextual factors can improve comfort, safety and energy efficiency. Some work to address such issues in the EUD perspective has started to be put forward. One

of the first proposals was iCAP (Dey et al., 2006), which introduced the possibility to create if-then rules and to support personalization of dynamic access to home appliances. Ghiani et al. (2017) aim to provide an environment able to support intuitive editing of a broader set of rules in terms of possible trigger and action types, and with additional possibilities, such as rule reuse and sharing. IFTTT[4] is a popular environment that allows users to easily connect existing applications in such a way that if something happens in one, then some effect can be generated (for example a functionality is activated) in a kind of Trigger-Action Programming (TAP). Unfortunately, IFTTT has limited expressivity since it only allows a single trigger per rule. Ur et al. (2014) wondered about the balance between expressivity and usability of TAP, and have conducted a study to find out how the average end user can manage flexible trigger-action rules in the home domain. The results show that average users can successfully manipulate multiple triggers and actions to formulate rules, but studies are still needed to assess the attitude of the users to understand the differences between rules that are similar and that slightly differ in the trigger (e.g., a simple check or a state change). Huang and Cakmak (2015) found that the distinction between relevant concepts can be a source of problems, since users can have difficulties interpreting the difference between events and conditions or between the possible types of actions (for example extended actions, which reverts back to the original state after some time automatically and sustained actions, which do not revert to the original state automatically). Misunderstandings can cause undesired behaviours (e.g. unlocking doors at the wrong time or cause unintended energy waste). It is important that EUD tools take into account the requirements emerging from this study, for example allowing users to differentiate between event triggers (that hold only when a contextual change occurs) and condition triggers (that hold whenever a condition is true).

Application composition is an approach to create applications by using software components (i.e. web services or other resources associated with objects/devices) as building blocks. In general, there are two main approaches to application composition (Davidyuk et al., 2015). In the automated composition, user intervention is minimal since the system automatically configures and provides most of the functionalities. In the interactive composition, the user has a high degree of control and can freely compose the final application. Recently, due to the importance of configuring the behaviour of IoT applications, rule-based approaches are receiving increasing interest, since end users can easily reason about contextual events and the corresponding behaviour of their applications (Ur et al., 2014). However, even if rule specification could seem simpler than specifying block of code, such approaches can become difficult for non-programmer users when complex rules have to be expressed. The correct formulation of logical expressions implies knowledge of some key concepts (e.g. Boolean operators, priority of operators) that may not always be intuitive for them. Some approaches do not even support events composition at all (as it happens with IFTTT). Therefore, further effort in enabling end users to specify rules combining multiple triggers and actions should be pursued because this would provide them with the possibility to indicate more flexible behaviours (Desolda et al., 2017; Ghiani et al., 2017). In (Metaxas and Markopoulos, 2017) an established theory of mental models is used to guide the design of interfaces for EUD so that people can easily comprehend and manipulate logical expressions. According to such theory, people find it easier to conceptualize logical statements as a disjunction of conjunctions (an OR of ANDs), as opposed to other logically equivalent forms. Thus, the authors propose a paradigm to facilitate the specification of complex logical expressions that however is still far from providing a general solution. More generally, one important aspect to consider is that one barrier to the uptake of EUD approaches for IoT is the lack of compelling motivations to adopt them, since users sometimes do not see any reason to invest time to learn how to use these tools and overcome the risks of failure (Blackwell, 2002).

[4] https://ifttt.com/discover

Another interesting aspect (yet underexplored in the EUD area) is how people can test and possibly assess whether the modified/created behaviour of the application actually results in the expected one. This need is especially relevant in IoT domains, where incorrect behaviour of applications or actuators can eventually have safety-critical consequences (e.g., in the elderly assistance domain and in the home domain). If we consider rule-based approaches, a way to reduce the likelihood of errors is to allow users to simulate the conditions and the events that can trigger a rule and the effects that they will bring about. However, most EUD environments do not include debugging aids (Coutaz and Crowley, 2016) since non-professional end users find debugging especially difficult. Manca et al. (2019) present a possible solution to help end users understand whether the specified trigger-action rules behave as desired and without conflicts. It provides answers to common why/why not questions concerning rules execution in specific context states that can be interactively defined, as well as conflict analysis functionalities. In any case, another important area for further investigation is devising debugging mechanisms that are adequate for end users. One further issue is that overall the studies on the usability of trigger-action programming tools have usually been carried out in laboratories, far from realistic contexts of use where users can immediately perceive the results of the execution of their rules. One exception is AppGate (Coutaz and Crowley, 2016), which was tested by the authors in their home. Thus, there is a need for longitudinal studies able to provide substantial empirical feedback on the usability and appropriation (Pipek, 2005) of this approach.

In the EUD for IoT perspective also social and humanoid robots play an important role: they can be seen as integrated sets of sensors and actuators, thus IoT platforms can make it easier to monitor and control them (Jalamkar and Selvakumar 2016). In general, there is a distinction between industrial robots, developed to accomplish specific tasks in specific work environments, and social humanoid robots, usually exploited in environments where they coexist and must relate with human beings (see an example in Figure 2). Thus, they can help us at our jobs, in housework, in the care of children, elderly and disabled people, in hospitals, schools, hotels and so on. Such robots interact with us by voice, gestures and all the other modalities typical of human communication. The available robots can be programmed through some programming language, which is usually oriented to engineers and requires considerable effort to learn. Thus, the issue of making the development of robot applications easier has started to be considered as well.

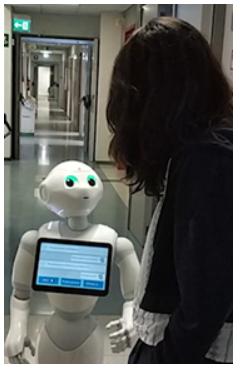

Figure 2 – An example of a humanoid robot.

What has been done so far to facilitate EUD in this area has mainly consisted in applying iconic data flow visual languages, such as Choregraphe (Pot et al., 2009) or the use of some block-based programming languages (Laval 2018, Weintrop et al, 2018). However, such solutions seem to work well when they consider scenarios in which the possible options to address are quite limited, but this is not true with modern humanoid robots which can flexibly react to many possible events and perform a wide variety of actions. Recent work (Leonardi et al., 2019) has aimed to investigate whether adopting a trigger-action paradigm, such as the one supported by tools such as IFTTT and Zapier[5], can enable people without particular programming knowledge to personalize the behaviour of humanoid robots. The goal is to exploit its compact and intuitive structure connecting dynamic situations with expected reactions without requiring the use of complex programming structures. Since humanoid robots can be used in various every day environments (equipped with several IoT devices/things/sensors), the possibility to detect what happens in the surrounding environment opens the way to exploit triggers that use the data detected by both the robot and IoT objects, and to link the robot behaviour to what happens around it. Thus, EUD tools should provide users with suitable techniques for specifying such triggers to describe context-dependent robot behaviour. However, that study (Leonardi et al., 2019) was an in-lab test, in which users received explicit task assignments to limit the possibility of ambiguity and to better compare the collected results. In order to have more precise indications on the effectiveness of this approach it would be interesting to challenge users through less explicit task instructions, and conduct longitudinal studies assessing the use of the tailoring tool for longer periods of time, to investigate whether further aspects emerge (e.g. if and how the way to personalise the robot would change over time).

A DESIGN SPACE FOR EUD OF IOT AND ROBOTIC APPLICATIONS

In order to analyse the main current approaches, and then discuss the future challenges, we find useful to introduce a design space, which is based on previous work (Paternò and Santoro, 2017), but that in this paper we better define and extend, in order to consider robotic applications as well. The purpose of the design space is to identify the main aspects that characterise methods and tools for EUD of IoT and/or robotic applications. As we introduced, for the robotic part we are mainly interested in humanoid robotic applications as they are more likely to be encountered in daily life scenarios, and share some aspects with IoT applications since they contain various sensors and actuators in a human-like structure.

One first characterising concept of EUD approaches is the type of metaphor they adopt. They have to represent the development concepts to people without programming experience. Thus, they should use concepts and representations that are used in the users' real world, with the aim to be more immediately understandable. In this way, metaphors provide users with easily understandable cognitive hints expected to facilitate the creation or customisation of an application by decreasing the learning effort needed by a non-professional developer to manipulate programming concepts and artefacts. However, previous studies (Blackwell and Green, 1999) showed that poorly designed metaphors do not improve usability, thus suggesting caution in their introduction.

Figure 3 shows examples of metaphors that have been investigated for facilitating end user development. For instance, using the jigsaw metaphor each software component is seen as a piece of a puzzle and the shapes of the various pieces provide the cognitive hints needed to understand the possible compositions. Another metaphor often proposed is the pipeline, in which applications are represented graphically as directed graphs where nodes correspond to elementary services or components, and links (i.e. pipelines) connect them. Often they are represented through icons associated with high-level functionalities, with some output and input ports representing the input and the output data, and the application development

[5] https://zapier.com/

mainly consists in indicating from where such functionalities receive input and where they send the results of their processing. The timeline is another relevant metaphor that has been considered. It basically provides a temporal reference along which events/objects are aligned, so helping in organising relevant information in a chronological order. Timelines are typically represented by a line on which various elements are graphically positioned, thus, in timelines the temporal relationships (between e.g. events) are basically represented as spatial relationships. TagTrainer (Tetteroo et al., 2015) is an approach exploiting timelines for caregivers to developing rehabilitation exercises for patients with hand or arm mobility problems based on the manipulation of everyday objects. Rules are a type of metaphor that seems particularly relevant for context-dependent applications since users can specify the desired behaviour by using a number of e.g. if-then statements expressing how the system should behave when specific situations occur. One of the first proposals using rules for EUD was iCAP (Dey et al. 2006). Recently, due to relevancy of contextual dynamic aspects that can potentially affect the behaviour of applications in IoT-based environments, rule-based approaches are receiving increasing interest. Indeed, IoT applications are characterised by the use of various sensors distributed in various points, and thus it is important to allow users to describe how the application should react to specific events detected by such sensors. HANDS (Pane et al., 2002) was an environment with similar goals, more oriented to children. It uses the cards metaphor: All objects in HANDS are represented by cards, which have user-defined properties, while the program execution, that is, the manipulation of cards, is represented by an agent.

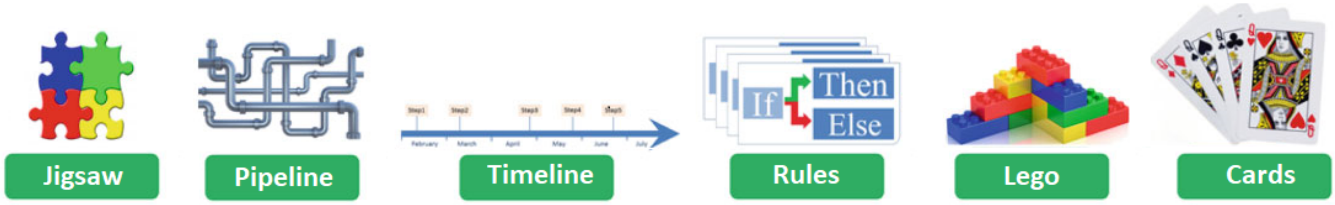

Figure 3 - Example metaphors for software development.

One distinct but often connected aspect is the programming style offered by the considered EUD approach. For programming style, we consider how the EUD environment allows the creation or modification of an application. Thus, in this case we focus on the concrete solution in terms of programming. Examples of programming styles are programming by example, trigger-action –based approaches, natural language techniques, mashups, mock-up –based and tangible programming techniques. The programming style based on user interface mock-ups as design tools (Beaudouin-Lafon, M. and Mackay, 2002) has long been considered due to its intuitiveness and effectiveness, and various tools for rapid prototyping for early stages of design, and iterative and evolutionary prototyping have been proposed. They can still be useful in IoT domains as well. One programming style relevant for EUD is based on natural language, a way of programming using a subset of constructs expressed in natural language which should model the user's intents. An example approach exploiting this programming style can be found in the work of Perera et al. (2015), which analyse how a natural language approach can support the definition of policies to manage the domestic environment. The authors consider the "sticky note" technique for defining the tasks requiring information exchange between IoT appliances and services. The findings reveal mainly that: the average number of words per note was relatively small; people in general adjust their language depending on the type of addressee (human vs. machine); and their technical background affects the way users communicate with machines. Another relevant approach in this area is represented by tangible interfaces, where a person interacts with digital information through the physical environment. An example of tangible metaphor is the fridge magnet metaphor used in (Truong et al., 2004). It mimics refrigerator magnets where the magnets offer a set of words that users can arrange into phrases. It also provides an interface for automated capture and playback (which allows users to replay events that were automatically

recorded in the home). Mashup refers to a composition of contents and/or features from several sources that determines new client-side interactive applications. It is an approach that has mainly been considered with Web technologies, since they are more open and facilitate it. One limitation is that it only supports the possibility of creating new applications from existing components, but not creating new components from scratch. Trigger-action programming seems to meet well the requirements for IoT applications since it allows their personalization according to the dynamic events and conditions that can be identified through the wide variety of possible sensors. It is an approach that has stimulated interest also at a commercial level. For example, IFTTT has more than 320,000 automation scripts (called "applets") offered by more than 400 service providers. The applets have been installed more than 20 million times, and more than half of IFTTT services are IoT device-related (Mi et al., 2017). The programming-by-example style allows the user to furnish examples of sequences of interactions from which the environment understands what the corresponding expected general behaviour is. It has been applied in various domains. For example, Improv (Chen and Lin, 2017) aims to support end users in dynamically defining cross-device interactions in order to leverage the capability of additional devices. Thus, users first demonstrate the target UI behaviour using the native input on the primary device. Improv parameterizes the user-demonstrated behaviour. Then, the user demonstrates the input on an accessory device, and the tool associates it with the parameterized behaviour so that the user can obtain the same original application behaviour through the cross-device interaction demonstrated.

Another design dimension that we have deemed useful for our analysis is the platform supported for the development activities. Traditionally it had been the desktop, but other platforms are being more and more considered, e.g. mobile, or even multiple platforms can be used, e.g. desktop and mobile together (Chen and Li, 2017). An indication of the relevant application domains which the concerned EUD approach can be applied to is useful as well. The domain can vary depending on the case; in the IoT area examples of application domains often considered are home automation, ambient assisted living, rehabilitation.

In the event dimension we consider the types of events that can have an impact on the behaviour of IoT applications. We have identified five possible categories of events: interaction events (i.e. events occurring when interacting with the application), those associated to aspects such as the user, the technology (e.g. appliances or devices), the surrounding environment (e.g. light, noise), and the social relationships. In addition to events, the possible actions should be considered as well. This level describes which type of changes/actions the considered EUD environment allows. Different types of actions can be identified, e.g. those performed in appliances (to change the state of actuators), user interface modifications (e.g. to change its presentation, content or navigation), execution of functionalities (e.g. access to an external service such as a weather forecast service). Associated to the latter two dimensions (event and action types) there are two more dimensions corresponding to the types of composition operators that are supported. Event compositions operators analyses the possibility to build composite expressions of events, which can be obtained in various manners, by using e.g. Boolean operators or temporal operators. Action compositions operators analyses the possibility to build composite expressions of actions. Constructs similar to those occurring in programming languages can be used (e.g. sequence, for, while, if).

A COMPARISON BETWEEN A SET OF REPRESENTATIVE APPROACHES

Taking into account the dimensions described before, in order to show how they can be used to analyse research contributions in this area, we have identified a number of representative contributions, which provide good coverage and a variety of values in such dimensions, and are well cited or have put forward novel work that seems worth discussing. In order to select them, we used various digital libraries and search engines (ACM, IEEE, Springer, ScienceDirect, Scholar). We focused on papers published in conferences and/or journals, and which also present some kind of evaluation. Each considered contribution

is associated with a row in the following table, in which the columns correspond to the dimensions identified, and the values of each row summarise how each approach supports them. In the table the first 13 contributions focus on EUD for IoT, the last 8 proposals focus on EUD for robotics; the division is highlighted by a bold horizontal line in the table. In the following we will describe such table according to each dimension.

Platform

In the works considered, the two development platforms that have been used most frequently are the Web (IFTTT, AppsGate, E-5W, meSchup, SmartFit Rule Editor, PersRobIoTE, Code3, English2NAO) and the desktop (TagTrainer, iCAP, TiViPE, InteractionBlocks, CoBLOX, Choreographe, Target-Drives-Means). The other platforms that have been considered are the mobile one (Epidosite, HomeBLOX, Puzzle) and also the tangible platform, which typically appears used in combination with other possibilities (see Gallagstrip, HomeRules and T4Tags2.0). Over the years, it can be seen that, in both the IoT and the robotics domain, the Web is becoming the most preferred one, probably due to its flexibility in accommodating various form-factor devices, which can be effectively exploited with responsive design.

Domain

As for the application domains considered, in the area of EUD for IoT, the home is the most preferred one (AppsGate, Epidosite, HomeBLOX, T4Tags 2.0, HomeRules). This is not by chance: the advancement of IoT has led to a plethora of Web-enabled devices and services in smart homes which have made them one of the most popular testbeds for developing systems for personalization and management of automated tasks based on the needs, goals, and routines of their residents. Another recurrent domain is the one, more general-purpose, focusing on task automation (IFTTT, Puzzle, E-5W). The application domain of context-aware applications has been considered in iCAP and GALLAG Strip, while the domain of e-wellness has been considered in SmartFit Rule Editor. The area of assistive applications and rehabilitation therapies have been of interest for both EUD in IoT-enabled contexts (TagTrainer) and EUD for robotics (TiViPE and English2NAO), as in these sectors the need to have personalized customizations targeting the specific needs, skills and abilities of patients is a key aspect. Regarding EUD for robotics, the considered domains are those focusing on programming the behaviour of humanoid robots (e.g. PersRobIoTE, Choreographe), autonomous dynamic robots (Target-Drives-Means), mobile manipulator robots (Code3) as well as robots collaborating with humans in industrial settings (CoBLOX). Programming the human-robot interaction/dialogue is the primary goal of the Interaction Blocks system.

Events

Regarding the events, all the approaches consider interaction events, whereas much fewer approaches consider the full range of event types (interaction, user-related, environment-related, technology-related, social relationships-related). As for the events considered in the EUD approaches for robots, all focus on the events detected by robot's embedded sensors, which can be considered part of the Technology category. In addition, the environment in which the robots act has also been considered when it is needed to specify a context-dependent robot behaviour. Therefore, other recurrent events in the area of developing robot behaviour for IoT settings are those associated with the Environment category (for instance, events related to environment noise, motion, time). As for the works focusing on EUD for IoT, all of the contributions have considered the events associated with Interaction, Environment and Technology categories. Some of them also consider events related to user characteristics (IFTTT, TagTrainer, Puzzle, E-5W, iCAP, SmartFit RuleEditor), for instance user preferences, skills, position, posture.

Metaphors

Looking at this dimension, the most used representation adopted to make intuitive the specification of the meant behaviour is the one based on rules (IFTTT, AppsGate, E-5W, iCAP, meSchup, T4Tags 2.0, SmartFit Rule Editor, HomeRules, PersRobIoTE), which has been generally translated, in terms of programming style, to trigger-action rules or one of its variants (if-then-else, event-condition-action).

Beyond rules, also timelines have been exploited in some tools (TagTrainer, InteractionBlocks). Components-based metaphors have also been used in HomeBLOX, TiViPE and Target-Drives-Means. The jigsaw/puzzle metaphor has been used both in EUD for IoT (Puzzle), as well as in the area of EUD for robotics (see Code3 and CoBLOX). Another metaphor adopted is the comic strip, used in GALLAG Strip, which shows action states in a sequence of frames, and enables programming by physical demonstration of envisioned interactions with the same sensors and objects that that are part of users' daily lives (rather than their models or abstract representations).

Programming style

Beyond the programming styles used to implement the rule metaphor (i.e. the "if-then" rules or the event-condition-action one), other recurrent programming styles are Programming by Demonstration (Epidosite, HomeRules, GALLAGStrip, Code3), in which the user performs actions on concrete examples and then the system records these actions and infers a generalized program that can be applied to new examples, and block-based programming (see e.g. TiViPE and CoBLOX), where instructions are mainly represented as blocks. A similar approach is used in Choreographe, where an iconic flow (or pipeline) of connected icons represents the robot behaviour to obtain. Natural language is another programming style that has often been used to make programming interfaces more approachable especially for novice users. It has often been used in combination with other approaches such as rules (this is the case of e.g. AppsGate, PersRobIoTE). A less used programming style is the process-driven one (homeBLOX), which exploits processes to specify not only automation sequences but also contextual situations that will trigger automations. Context information is aggregated as a result of a process, rather than being described as a Boolean expression of triggers (as typically happens with trigger/action –based approaches). Target-Drives-Means uses a programming paradigm that is quite different from those that usually consider the behaviour of a system (a robot in this case) as a sequence of actions. Instead, it is based on a number of reusable pre-programmed components (with information flowing between them), which are grouped together in higher-level behaviours that all potentially run in parallel and compete for activation (the logic of the program is specified by associating conditions that should activate functions).

Table 1: Analysis of a set of representative contributions according to the proposed design space

Name	Platf.	Domain	Events	Metap.	Progr. style	Actions	Event Comp.	Act. Com.	Further Support
IFTTT https://ifttt.com	Web	Task Automa.	Int Tech Env Use	Rules	TA rules	Devices; Web services	Not support.	SEQ	Inform users when rules are executed and failures
AppsGate (Coutaz & Crowley, 2016)	Web	Home	Interac. Tech Environ	Rules	If-Then-Else+ Nat.lang.	Devices and services available at home	Not support.	SEQ	Simulation (Time-based) + Debug (Timel. + Depend. Graph)
TagTrainer (Tetteroo et al., 2015)	Desk.	Assistive	Interac. User Tech.	Timelines.	Tangible	Exercises/actions involving physical objects	AND	SEQ	Tool notifies users about errors or inconsistencies during creation
Puzzle (Danado & Paternò, 2014)	Mob.	Task Automation	Int Tech Soc Env Us	Jigsaw		Application UI, appliances and devices	Not support.	SEQ /LOOP	
E-5W (Desolda et al., 2017)	Web	Task Automation	Int Tech Soc Env Us	Rules	TA rules	Composition of multiple objects and services	AND OR	SEQ	
Epidosite (Li et al., 2017)	Mob.	Home Automation	Interac. Env. Techn.		Progr. By Example.	Control smart appliances and devices at home	Not support.	SEQ	
iCAP (Dey et al., 2006)	Desk.	Context-aware app	Int Tech Soc Env Us	Rules	If-then rules	Context-aware applications	AND OR Tempor.	Not supp.	Rule simulation + Ambiguity check + Resolution of conflicts
homeBlox (Rietzler et al., 2013)	Mob.	Smart homes	Interac. Env. Tech.		Proc.-driven Graphs	House appliances/devices & services	AND OR	AND OR	
meSchup (Kubitza & Schmidt, 2015)	Web	Smart Environ.	Interac. Env. Tech.	Rules	If-then-else+Mashup	Composite behaviour of smart devices	Not support.	Not sup.	Simulation
T4Tags 2.0 (Bellucci et al., 2019)	Web + Tangib	Home	Interac. Env. Tech.	Rules	T-A rules	Mail, sounds, manage power outlet & light	AND OR	SEQ	Informs user if trigger compos. is always True/False
SmartFit (Barricelli & Valtolina, 2017)	Web	Wellness	Int Tech Soc Env Us	Rules	ECA rules	Suggestions, warnings, messages	AND OR Tempor.	SEQ	
HomeRules (De Russis & Corno, 2015)	Mob.+ Tangib	Home	Int. Env. Tech.	Rules	ECA rules + PbD	Controlling smart devices at home	OR	SEQ	The user can disambiguate the rule structure
GALLAG Strip (Lee et al., 2013)	Mob. + Tangib	Context-aware app	Inter. Env. Tech	Comic Strip	Progr. by Demonstrat.	Home appliance & devices, reminders,	AND	SEQ	
PersRobIoTE (Leonardi et al., 2019)	Web	Humanoid robots	Int Tech Soc Env Us	Rules	TA rules+ Nat.lang.	(Pepper) robot; IoT appl./dev.; alarm/notif.	AND OR NOT	SEQ PAR	Simulation + Debug (why/why not) + Conflict detection/resolution
TiViPE (Barakova et al., 2013)	Desk.	Assistive	Int. Env. Technology	Blocks	Textual commands	(NAO) robot: wait, LEDs, audio, motor	Not support.	SEQ PAR	
Code3 (Huang & Cakmak, 2017)	Web	Manipul. Robots	Interac. Env. Tech.	Jigsaw	Progr. by Demonstrat.	(PR2) Robot perceiv./manipul. objects/environ.	Not support.	SEQ	
Interaction Blocks (Sauppè & Mutlu, 2014)	Desk.	Human-Robot Interact.	Interac. Env. Techno.	Timel.	Pattern-based	User Dialogue; (NAO) robot mov. (gaze, head)	Not support.	SEQ	
CoBlox (Weintrop et al., 2018)	Desk.	Industrial robots	Interac. Env. Technology	Jigsaw	Templates	1-arm(Roberta) robot: arm, place object, gripper	Not support.	SEQ	Virtual Robot simulator
Choreographe (Pot et al., 2009)	Desk.	Humanoid robots	Interaction Env. Tech.	Pipeline		Controlling (NAO) robot: move, speech, LEDs	Not support.	SEQ PAR	
English2NAO (Buchina et al., 2016)	Web	Assistive	Interaction Env. Tech.		Visual Not. + NatLang.	(NAO) robot (speech, object interaction)	Not support.	SEQ	
Target-Drives-Means (Berenz & Suzuki, 2014)	Desk.	Autom. Dynam. Robot	Int. Env. Technology	Comp.	Parallel behaviors	Robot behav. move head, pick/search	AND	SEQ PAR	User specify priorities for disambiguation

Actions

The actions that have been supported are strictly connected with the specific application area that has been considered in each approach. For instance, the approaches focusing on smart homes and, more in general on smart environments, involve actions allowing the management of smart devices, appliances and services typically available at home and in smart environments (see AppsGate, Epidosite, HomeBLOX, T4Tags 2.0, HomeRules). Suggestions/reminders/warnings directed to users have been considered in SmartFit Rule Editor.

As for works associated with robots, all of them focus on managing the tasks that the considered robot is expected to carry out. However, some of the works encompass a wider set of actions. For instance, PersRobIoTE not only provides means for controlling the (Pepper) robot considered, but also for controlling the appliances and smart devices available in the users' surrounding environment, as well as providing some alarms and reminders to them. In this case, the integration between the humanoid robot and IoT smart contexts has been supported by providing suitable techniques for specifying context-dependent behaviour not only of the robot but also of the surrounding IoT-enhanced environment.

Event composition

The composition of events has been considered only in some cases, also due to the cognitive effort required by users to specify such composite situations. In the works that considered the specification of more flexible contextual conditions, the composition is typically supported through Boolean operators (e.g. AND, OR) and even NOT as in PersRobIoTE (to indicate when something does not happen in a period of time). This is the case of TagTrainer, E-5W (in which all the events in a rule are either in AND or OR), iCAP, homeBLOX, T4Tags2.0, SmartFit Rule Editor, homeRules, GallagStrip, PersRobIoTE. Among them some works also consider the possibility of specifying time-related constraints on events. For instance, in iCAP events can be related through e.g. 'before' and 'after'.

Action composition

Regarding the combination of actions, several proposals consider it, the most used approach being simply to have a sequence of actions (TagTrainer, E-5W, Epidosite, T4Tags 2.0, SmartFit Rule Editor, homeRules, GALLAG Strip). Recently also IFTTT supports the composition of actions through the new 'Maker' tier which allows developers to add multiple triggered actions. In the area of EUD for programming robotic behaviour, the need to develop natural (and complex) robot behaviour even resembling humans has led to works that provide means for better structuring such behaviour. Therefore, all the works support the possibility to combine the possible actions in a sequence of actions, whereas some of them also consider the possibility to combine the actions in parallel, since the robot is able to perform different actions at the same time through its various components (see e.g. PersRobIoTE, TiViPE, Choreographe, Target-Drives-Means). In this regard, it is worth mentioning Target-Drives-Means that directly structures its programming style in terms of parallel behaviours competing for activation.

Further support

Regarding the further support that such tools offer to their end-user developers, less than a half of them support users during their development, basically to better validate the correctness of the specified behaviour. The most frequent type of support is based on *simulation* of the programmed behaviour in a virtual context (see e.g. iCAP, meSchup, AppsGate, PersRobIoTE). For instance, in AppsGate there is the possibility to run programs using a virtual date and time. In addition, still with AppsGate, end user developers are further supported in monitoring the programmed behaviour by means of *timelines* (to let users monitor home states over time) and *dependency graphs* (which let users monitor home states through relations between entities such as devices and rules). TagTrainer offers a validation tool that notifies users about errors or inconsistencies during creation of programs. iCAP, apart from providing end users with the possibility of simulating rules, also offers support for checking rule ambiguity and resolution of potential conflicts. T4Tags 2.0 restricts the choice of the triggers that can be combined through

conjunction and disjunction: the toolkit guides the user towards implementing correct behaviour and avoids the creation of rules that lead to inconsistent/invalid states. The interface does not allow conjunction of two event triggers (as it is a very unlikely situation). If the conjunction or disjunction of two triggers results in a rule that always returns the same value (True or False), the system provides a prompt to the user, informing that the specific behaviour cannot be implemented, explaining the cause. HomeRules assists end users during the definition of rules highlighting possible problems, also allowing step-by-step simulation of rules. PersRobIoTE offers means for simulating rules, automatically detecting possible conflicts and also suggesting ways to solve them, as well as debugging features exploiting the why-why not paradigm. Also CoBLOX offers a virtual root simulator. Using Target-Drives-Means the users themselves have to specify behaviour priorities to disambiguate possible ambiguous situations (this feature is also available in the work of PersRobIoTE). IFTTT provides another type of support that aims to inform users whenever a rule is executed, and in case of failure provides associated feedback as well.

DISCUSSION

As a result of this analysis, a number of observations and implications can be derived, also in terms of potential areas that require further research in the near future.

Looking at Table 1, first of all, one area that seems to require further development regards intuitive ways to flexibly and more expressively specify the situation/context of interest that should trigger expected behaviour, especially in EUD for robotics, where many approaches do not support this event-composition –related aspect at all. While the use of Boolean expressions is the preferred approach that has been used up to now, there is the need to avoid the specification of complicated Boolean expressions, which may be difficult for users to specify, control, and maintain over time.

Another aspect that has stimulated less interest in the past but recently is attracting increasing attention is the support that should be offered to end-user developers to somehow facilitate the adoption of EUD in their lives. Till now, this has mainly been done by means of providing automatic support prompting users about ambiguous situations or situations that require further attention, or automatically identifying situations that cannot be verified, and thereafter suggesting a suitable solution.

In addition, the combination of multiple interaction modalities for EUD seems another promising direction (although exploited only in a few works), since it allows for better exploiting the advantages of different modalities and therefore it should facilitate end users in approaching such tools. In this regard, the proliferation of IoT and the widespread adoption of sensing and interaction technologies make multimodal EUD tools a promising and timely approach able to assist users in specifying the behaviour of intelligent environments. In particular, multimodal platforms exploiting the tangible modality are of special interest due to their natural connection with physical smart things of IoT environments, better supporting the transition from the digital world to the real one. More general implications can be derived, as elaborated in the following.

Support for managing complex, real-life personalization

As it has already been highlighted in the work of (Bellucci et al., 2019) and (Brich et al., 2017), the trigger-action paradigm, even enhanced by AND/OR compositions, turns out to be good for modelling and programming basic and well-structured situations (an example was reported in domestic settings), but not perfect for managing more complex scenarios and task automations, which may need further, more expressive programming paradigms for supporting the different, ever-changing needs of users, who can have varying expertise (and interests) in technology and programming. Also in (Bellucci et al., 2019) it is

highlighted how some personalization scenarios either require specialized functionality (such as dedicated applications), or cannot even be expressed using the trigger-action paradigm. In this regard, enhanced programmability, such as programming by example and especially process-driven approaches, could represent an interesting option that should be further pursued.

Offer the most appropriate abstraction level to end-user developers

The specification of context-dependent behaviour can be supported at various levels in programming, from basic hardware primitives to social behaviour. EUD tools should be able to hide the complexity of the many underlying technologies involved, and highlight the main conceptual aspects that need to be understood and manipulated through intuitive metaphors and programming styles. For instance, in the case of a context-aware application it is important to identify the relevant situations triggering some specific behaviour without having to learn the low-level sensing technology, and enable setting the consequent reactions in a condensed and understandable manner, so that even non-professional programmers can make sense of them.

Users need the most appropriate metaphor for their routines and practises

As we have seen, there are many different abstractions for IoT end user developers to choose from. They all aim to hide the implementation details at different levels and through different representations. One issue to consider is that many users are not used to thinking in terms of algorithmic computation, and thus need representations and concepts more suitable for them. Therefore, the selection of the most suitable metaphor should be done properly, because end user developers will accordingly form a different mental model of the system, and this will also impact the specific personalization capabilities. In addition, the intuitiveness of the metaphor will also affect the way in which end users will include EUD activities in their everyday routines and life practises.

Users need effective means to understand and validate the correctness of the specified behaviour

Defining context-dependent behaviour may imply specifying several contextual situations where various behaviours should occur. When there are many such possible behaviours, the resulting intertwined behaviour could be difficult to control especially by people with limited or no programming expertise at all. In addition, in real contexts, users will likely experience some surprising behaviours, wondering why a certain unwanted behaviour occurred in a specific situation. When this happens (i.e. there is a mismatch between users' expectations and actual system behaviour), non-expert IoT users should be provided with tools that clarify the underlying rationale and logic of the system. In order to effectively localize issues in the currently specified behaviour, users should be supported by explanations (better if they are provided in natural language) of the reasons for the unexpected behaviour, possibly accompanied by concrete examples or counterexamples highlighting the situations in which a specific rule is verified or not. Also simulation modes represent a good direction to enable people with less programming expertise to maintain full control of their IoT environment by providing a safe environment where users can try their automations and also test new ideas. However, in order to be effective and actually adopted by users, the simulation environment should reflect the current situation at least to a certain degree (even if it often changes, as happens in domestic environments), and it should exploit meaningful yet intuitive visualization techniques to enable people to make sense of the potentially very large set of devices and smart objects to control (Brich et al., 2017). However, it is worth pointing out that, in specific domains (e.g. assistive applications), end users might not feel sufficiently sure of using the simulation environment. For instance, in (Buchina et al., 2016) it emerged that caregivers felt confident of implementing their scenarios for their patients (ASD children) only when having the possibility to test them concretely with the robotic assistant.

Integration of humanoid robots with IoT environments

Differently from industrial robots, humanoid robots can be used in various everyday environments (equipped with several IoT devices/things/sensors). The possibility to detect what happens in the

surrounding environment opens the way to exploit triggers that use the data detected by both the robot and IoT objects and to link the robot behaviour to what happens around it. Thus, EUD tools should provide users with suitable techniques for specifying such triggers to describe context-dependent behaviour integrating the capabilities of the smart IoT environment and the robot.

Support and promote user creativity and reuse of programs

Since the behaviours to specify tend to be multi-faceted, we should provide users with means to re-use already created rules or even parts of such rules (e.g. blocks of actions) through familiar concepts in order to easily refer to them in further, more structured rules. In addition, providing the possibility to share rules (as it happens e.g. in T4Tags 2.0 and also PersRobIoTE) supports users' sustained engagement and creativity: users can be inspired by system usage from other users as well as adopt or tailor others' customizations for their personal needs.

Integration of intelligent techniques to support EUD.

End users can be helped in the identification of relevant ways to modify IoT and robot applications by some kind of intelligent mechanisms. Various possible approaches can be possible in this perspective. One possible approach is to exploit machine learning to detect the relevant rules from the actual user behaviour. For example, an intelligent thermostat is able to detect at which temperature the heating system should be turned on based on previous user choices. A more general solution is rather difficult because it implies the ability to monitor in a continuous and stable way many possible relevant contextual aspects and actions performed (corresponding to changes in the appliances status or commands sent to applications). Another possible approach is to introduce a rule recommendation system. Some can be obtained through generalization of the content of some part of the existing rules. Thus, for example, if there is a rule that says that when the user enters the bedroom then the lights should be on, one possible suggestion obtained through generalization can be that when the user enters any room then the lights should be on. Another type of recommendation can be obtained by trigger refinement, which means to narrow when the trigger should be fired by adding conditions that make the rules meet the needs more precisely.

CONCLUSIONS

In this paper we discuss the historical evolution of EUD, then we analyse the main current challenges with respect to recent technological trends (IoT and social robots) through the use of some conceptual dimensions, and conclude with a discussion of a possible research agenda for the field. The presented conceptual framework is useful to facilitate a better understanding of the important aspects to consider when designing EUD environments for IoT and/or robots, and it can be used as the basis for comparative analysis amongst various approaches and inform discussion about areas that can be further investigated. Additional aspects that are currently starting to emerge include new evaluation techniques that would facilitate effective in-the-wild evaluation of EUD approaches in longitudinal studies to investigate how the use and appropriation of the tools vary over time. Indeed, most usability studies in this area have been carried out with a limited number of users, and often in laboratories. In addition, another interesting direction of research is represented by aspects associated with how to better support end-user awareness of relevant security and privacy concerns in IoT settings, by identifying intelligent and usable mechanisms for controlling them.

REFERENCES

Aghaee, S., Pautasso, C., 2014. End-User Development of Mashups with NaturalMash. J. Vis. Lang. Comput. 25(4), pp. 414-432.

App Inventor MIT (2010), http://info.appinventor.mit.edu/

Atzori, L., Iera, A., and Morabito, G., 2010. The Internet of Things: A survey. Computer Networks, Volume 54, Issue 15, 28 October 2010, Pages 2787–2805. doi:10.1016/j.comnet.2010.05.010

Barricelli BR, Cassano F, Fogli D, Piccinno A, 2019, End-user development, end-user programming and end-user software engineering: A systematic mapping study, Journal of Systems and Software 149, 101-137

Barricelli, B.R., Valtolina, S.: A visual language and interactive system for end-user development of internet of things ecosystems. J. Vis. Lang. Comput. 40: 1-19 (2017) -

Beaudouin-Lafon M., and Mackay W., 2002. Prototyping Tools and Techniques. The Human Computer Interaction Handbook. J.A. Jacko and A. Sears, eds., p.1006–1031.

Bellucci, A., Vianello, A., Florack, Y., Micallef, L., Jacucci, G.: Augmenting objects at home through programmable sensor tokens: A design journey. Int. J. Hum.-Comput. Stud. 122: 211-231 (2019).

Berenz, V., Suzuki, K.: Targets-Drives-Means: A declarative approach to dynamic behavior specification with higher usability. Robotics and Autonomous Systems, 62(4), 545-555 (2014). doi: 10.1016/j.robot.2013.12.010

Blackwell, A.F., 2002. First steps in programming: a rationale for attention investment models," IEEE Symposia on Human-Centric Computing Languages and Environments, pp. 2-10.

Blackwell, A.F. & Green, T.R.G. (1999). Does Metaphor Increase Visual Language Usability? In Proceedings 1999 IEEE Symposium on Visual Languages VL'99, pp. 246-253.

Bellotti, V, and Edwards, W.K., 2001. Intelligibility and Accountability: Human Considerations in Context-Aware Systems, Human-Computer Interaction, 16(2-4): 193-212.

Brich J., Walch M., Rietzler M., Weber M., Schaub F., (2017) Exploring End User Programming Needs in Home Automation, ACM Transactions on Computer-Human Interaction (TOCHI) 24 (2), 11

Buchina, N., Kamel, S., and Barakova, E.I., 2016. Design and evaluation of an end-user friendly tool for robot programming. In Proceedings of IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN ‘16). IEEE, 185-191. DOI: https://doi.org/10.1109/ROMAN.2016.7745109

Burnett M., 2009. What Is End-User Software Engineering and Why Does It Matter? In: Pipek V., Rosson M.B., de Ruyter B., Wulf V. (eds) End-User Development. IS-EUD 2009. Lecture Notes in Computer Science, vol 5435. Springer, Berlin, Heidelberg

Burnett M., and Scaffidi, C., End-User Development. The Encyclopedia of Human-Computer Interaction, 2nd Ed, cap.10 https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/end-user-development

Carmien, S.P., Fischer, G.: Design, Adoption, and Assessment of a Socio-Technical Environment Supporting Independence for Persons with Cognitive Disabilities. Proc. CHI 2008, pp. 597 – 606

Chen X., Lin Y., Improv: An Input Framework for Improvising Cross-Device Interaction by Demonstration, ACM TOCHI, 2017.

Cheverst, K., Davies, N., Mitchell, K., and Efstratiou, C., 2001. Using Context as a Crystal Ball: Rewards and Pitfalls, Personal and Ubiquitous Computing: Volume 5 Issue 1.

Corno, F., de Russis, L., and Monge Roffarello, A., 2017. A High-Level Approach Towards End User Development in the IoT. In: CHI 2017: The 35th Annual CHI Conference on Human Factors in Computing Systems, Denver, CO (USA), May 6–11, 2017. pp. 1546-1552

Coutaz, J., and Crowley, J.L., 2016. A first person experience with end-user development for smart home. IEEE Pervasive Computing, vol. 15, no 2, May-June 2016: 26:39

Cuccurullo, S., Francese, R., Risi, M., Tortora, G.: MicroApps Development on Mobile Phones. In Costabile, M., Dittrich, Y., Fischer, G., Piccinno, A., eds. : End-User Development 6654. Springer Berlin / Heidelberg (2011) 289-294

Cypher, A. (Ed.), 1993. Watch What I do: Programming By Demonstration. MIT Press.

Cypher A., Dontcheva M., Lau T., Nichols J., No Code Required Giving Users Tools to Transform the Web, Morgan Kaufmann, ISBN 978-0-12-381541-5, 2010

Danado J., Paternò F., Puzzle: Puzzle: A Mobile Application Development Environment using a Jigsaw Metaphor, Journal of Visual Languages and Computing, 25(4), pp.297-315, 2014, Journal of Visual Languages and Computing, 25(4), pp.297-315, 2014.

Dax, J., Ludwig, T., Meurer, J., Pipek, V., Stein, M., and Stevens, G., 2015. FRAMES – A Framework for Adaptable Mobile Event-Contingent Self-report Studies. IS-EUD 2015: 141-155.

Davidyuk O., Sanchez I., Gilman E. and Riekki J.: An Overview of Interactive Application Composition Approaches, Open Computer Science. Volume 5, Issue 1,ISSN (Online) 2299-1093, DOI: 10.1515/comp-2015-0007, December 2015.

De Russis, L., Corno, F.,: HomeRules: A Tangible End-User Programming Interface for Smart Homes. CHI Extended Abstracts 2015: 2109-2114

Desolda, G., Ardito, C., and Matera, M., 2017. Empowering End Users to Customise their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools, ACM Trans. Comput.-Hum. Interact. 24(2): 14:1-14:33, 2017

Dey, A.K., and Newberger, A., 2009. Support for context-aware intelligibility and control. CHI 2009: 859-868

Dey, A.K., Sohn, T., Streng, S., and Kodama, J., 2006. iCAP: Interactive Prototyping of Context-Aware Applications. Pervasive 2006: 254-271

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A., Mehandjiev, N., 2004. Meta-design: a manifesto for end-user development. Commun. ACM 47 (9), 33–37. http://doi. acm.org/10.1145/1015864.1015884 .

Fischer, G., & Giaccardi, E. (2006). Meta-Design: A Framework for the Future of End User Development. In H. Lieberman, F. Paternò & V. Wulf (Eds.), End User Development (Vol. 9, pp. 427-457). Dordrecht, The Netherlands: Springer. doi: 10.1007/1-4020-5386-X_9.

Ghiani G., Paternò F., Spano D., Cicero Designer: an Environment for End-User Development of Multi- Device Museum Guides, Proceedings EUD'09, Siegen, Springer Verlag, LNCS 5435, pp.265-274.

Ghiani G., Paternò F., Spano L.D., Pintori G., An environment for End-User Development of Web mashups, International Journal of Human-Computer Studies Volume 87, March 2016, Pages 38–64, Elsevier

Ghiani, G., Manca, M., Paternò, F., and Santoro, C., 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. ACM Trans. Computer-Human Interaction 24(2): 12:1-14:52

Huang, H., and Cakmak, M., 2015. Supporting mental model accuracy in trigger-action programming. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). ACM, New York, NY, USA, 215-225. DOI=http://dx.doi.org/10.1145/2750858.2805830

Huang, H., and Cakmak, M., 2017. Code3: A System for End-to-End Programming of Mobile Manipulator Robots for Novices and Experts. In Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI '17). ACM, New York, NY, USA, 453-462. DOI: https://doi.org/10.1145/2909824.3020215

IFTTT, https://ifttt.com/

Klokmose C. N., Eagan J. R., Baader S., Mackay W., and Beaudouin-Lafon M.. 2015. Webstrates: Shareable Dynamic Media. In Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology (UIST '15). ACM, New York, NY, USA, 280–290. DOI: http://dx.doi.org/10.1145/2807442.2807446

Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M., Wiedenbeck, S., 2011. The state of the art in end-user software engineering. ACM Comput. Surv. 43 (3), 1–44. doi: 10.1145/1922649.1922658

Ko, A.J., and Myers, B.A., 2004. Designing the whyline, a debugging interface for asking why and why not questions about runtime failures. In Proceedings CHI'2004: Human Factors in Computing Systems (pp. 151–158). Vienna, Austria.

Ko, A.J., and Myers, B.A., 2005. A framework and methodology for studying the causes of software errors in programming systems. Journal of Visual Languages and Computing, 16(1), 41–84.

Ko, A.J., and Myers, B.A., 2009. Finding causes of program output with the java whyline. In CHI'2009: Human Factors in Computing Systems (pp. 1569–1578). Boston, MA.

Ko, A.J., Myers, B.A., Coblenz, M., and Aung H.H., 2006. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. IEEE Transactions on Software Engineering, 33(12), 971–987.

Kubitza, T., and Schmidt, A., 2015. Towards a Toolkit for the Rapid Creation of Smart Environments. IS-EUD 2015: 230-235

Kulesza, T., Burnett, M.M., Wong, W.-K., Stumpf, S., 2015. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. IUI 2015: 126-137

Kulesza, T., Stumpf, S., Wong, W.-K., Burnett, M.M., Perona, S., Ko, A., and Oberst, J., 2011. Why-oriented end-user debugging of naive Bayes text classification. ACM Transactions on Interactive Intelligent Systems (TiiS) 1, 1 (2011).

Jalamkar D, Selvakumar AA (2016) Use of Internet of Things in a Humanoid Robot – A Review. Adv Robot Autom 5:149.

Laval J., 2018. End User Live Programming Environment for Robotics. Robotics & Automation Engineering Journal, 3(2), June 2018.

Lee, J., Garduño, L., Walker, E., and Burleson, W. 2013. A tangible programming tool for creation of context-aware applications. In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp '13). ACM, New York, NY, USA, 391-400. DOI: https://doi.org/10.1145/2493432.2493483

Leonardi N., Manca M., Paternò F., Santoro C., Trigger-Action Programming for Personalising Humanoid Robot Behaviour, 2019 ACM Conference on Human Factors in Computing Systems (CHI'19), Paper 445..

Leshed G., Haber E. M., Matthews T., Lau T. A.: CoScripter: automating & sharing how-to knowledge in the enterprise. CHI 2008: 1719-1728

Li T.JJ., Li Y., Chen F., Myers B.A. (2017) Programming IoT Devices by Demonstration Using Mobile Apps. In: Barbosa S., Markopoulos P., Paternò F., Stumpf S., Valtolina S. (eds) End-User Development. IS-EUD 2017. Lecture Notes in Computer Science, vol 10303. Springer, Cham Programming IoT Devices by Demonstration Using Mobile Apps, IS-EUD 2017

Lieberman, H., Paternò, F., Klann, M., Wulf, V., 2006. End-User Development: An Emerging Paradigm. In End User Development, Henry Lieberman, Fabio Paternò, and Volker Wulf (eds.). Springer, The Netherlands, 1-8.

Lim, B. Y., 2012. Improving understanding and trust with intelligibility in context-aware applications. PhD thesis, Carnegie Mellon University.

Lim, B. Y., and Dey, A.K., 2010. Toolkit to support intelligibility in context-aware applications. UbiComp 2010: 13-22

Lim, B. Y., Dey, A.K., and Avrahami, D., 2009. Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems, Proceedings CHI 2009, pp. 2119-2128, ACM Press.

Lucci, G., and Paternò, F., 2014. Understanding End-User Development of Context-Dependent Applications in Smartphones. HCSE 2014: 182-198

Macías, J.A. and Paternò, F. (2008). Customization of Web Applications through an Intelligent Environment Exploiting Logical Interface Descriptions. Interacting with Computers. Elsevier. Volume 20 (1), pp. 29-47.

Manca M., Paternò F., Santoro C., Corcella L., Supporting end-user debugging of trigger-action rules for IoT applications, International Journal of Human-Computer Studies, Vol.123, 56-69

Markopoulos P., Nichols J, Paternò F., Pipek V., End-User Development for the Internet of Things, ACM Transactions on Computer-Human Interaction (TOCHI) 24 (2), 9, 2017

Matera M., Picozzi M., Pini M., Tonazzo M.: PEUDOM: A Mashup Platform for the End User Development of Common Information Spaces. ICWE 2013: 494-497

Metaxas, G., and Markopoulos, P., 2017. Natural contextual reasoning for end users. ACM Transactions on Computer-Human Interaction (ACM TOCHI), Vol.24, Issue 2, Article N.13.

Mi, X., Qian, F., Zhang, Y., Wang, X.F., 2017. An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance, Proceedings of Internet Measurement Conference (IMC) '17, November 1–3, 2017, London, UK

Myers B. and Buxton W., "Creating Highly Interactive and Graphical User Interfaces by Demonstration," Proceedings of SIGGRAPH '86, Vol. 20, No. 4, ACM, Dallas, TX, August 1986, pp. 249 - 258.

Myers, B. A., Ko, A. J., Scaffidi, C., Oney, S., Yoon, Y. S., Chang, K., Kery, M. B., and Li, T. J.-J., (2017) Making End User Development More Natural. In: Paternò F., Wulf V. (eds) New Perspectives in End-User Development. Springer, Cham

Nardi, Bonnie A. (1993): *A Small Matter of Programming: Perspectives on End User Computing.* Cambridge, Massachusetts, MIT Press

Nichols, J., and Lau, T. 2008. Mobilization by demonstration: using traces to re-author existing web sites. In Proceedings of the Symposium on Intelligent User Interfaces. 149–158.

Pane J. F., Myers B. A., Miller L. B.: Using HCI Techniques to Design a More Usable Programming System. HCC 2002: 198-206

Paternò, F., 2013. End User Development: Survey of an Emerging Field for Empowering People, ISRN Software Engineering, vol. 2013, Article ID 532659, 11 pages, 2013. doi:10.1155/2013/532659

Paternò F, Santoro C., A design space for end user development in the time of the internet of things, in New perspectives in end-user development, Springer 43-59, 2017

Pausch R., Burnette T., Capeheart A.C., Conway M., Cosgrove D., DeLine R., Durbin J., Gossweiler R., Koga S., White J., Alice: Rapid Prototyping System for Virtual Reality. IEEE Computer Graphics and Applications, May 1995

Perera, C., Aghaee, S., and Blackwell, A.F., 2015. Natural Notation for the Domestic Internet of Things. Proceedings IS-EUD 2015: 25-41, Springer Verlag.

Pipek, V., 2005. From tailoring to appropriation support: negotiating groupware usage. University of Oulu. Retrieved from http://herkules.oulu.fi/isbn9514276302/isbn9514276302.pdf.

Pipek, V., and Wulf, V., 2009. Infrastructuring: Towards an Integrated Perspective on the Design and Use of Information Technology. Journal of the Association of Information Systems (JAIS), Volume 10, Issue 5, May 2009, 306-332.

Pot E., Monceaux J., Gelin R., Maisonnier B.. 2009. Choregraphe: a graphical tool for humanoid robot programming. In Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'09). IEEE, 46-51. DOI: https://doi.org/10.1109/ROMAN.2009.5326209

Repenning, A., 1995. Bending the Rules: Steps toward Semantically Enriched Graphical Rewrite Rules, 1995 IEEE Symposium on Visual Languages, Darmstadt, Germany, pp. 226-233, Sept. 5-9, 1995.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM 52, 60-67 (November 2009)

Rietzler, M., Greim, J., Walch, M., Schaub, F., Wiedersheim, B., and Weber, M., 2013. homeBLOX: introducing process-driven home automation. In (UbiComp '13 Adjunct). ACM, New York, NY, USA, 801-808.

Sauppé, A., and Mutlu, B., 2014. Design patterns for exploring and prototyping human-robot interactions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 1439-1448. DOI=http://dx.doi.org/10.1145/2556288.2557057

Smith, D.C., Cypher, A., Spohrer, J., 1994. KidSim: Programming Agents Without a Programming Language, Communications of the ACM 37(7), 54-67, July 1994.

Sutcliffe, A.G., and Papamargaritis, G., 2014. End-user development by application-domain configuration. Journal of Systems and Software 91: 85-99.

Tetteroo, D., Vreugdenhil, P., Grisel, I., Michielsen, M., Kuppens, E., Vanmulken, D., and Markopoulos, P., 2015. Lessons Learnt from Deploying an End-User Development Platform for Physical Rehabilitation. In Proceedings CHI '15. ACM Press, pp. 4133-4142. DOI=http://dx.doi.org/10.1145/2702123.2702504

Truong K. N., Huang E. M., Abowd G.D., "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home.", Proceedings of Ubicomp 2004, pp. 143-160

Ur, B., McManus, E., Yong Ho, M.P., and Littman, M.L., 2014. Practical Trigger-Action Programming in the Smart Home. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). 803-812.

Ur, B., Yong Ho, M.P., Brawner, S., Lee, J., Mennicken, S., Picard, N., Schulze, D., and Littman, M.L., 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 3227–3231. DOI: http://dx.doi.org/10.1145/2858036.2858556

WEF, World Economic Forum (2018) Internet of Things, Guidelines for Sustainability - REF 310517 - January 2018 http://www3.weforum.org/docs/IoTGuidelinesforSustainability.pdf

Weintrop, D., Afzal A., Salac J., Francis P., Li B., Shepherd D. C., and Franklin D.. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 366, 12 pages. DOI: https://doi.org/10.1145/3173574.3173940

Yarosh, S., and Zave, P., 2017. Locked or Not?: Mental Models of IoT Feature Interaction. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 2993-2997. DOI: http://dx.doi.org/10.1145/3025453.3025617