Norme e regole, tra adattamento e resistenza, nella città e negli insediamenti: la documentazione d'archivio e la costruzione reale / Norms and rules, between adaptiveness

Chapter 6

Interacting with Internet of Things

Fulvio Corno (Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, fulvio.corno@polito.it) [0000-0002-9818-0999]

*Luigi De Russis (Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, luigi.derussis@polito.it) [0000-0001-7647-6652]

Alberto Monge Roffarello (Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, alberto.monge@polito.it) [0000-0002-9746-2476]

Juan Pablo Saénz Moreno (Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, juan.saenz@polito.it) [0000-0003-0928-3089]

Abstract

The Internet of Things provides many opportunities for people, bridging the digital and physical worlds to enhance information access and data exchange among interconnected entities. It also opens the way to several interaction opportunities and challenges for those living within the spaces enabled by such technologies.
This chapter will examine the challenges of interacting with Internet-of-Things-enabled environments while conducting a state-of-the-art review of related efforts in different settings, including smart homes, healthcare, and industry. The chapter will mainly elaborate on the role of automation and personalization of these environments from the perspectives of end-users and developers. Finally, it will look for ways to bridge gaps between these two fundamental and complementary actors.

6.1 Introduction

The spread of the Internet of Things (IoT) has altered the way we interact with our surroundings, enabling people to connect and communicate with the physical world in previously unimaginable ways. IoT encompasses a vast network of interconnected entities – devices, online services, sensors, and actuators – that collect and exchange data, orchestrating a symphony of actions and responses.

As such, IoT-enabled environments have become ubiquitous, integrating those interconnected things into our daily lives, from smart homes to healthcare and wellbeing monitoring. Despite numerous advantages, the interaction with the IoT is subjected to various challenges, from the risks of "disconnecting" users from the physical world (Angelini, Couture, Abou Khaled, & Mugellini, 2018) to more specific issues like cognitive overload (Shirehjini & Semsar, 2016) and the psychological orientation of information sources (Kim K. , 2016). Other challenges, which are the main focus of this chapter, arise when users want to *customize* their IoT-enabled environments. Indeed, the interconnectedness and fusion of the physical and digital worlds not only enhances convenience but also opens up possibilities for tailoring experiences to meet individual needs and preferences, forming the basis for customizability opportunities that lie at the heart of the exploration of this chapter.

Currently, many IoT devices and systems tend to be customizable, either through some programming frameworks or with their own user interfaces and dashboards. This personalization effort empowers people to exert greater control over their environments, optimize resource utilization, and streamline daily routines. A person can write a script (e.g., using Home Assistant's Scripting functionality[1]) to receive a message every time their preferred weather forecaster gives a high probability of rain, for example. Another one can compose an *if-then* rule such as "*if* the weather forecast reports 90% of rains, *then* send me a message" through a suitable user interface (e.g., IFTTT[2]) to realize the same objective. These **automations** are the key ingredient behind the personalization of IoT-enabled environments, even if they are created and handled differently. As people have different needs, backgrounds, and expertise, available personalization tools for the IoT need to adapt and offer suitable alternatives. While professional software developers can be at ease if they need to write code to create automations for their home, an end user without any

[1] https://www.home-assistant.io/docs/scripts/, last visited on June 10, 2023
[2] https://ifttt.com/, last visited on June 10, 2023

technical background needs a set of options and tools to create similar automations. In a sense, the distinction between an end user and a developer becomes increasingly thin and intertwined. For instance, when personalizing the behavior of the various things in the IoT environment, end users are acting as developers (i.e., end-user developers) since they need to handle events and actions, face bugs in their automations, and can be exposed to possible unpredictable behaviors (e.g., a light that turns on at a random time). End users can personalize their IoT environments if developers provide them with the proper set of tools and options; developers can produce better work if they know end users' expectations and automation needs.

It is, therefore, fundamental to explore the perspective on IoT and personalization options under both point of views and understand the characterization of the IoT from both of them. This chapter aims to fulfill this goal by showing what is an IoT-enable environment and which are the available personalization options from the point of view of end users and developers, looking for ways to bridge any gap between the two. In the following, end users are intended as the inhabitants of the IoT-enabled environment, ultimately the final users of any IoT system. Often, they do not have a technical background and do not know how to code. Conversely, developers can be either novice or expert software programmers. As we have hinted before, both are fundamental actors for effectively *enabling* the personalization of IoT environments.

The chapter is structured as follows. It starts with background information and helpful terminology for understanding the general context. Then, it analyzes the end user perspective. Personalization of IoT environments by end users is contextualized with respect to the state of the art, reporting and discussing challenges, efforts, and possible solutions. In particular, the chapter identifies three main challenges for end users: 1) how they can properly and effectively *define* automations, 2) how they can *discover* them, and 3) how they can *debug* and understand any issue with them.
The developer perspective, similarly, describes the context, challenges, and their possible solutions

with respect to the state of the art. In this case, the two main challenges are: 1) how they *understand* IoT-enabled environments and 2) how they face *integration* issues.

Finally, the chapter concludes with reflections and recommendations on bridging the gap between these two perspectives.

6.2 Background

6.2.1 Definitions and Application Examples

Nowadays, the Internet of Things (IoT) is a well-established paradigm that has gained prominence in several aspects of our everyday lives (Stankovic, 2014). The definition of IoT has been enriched in the past, considering different contexts and use cases. For example, the Social Internet of Things (SioT) is a subset of the IoT family that integrates IoT with social networking (Shahab, Agarwal, Mufti, & Obaid, 2022). The Industrial Internet of Things (IioT) encompasses machine-to-machine interaction, playing a crucial role in enhancing comprehension of manufacturing processes and facilitating sustainable production methods (Sisinni, Saifullah, Han, Jennehag, & Gidlund, 2018). Internet of Everything (IoE) extends IoT by taking into account business and industrial processes, and it is based on four main pillars, i.e., people, data, process, and things (Miraz, Ali, Picking, & Excell, 2015). Another related concept is digital twins, i.e., virtual replicas of physical objects or systems that enable real-time monitoring, analysis, and simulation to improve performance and decision-making (Jones, Snider, Nassehi, Yon, & Hicks, 2020).

Given the different definitions, examples of IoT applications are common across various domains, including but not limited to smart homes, healthcare, IIoT, smart cities, and agriculture. The review by Alaa et al. (Alaa, Zaidan, Zaidan, Talal, & Kiah, 2017) provides a comprehensive overview of IoT applications in the smart home context, showing that researchers have investigated a number of use cases that range from security and privacy applications to energy management (see Figure 6 of the paper for further details). In the broader domain of smart cities, IoT can be used to enhance and

support a wide range of use cases, from the healthcare domain to the domains of smart energy, transportation, and governance (see (Bellini, Nesi, & Pantaleo, 2022), Figure 3). In the healthcare domain, in particular, IoT can be exploited for monitoring patients (Laplante & Laplante, 2016), either in hospitals or at home, e.g., by setting up Ambient Intelligence environments that allow learning about patient's data and executing actions (Darshan & Anandakumar, 2016). In IIoT, application examples can be found in the context of manufacturing, agriculture, and even in the military domain (Jaidka, Sharma, & Singh, 2020). In the agriculture domain, IoT is exploited to manage farm information systems and water resources, as well as to monitor diseases, fields, greenhouses, livestock, pests, and soil (Kim, Lee, & Kim, 2020).

Independently by the specific definition or use case, embedding computing and communication capabilities into objects of common use (Miorandi, Sicari, De Pellegrini, & Chlamtac, 2012) has given rise to a programmable world. It has encouraged the development of a broad range of solutions in several domains, such as smart buildings, smart cities, environmental monitoring, healthcare, logistics, smart business, smart agriculture, and security and surveillance (Al-Fuqaha, et al., 2015), (Zanella, Bui, Castellani, Vangelista, & Zorzi, 2014), (Islam, Kwak, Kabir, Hossain, & Kwak, 2015).

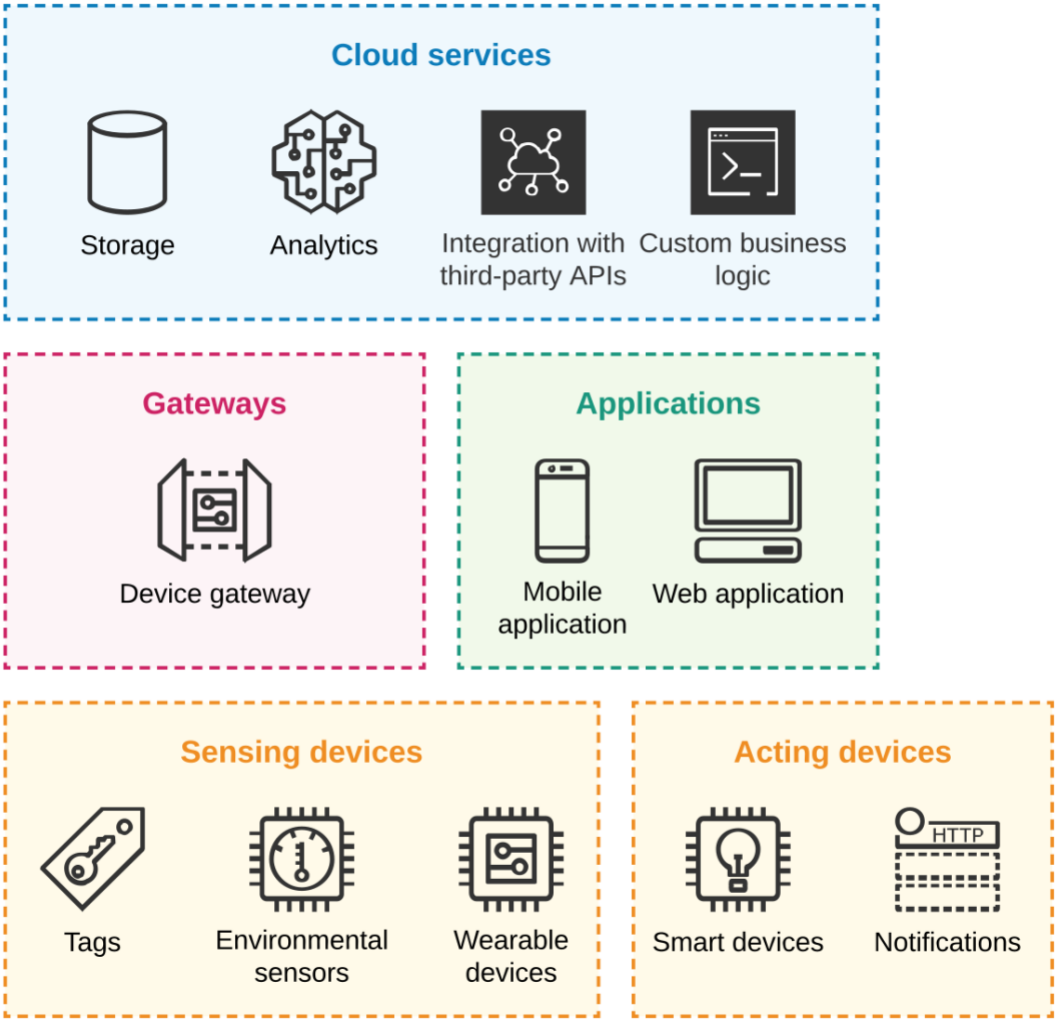

Figure 6.2.1. Architectural elements in IoT systems

As shown in Figure 6.2.1, from a technical point of view, IoT systems can be characterized by four principal architectural elements: devices, gateways, cloud services, and applications.

Devices are hardware elements with built-in communication capabilities that collect sensor data (**sensing devices**) or perform actions (**acting devices**). Sensing devices provide information about the physical entities that they monitor. This information may concern the physical entity's identification (tags) or measurable qualities such as temperature, humidity, pressure, luminosity, sound level, location, images, presence, and movement. They might be environmental sensors and wearable devices, in which case they tend to measure physiological quantities. Acting devices refer to smart

devices that cause or trigger changes in the physical environment, such as smart lights, motors, displays, etc. These acting devices also encompass push notifications through which end users are informed about the occurrence of a given event.

Gateways or 'edge' devices collect, preprocess, and forward the data from the *sensing devices* to the cloud and, conversely, route the requests sent from the cloud to the *acting devices*. They may support intermediate sensor data storage and preprocessing, gathering sensor data, and performing computation and reasoning tasks. If more computing or storage capacity is required, the *gateways* communicate with the *cloud services* and delegate the most demanding tasks. Furthermore, *gateways* also interact with the actuators; they control the *acting devices* based on the outputs from their computations or the instructions they receive from the *cloud services*. Additionally, *gateways* support other tasks such as service discovery, geolocalization, and verification.

Cloud services have three primary responsibilities: acquiring and storing data from the *sensing*, providing real-time and offline data analytics, and managing the *acting devices*. Data acquisition and storage concerns harvesting and storing a large amount of data collected by sensing devices for further processing and analysis. Providing real-time and offline data analytics refers to examining, cross-connecting, and transforming acquired sensor data to discover helpful information to support decision-making. Machine learning and data mining technologies and algorithms are essential in this regard. Managing the *acting devices* refers to generating and delivering remote notifications and interfacing with third-party APIs through which specific *acting devices* can be reached and managed.

Finally, **applications** range from web-based dashboards to domain-specific web and mobile applications and represent how end users can visualize the device's data and status, visualize the analyses' results, and interact with the overall system.

The implementation and orchestration of these architectural elements rely on several enabling technologies. According to (Atzori, Iera, & Morabito, 2010), these enabling technologies can be classified into identification, sensing and communication technologies, middleware components,

end-user software applications, services composition, service management, and object abstraction. While identification, sensing, and communication technologies mainly concern hardware components, the other enabling technologies rely on *software* to address various features that IoT systems expose (Miorandi, Sicari, De Pellegrini, & Chlamtac, 2012).

While this reference architecture represents a typical and general IoT system, there will be cases where the same elements might not be present. Even in those cases, the general function and structure depicted here remains valid.

6.2.2 Challenges and Opportunities

Prior work demonstrates that Interacting with the IoT can be subjected to numerous challenges that depend on the specific contexts in which such an interaction occurs. For example, the work by Angelini et al. (Angelini, Couture, Abou Khaled, & Mugellini, 2018) highlights that most of today's interfaces for IoT objects are based on smartphones or web apps. These interfaces tend to "disconnect" users from the physical world and isolate them into "digital bubbles," decreasing user experience. Shirehjini and Semsar (Shirehjini & Semsar, 2016), instead, warned that interacting with IoT may cause cognitive overload, e.g., when manually selecting devices in complex settings, loss of user control, and over-automation. The psychological orientation of information sources can also undermine the quality of human-IoT interaction, as the IoT complicates traditional models of communications (Kim K. , 2016).

Researchers have produced prominent efforts to mitigate the above challenges, e.g., by exploring innovative approaches and paradigms for human-IoT interaction. The Internet of Tangible Things (Angelini, Couture, Abou Khaled, & Mugellini, 2018), for example, has been proposed as a new IoT paradigm in which interfaces of IoT objects are situated in the physical world, rather than behind a smartphone or a website. According to the authors, designing objects that embody tangible interfaces can support immediate and meaningful interaction, stimulating reflection and the understanding of the system. Similarly, Kranz et al. (Kranz, Holleis, & Schmidt, 2009) explored the

concept of embedded interaction, i.e., the idea of "seamlessly integrating the means for interaction into everyday artifacts" (p. 46). Specifically, the authors proposed a set of prototypes to highlight novel opportunities for human-computer interaction. Kim (Kim K. , 2016), instead, investigated human-IoT interaction by adopting psychological and user-experience approaches rather than focusing exclusively on technological aspects, identifying social presence and perceived expertise as two important mediators. A comprehensive review of the Social Internet of Things has been recently produced by Shahab et al. (Shahab, Agarwal, Mufti, & Obaid, 2022).

Other researchers focused on creating novel interfaces for human-IoT interaction, leveraging advanced visualization techniques. For example, Shirehjini and Semsar (Shirehjini & Semsar, 2016) exploited 3D visualization to implement an interface for mixed-initiative interaction in smart environments. Phupattanasilp and Sheau-Ru (Phupattanasilp & Sheau-Ru, 2019), instead, used augmented reality to support IoT data visualization in the farming context.

Overall, all the above issues can be associated with the fact that IoT can be *personalized* by end users, who interact with their IoT ecosystems to "program" the behavior of smart devices and online services on the basis of their personal needs. Such a possibility creates a range of issues and opportunities, representing the core of this chapter. End users and developers are the two main actors that frequently interact and work within this context. The former are the people who adopt and use such technologies within their environment, while the latter are the creators and updaters of the entire IoT system or application. Given the flexibility and variability of available "connected entities" (e.g., sensors, actuators, online services), developers cannot predict all the possible contexts of usage of their creations, and end users might want to personalize the behavior of their IoT devices and systems to better fit their specific needs in the context at hand. The rest of this chapter is dedicated to exploring in detail such a dynamic interplay between these two actors, starting from end users.

6.3 The end-user perspective

In the Internet of Things realm, end users need to be able to personalize the functionality of smart devices and online services, even if they lack programming expertise. In this scenario, the vision of End-User Development (EUD) (Paternò, Lieberman, & Wulf, 2006) strives to empower end users, who possess firsthand knowledge of the specific requirements, to customize the behaviors of their "connected entities." Typically, such a customization is supported through the trigger-action programming paradigm, which allows users to define IF-THEN rules in the form of *"if something happens on a device or service, then execute something on another device or service."* This section reviews the state-of-art for EUD and trigger-action programming within IoT-enabled environments. Besides introducing the context of such a growing and pressing research topic, we present issues and related challenges, as well as solutions emerging from the Human-Computer Interaction (HCI) research community. Table 6.3.1 provides a summary of the issues, challenges, and solutions detailed in this section.

Table 6.3.1. A summary of the issues, challenges, and solutions reviewed in this section.

Issue	**Challenge**	**Solutions**
Low-level of abstraction: Trigger-action programming platforms adopt representation models that heavily depend on the underlying technology.	***Supporting IF-THEN Rules Definition:*** Simplify the processes end users need to define IF-THEN rules.	● Visual metaphors ● Underlying models ● Interaction techniques
Information overload: The number of potential combinations between triggers and actions supported by trigger-action programming platforms is continuously growing.	***Promoting IF-THEN Rules Discovery:*** Support users in discovering IF-THEN rules and related functionality useful to satisfy their personalization needs.	● Recommender systems ● Conversational agents
Run-time problems: The definition of IF-THEN rules may introduce run-time problems, including security threats, especially when users are not accustomed to programming.	***Defining Safe IF-THEN Rules:*** Provide users with tools that allow them to define safe IF-THEN rules that do not introduce any run-time problems.	● Problems standardization ● Offline verification ● Debugging tools

6.3.1 Context

The advent of IoT has already brought numerous benefits to society, spanning from individual-level applications to global-scale impact (Cerf & Senges, 2016). Individuals can now interact with a wide array of connected devices, such as lamps, thermostats, and various household appliances like refrigerators and ovens, which can be connected to the Internet, transforming homes into smart environments. The influence of the IoT extends beyond personal spaces to workplaces and even entire cities, where smart environments utilize connected devices to enhance residents' comfort and convenience (Cook & Das, 2005). In addition to physical devices, a wide range of online services, including social networks, news platforms, and messaging applications, are extensively utilized by nearly everyone and can be integrated in the IoT ecosystem. As of 2022, the estimated number of Internet users worldwide was 5.3 billion, with over 2.9 billion individuals actively engaged in a single social media, i.e., Facebook (Petrosyan, 2023). This widespread adoption enables users to effortlessly connect with a sophisticated network of connected entities, encompassing both smart devices and online services. These entities have the capability to communicate not only with each other but also with humans and the surrounding environment.

Given the existence of such an interconnected network and its continued expansion, researchers – especially in HCI and ubiquitous computing – have long agreed on the urgent challenge of providing users with efficient EUD methodologies and tools to customize and personalize it. According to (Paternò, Lieberman, & Wulf, 2006), EUD can be defined as *"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 artifact."* One of the pioneering works in this area is iCAP (Dey, Sohn, Streng, & Kodama, 2006), a rule-based, visual, and PC-based system designed to construct context-aware applications without users needing to engage in coding. Recently, EUD approaches and methodologies have been extensively investigated in various contexts, including mobile environments (Namoun, Daskalopoulou, Mehandjiev, & Xun, 2016), smart homes (Brich, Walch, Rietzler, Weber, & Schaub, 2017), and web mashups (Daniel & Matera, 2014). From a

commercial point of view, end users can also leverage visual programming platforms such as IFTTT[3] and Zapier[4] to customize and personalize the joint behaviors of their connected entities, e.g., to blink a lamp when someone is at the entrance door, all without the necessity of coding.

A common link between research-based and commercial tools lies in the exploited programming paradigm. Specifically, most EUD solutions in the IoT context adopt the trigger-action programming paradigm (Desolda, Ardito, & Matera, 2017). In its basic form, such a paradigm allows users to connect an event to an action to be automatically executed so that users can establish connections between a pair of devices or online services. The result is an IF-THEN (or trigger-action) rule where, upon detecting a specific event (the trigger) on one device or service, an automatic action is executed on the other device or service. Examples may include:

- if I publish a photo on *Facebook*, then upload it to my *Google Driv*e;
- if the *Nest* security camera detects a movement, then blink the kitchen's *Philips Hue* lamp;
- if the *Nest* thermostat detects that the temperature rises above 22 degrees Celsius, then open the *SmartThings* window.

In most contemporary trigger-action programming platforms, especially commercial tools like IFTTT and Zapier, users can define IF-THEN rules through wizard-based procedures (Desolda, Ardito, & Matera, 2017).

[3] https://ifttt.com/, last visited on June 10, 2023
[4] https://zapier.com/, last visited on June 10, 2023

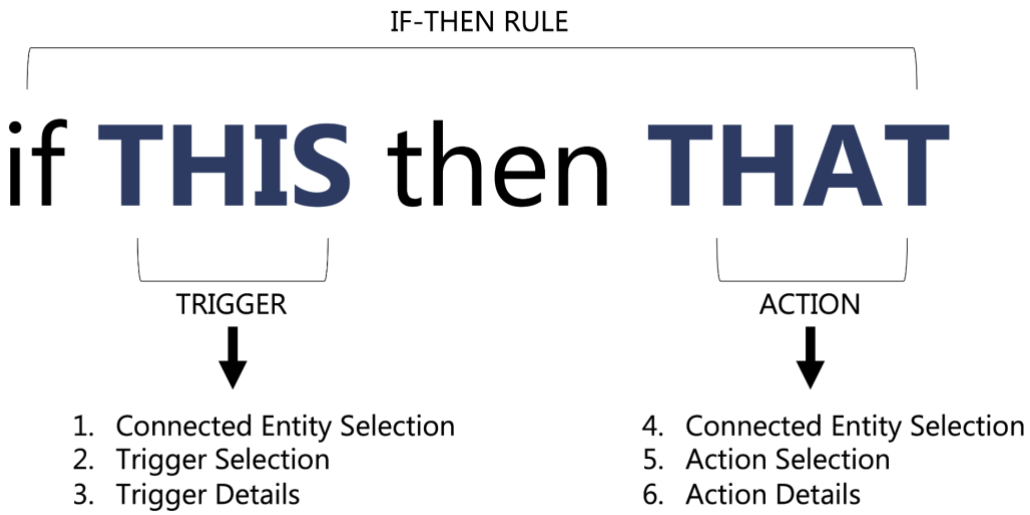

Figure 6.3.1. The process that users must follow to define IF-THEN rules in most trigger-action programming platforms, e.g., IFTTT and Zapier.

The process of defining rules is summarized in Figure 6.3.1. It involves separately defining a trigger and an action – that are then linked together. The following steps must be followed to define a trigger:

1. Choose the relevant connected entity involved in the trigger by selecting from available supported entities in a menu (Connected Entity Selection). Usually, entities are represented by their respective manufacturers or brands, such as *Nest* thermostats. Figure 6.3.2, for example, shows the Connected Entity Selection step on IFTTT and Zapier, respectively.
2. Select the specific trigger to be monitored from another menu (Trigger Selection). For example, a trigger could represent the detection of a high temperature.
3. Provide additional details for the trigger (Trigger Details), such as the identifier of the specific *Nest* thermostat or a temperature threshold value.

The three steps are then similarly repeated to define the action associated with the rule.

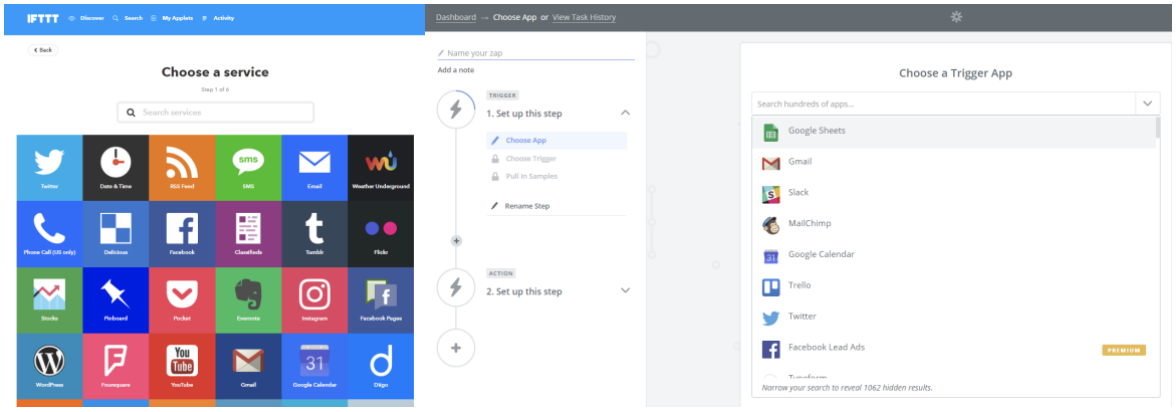

Figure 6.3.2. The selection of a connected entity to define a trigger in IFTTT and Zapier, respectively.

According to (Barricelli & Valtolina, Designing for End-User Development in the Internet of Things, 2015), trigger-action programming provides an exceptionally straightforward and easy-to-learn solution for developing end-user applications in the IoT domain. While certain behaviors may necessitate a higher level of expressiveness, such as incorporating multiple actions or additional trigger conditions, it is noteworthy that several widely used trigger-action programming platforms, including IFTTT and Zapier, still adhere to the fundamental basic structure of the trigger-action programming paradigm summarized in Figure 6.3.1 (Brackenbury, et al., 2019). One of the reasons for such a choice is that studies have highlighted the tendency for users to misinterpret the behavior of rules involving multiple triggers and actions (Huang, Azaria, & Bigham, 2016), e.g., because end users are generally unable to distinguish between states, instantaneous triggers, and conditions.

Ideally, trigger-action programming can express most of the behaviors that potential users desire (Ur, McManus, Pak Yong Ho, & Littman, 2014). However, despite its widespread adoption, researchers agree that this paradigm's current implementation introduces its own issues. As summarized in Table 6.3.1, we can identify three main issues:

- **Low-Level of Abstraction.** In the IoT context, the presence of new connected entities cannot always be predicted in advance (Zaslavsky & Jayaraman, 2015). These connected entities may appear and disappear dynamically, depending on factors such as user location (e.g., public services in a smart city). Unfortunately, current trigger-action programming platforms

rely on representation models that are heavily tied to specific technologies, thus poorly adaptable to the increasing complexity of the IoT ecosystem. These platforms typically work with specific connected entities that have been previously associated with a particular user. For example, they do not effectively handle situations where two different devices or online services offer identical functionalities but differ in terms of brands or manufacturers, but they treat them as completely distinct entities (Corno, De Russis, & Monge Roffarello, A high-level semantic approach to End-User Development in the Internet of Things, 2019). Consequently, as the number and diversity of connected entities continue to grow, the IoT ecosystem becomes increasingly complex (Barricelli & Valtolina, Designing for End-User Development in the Internet of Things, 2015), and the growing complexity of their interactions poses challenges for non-programmers in defining IF-THEN rules (Huang, Azaria, & Bigham, 2016).

- **Information Overload.** As reported in the previous issue, contemporary trigger-action programming platforms currently lack a high level of abstraction: they model each device and online service based on its underlying brand or manufacturer. This approach leads to a vast number of potential combinations of triggers and actions from various technologies. Consequently, the number of shared rules on these platforms is steadily increasing. For instance, Zapier supports thousands of devices and web applications, each with its own set of triggers and actions. Similarly, as of September 2016, IFTTT had already surpassed 200,000 publicly available and reusable rules (Ur, et al., 2016). Unfortunately, many of these platforms fail to offer users any support for discovering new combinations of triggers and actions (Ur, et al., 2016). As a result, the rapid proliferation of smart devices and online services leads to user interfaces that become overloaded with excessive amounts of information.
- **Run-Time Problems.** The last issue relates to the fact that contemporary trigger-action platforms do not provide end users with any support for understanding and debugging their

IF-THEN rules, e.g., in order to prevent potential conflicts (Caivano, Fogli, Lanzilotti, Piccinno, & Cassano, 2018) and evaluate the correctness of the rules (Desolda, Ardito, & Matera, 2017). Due to the current low level of abstraction in trigger-action programming platforms, however, users often misinterpret the intended behavior of their trigger-action rules (Brush, et al., 2011), thus deviating from the actual semantics of the rules and introducing errors (Huang & Cakmak, 2015). Unfortunately, errors in this context can have unpredictable and hazardous consequences (Brush, et al., 2011). For example, an incorrectly defined rule could unexpectedly unlock the main door of a house, resulting in a significant security threat.

6.3.2 Challenges

Overall, the three identified issues relate to distinct challenges that have inspired and continue to inspire research efforts in the field of EUD for the IoT (Table 6.3.1): supporting users in defining IF-THEN rules, promoting the discovery of IF-THEN rules and functionality, and defining safe IF-THEN rules. In this Section, we describe such challenges by highlighting the most promising solutions that have been proposed to address each of them.

Supporting IF-THEN Rules Definition. Modern trigger-action programming platforms utilize representation models that directly map onto technology, which fail to adequately accommodate the growing intricacy of the IoT ecosystem (specifically, the low-level abstraction issue reported in Section 6.3.1). For instance, platforms like IFTTT and Zapier categorize devices and services in the Connected Entity Selection step of the rule definition process solely based on their manufacturer or brand, as depicted in Figure 6.3.2. Specifically, users are constrained to use a vendor-centric approach and must configure all the detailed technical aspects necessary for executing the desired trigger-action automation. This situation presents significant challenges in terms of interoperability, as each IoT device and online service must be managed separately. For example, users cannot create a trigger-action rule that applies to all their connected lamps unless they share the same brand, nor

can they create rules for other types of devices that provide interior lighting. Consequently, formulating IF-THEN rules becomes challenging for end users without programming expertise (Huang, Azaria, & Bigham, 2016). As summarized in Table 6.3.2, researchers have tried to support IF-THEN rules definition and overcome the low-level of abstraction issue following three main directions: exploring alternative *visual metaphors* that could better support trigger-action programming, producing alternative *underlying models* to model triggers and actions with a higher level of abstraction, and adopting multiple *interaction techniques* to enable trigger-action programming.

Table 6.3.2. The three main strategies that researchers have followed to support the definition of IF-THEN rules by end users: exploring alternative visual metaphors, adopting alternative representation models, and adopting varied interaction techniques.

Strategy	**Description**	**Main References**
Visual metaphors	Exploration of alternative visual metaphors, e.g., using jigsaw puzzle pieces, that could better support trigger-action programming compared to wizard-based procedures.	(Danado & Paternò, 2014) (Desolda, Ardito, & Matera, 2017) (Corno, De Russis, & Monge Roffarello, My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor, 2019) (Akiki, Bandara, & Yu, 2017)
Underlying models	Exploration of alternative representation models, e.g., exploiting Semantic Web technologies, to model triggers and actions with a higher level of abstraction.	(Ur, McManus, Pak Yong Ho, & Littman, 2014) (Ghiani, Manca, Paternò, & Santoro, 2017) (Corno, De Russis, & Monge Roffarello, A high-level semantic approach to End-User Development in the Internet of Things, 2019) (Corno, De Russis, & Monge Roffarello, Devices, Information, and People: Abstracting the Internet of Things for End-User

		Personalization, 2021)
Interaction techniques	Exploration of alternative interaction techniques, e.g., vocal and multi-modal approaches, to specify triggers and actions if IF-THEN rules.	(Manca, Parvin, Paternò, & Santoro, 2020) (Gallo, Manca, Mattioli, Paternò, & Santoro, 2021) (Barricelli, Fogli, Iemmolo, & Locoro, 2022) (Monge Roffarello & De Russis, 2023)

Concerning the first strategy (Table 6.3.2, first row), researchers mainly explored the possibility of empowering end users to adopt block-based programming approaches (Desolda, Ardito, & Matera, 2017) instead of the form-based procedures exploited by contemporary trigger-action programming platforms (e.g., those displayed in Figure 6.3.2). An illustrative instance of such an approach can be found in Scratch (Resnick, et al., 2009), a visual programming language designed primarily for teaching coding to children. With block programming, users can drag and drop blocks of various sizes and shapes onto a workspace to connect them. In contrast to form-based approaches, block programming tools offer greater flexibility and encourage user creativity. The Jigsaw metaphor, in particular, is widely recognized as an effective approach for representing blocks. In the trigger-action programming context, for example, triggers and actions can be visualized as puzzle pieces that can be seamlessly interconnected. Due to the complementary nature of the puzzle pieces, the composition of trigger-action rules is simplified *by design*, with some wrong operations that are prevented by construction: pieces of the same type, e.g., two trigger pieces, cannot be connected, for example. Therefore, it is not surprising that such a metaphor has been found to be adequate to support the definition of IF-THEN rules. Three prominent examples of EUD tools exploiting the Jigsaw metaphor are the following:

- *Puzzle* (Danado & Paternò, 2014) is a mobile framework that allows end users without IT backgrounds to create, modify, and execute mobile trigger-action automation with the Jigsaw metaphor.

- *My IoT Puzzle* ((Corno, De Russis, & Monge Roffarello, My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor, 2019) is a web-based tool to compose and debug IF-THEN rules through the Jigsaw metaphor (Figure 6.3.3). Available triggers and actions, in particular, have been extracted from a large dataset of IF-THEN rules shared on IFTTT and extracted through a web-scraping procedure (Ur, McManus, Pak Yong Ho, & Littman, 2014).
- *ViSiT* (Akiki, Bandara, & Yu, 2017) is a method that enables non-technical users to employ a puzzle-like representation to define transformations, which are then automatically translated into executable workflows. For instance, this approach can be utilized to connect a smart refrigerator with an online shopping service, automating the process of restocking low-inventory items.

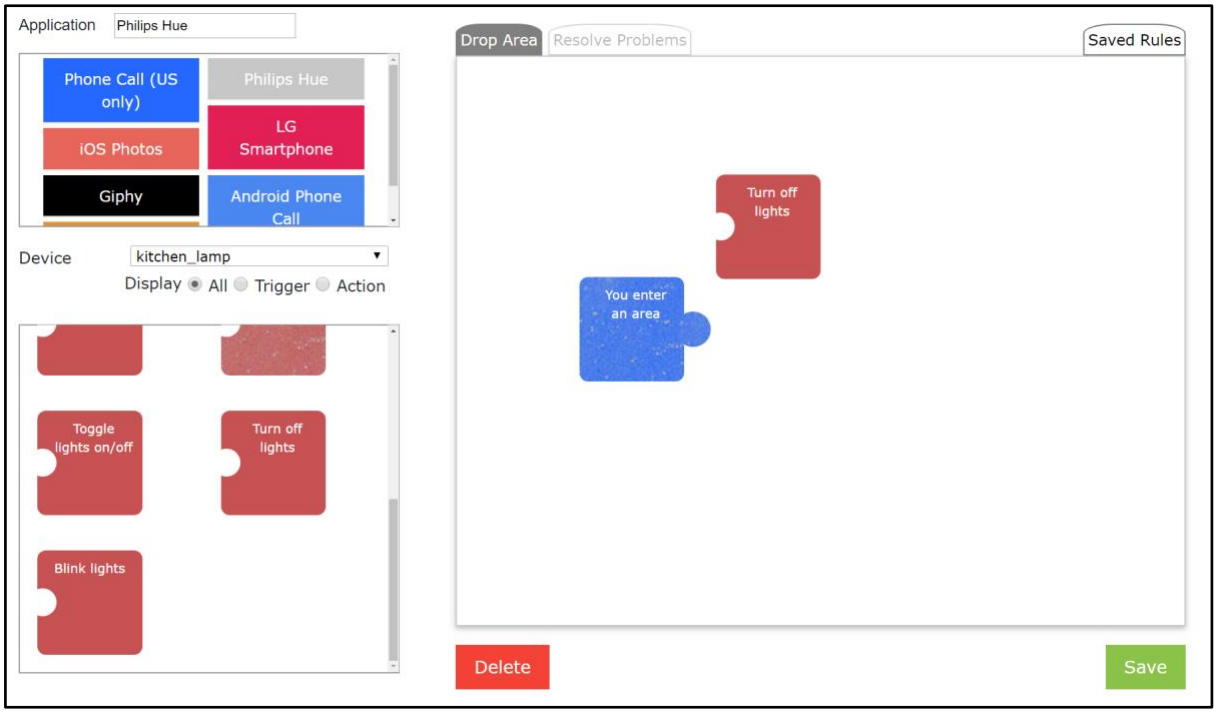

Figure 6.3.3. The My IoT Puzzle tool allows users to define the IF-THEN rules modeled by the IFTTT platform through the Jigsaw metaphor. Image taken from the original paper.

Besides focusing on the visual aspects and metaphors to represent triggers and actions, other researchers tried to support the definition of IF-THEN rules by changing the underlying representation models (Table 6.3.2, second row). Specifically, recent works explored the personalization of IoT ecosystems through the lens of *abstraction*: a direct approach to address the

low-level of abstraction concern (Section 6.3.1) is to enable users to define their triggers and actions using a higher level of abstraction, thereby minimizing the need to deal with technical details and specific attributes like brands and manufacturers, which can be abstracted away by the adopted representation model. In addition to simplifying the definition process, using a higher level of abstraction also matches end-users' mental models. In a study exploring the effectiveness of the trigger-action programming paradigm in the smart-home context, (Ur, McManus, Pak Yong Ho, & Littman, 2014) discovered that users would express triggers without mentioning specific sensors (e.g., "when the doorbell rings"), often referring to very abstract behaviors (e.g., by using "fuzzy triggers" like "when my pool chemicals drop lower than normal"). (Corno, De Russis, & Monge Roffarello, Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization, 2021) confirmed these findings beyond the smart-home context. By reporting on the results of a 1-week-long diary study during which participants were to collect trigger-action rules arising during their daily activities, the authors found that end users would adopt a higher level of abstraction compared to contemporary platforms like IFTTT and Zapier. Furthermore, the authors found that the adopted level of abstraction often depends on the context for which the personalization is envisioned: while users are typically interested in personalizing physical objects in the home, for example, they would be more interested in personalizing information in the context of a smart city.

Stemming from these findings, practical alternative representation models – allowing users to define IF-THEN rules with different levels of abstraction – have been proposed in the literature. For example, (Ghiani, Manca, Paternò, & Santoro, 2017) proposed *TARE*, an authoring tool to personalize the contextual behavior of IF-THEN rules. Specifically, an underlying architecture based on a context manager can activate, interpret, and apply abstract IF-THEN rules (e.g., "when the user falls asleep, switch off the bedroom TV") to specific contexts. Similarly, (Corno, De Russis, & Monge Roffarello, A high-level semantic approach to End-User Development in the Internet of Things, 2019) introduced *EUPont*, an ontological representation of End-User Development in the IoT. This

framework enables the creation of context-independent IF-THEN rules based on users' ultimate goals. With EUPont, users can bypass specific device commands, such as turning on a Philips Hue lamp or opening bedroom blinds, and directly instruct the system to illuminate the room, for instance.

The remaining strategy to support the definition of IF-THEN rules concerns exploring alternative interaction techniques to specify triggers and actions (Table 6.3.2, third row). Researchers, in particular, focused on empowering users to define their personalization via natural language, e.g., through chatbots or using voice-based interaction. For example, (Gallo, Manca, Mattioli, Paternò, & Santoro, 2021) proposed *Rule Bot*, a chatbot that directly maps the user's intention to the corresponding rule. After the chatbot welcomes, the user can enter a possible trigger or action through an iterative dialogue process, with the tool that can provide feedback and ask for additional information to complete the rule. (Manca, Parvin, Paternò, & Santoro, 2020), instead, introduced an integration of *Amazon Alexa* with a rule-based personalization platform that aims to facilitate the development of trigger-action rules enriched with voice-based functionality. Similarly, (Barricelli, Fogli, Iemmolo, & Locoro, 2022) proposed a new multi-modal approach to create Amazon Alexa routines, leveraging Echo Show devices. Recently, (Monge Roffarello & De Russis, 2023) designed and implemented two different prototypes of Intelligent Personal Assistants (IPAs) that allow end users to define trigger-action rules via voice. The first prototype utilizes a completely vocal interaction mechanism, enabling users to define a rule in a single sentence. In case of errors or misunderstandings, users can refine or correct the rule through subsequent dialogues. On the other hand, the second prototype combines the vocal definition of the trigger with a phase where the user is prompted to physically demonstrate the action to be automated.

Promoting IF-THEN Rules Discovery. Contemporary trigger-action programming platforms often model connected entities based on their respective brands or manufacturers, leading to a vast

number of potential combinations among triggers and actions from different technologies. Consequently, the proliferation of shared rules becomes a significant concern, contributing to the issue of information overload (Section 6.3.1). For example, there are numerous possibilities for a user who wishes to customize their smart home's behavior using IFTTT. For instance, they can specify a temperature for their *Nest* thermostat to be automatically adjusted when their *BMW* smart car approaches the home area. Alternatively, they can program the *Philips Hue* lamp in the kitchen to turn on whenever the *Arlo* security camera detects movement. Even within this limited scenario, the four mentioned connected entities provide a total of 15 triggers and 19 actions on IFTTT, resulting in 285 potential rules. Furthermore, the number of possible combinations increases significantly if we consider specific details of each trigger and action, such as the thermostat's temperature threshold or the lamp's light intensity. The challenge here is to support users in discovering useful IF-THEN rules and related functionality: compared to supporting IF-THEN rules definition (Table 6.3.1, first row's challenge), strategies to solve such a challenge (see Table 6.3.3 for a summary) do not require radical changes in the underlying models and visual metaphors but focuses on making existing personalization options "more visible."

Table 6.3.3. The two main strategies that researchers have followed to support users in discovering IF-THEN rules and related functionality.

Strategy	**Description**	**Main References**
Recommender systems	Development of recommender systems that can suggest IF-THEN rules or specific triggers and actions based on users' preferences and context.	(Srinivasan, Koehler, & Jin, 2018) (Corno, De Russis, & Monge Roffarello, RecRules: Recommending IF-THEN Rules for End-User Development, 2019) (Mattioli & Paternò, 2020)
Conversational agents	Development of conversational agents, typically in the form of chatbots, that can	(Huang, Azaria, & Bigham, 2016)

	map a natural-language request of the user into the intended IF-THEN rule(s).	(Corno, De Russis, & Monge Roffarello, From Users' Intentions to IF-THEN Rules in the Internet of Things, 2021)

The first practical strategy (Table 6.3.3, first row) concerns the development of recommender systems, a straightforward approach to solving the information overload issue in contemporary trigger-action programming platforms (Ur, et al., 2016). Using recommendation techniques can enhance the definition and reuse of trigger-action rules, thereby assisting individuals without technical or programming expertise in conveniently personalizing their smart devices and online services. Besides developing recommendation algorithms, researchers also integrated them into EUD tools for end users, either through custom trigger-action programming platforms or as extensions to existing interfaces. For example, *RuleSelector* (Srinivasan, Koehler, & Jin, 2018) implements an algorithm that selects the top action rules based on four rule selection metrics: total action coverage, confidence, interval count, and contextual specificity. *TAPrec* (Corno, De Russis, & Monge Roffarello, TAPrec: supporting the composition of trigger-action rules through dynamic recommendations, 2020), instead, is a EUD platform modeled after IFTTT specifically designed to facilitate the creation of trigger-action rules through dynamic recommendations. The tool offers suggestions during the rule composition process, including new rules to consider (Figure 6.3.4 (a)) or actions for auto-completing a rule (Figure 6.3.4 (b)).

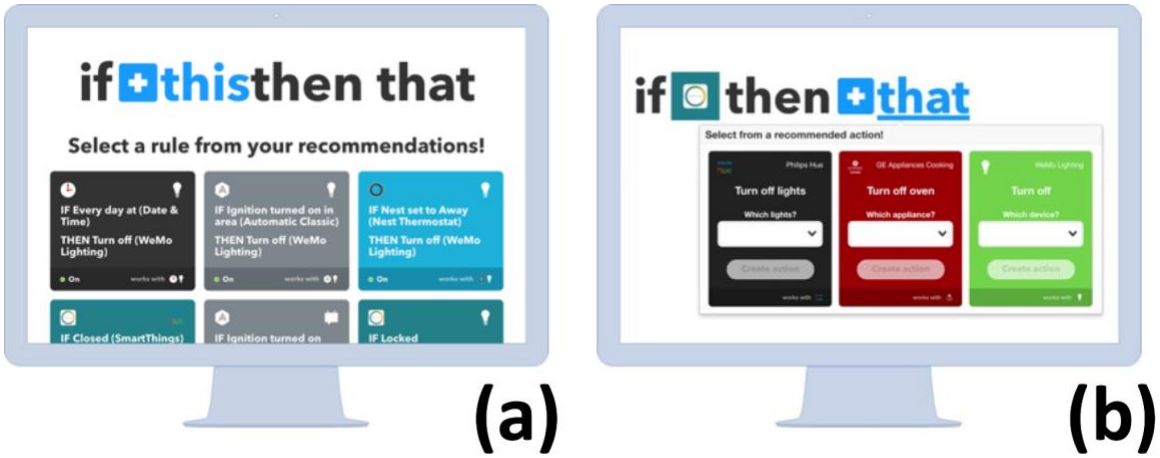

Figure 6.3.4. TAPrec (Corno et al., 2020) is a EUD platform modeled after IFTTT that integrates recommendations of IF-THEN rules (a) and recommendations of actions to auto-complete a rule (b). Images taken from the original paper.

TAPrec recommendations are computed by RecRules (Corno, De Russis, & Monge Roffarello, RecRules: Recommending IF-THEN Rules for End-User Development, 2019), a recommendation algorithm that suggests IF-THEN rules based on their ultimate objectives, bypassing specific details such as manufacturers and brands. This algorithm leverages an ontology to enhance rules with semantic information, facilitating the discovery of concealed relationships among rules in terms of shared functionality. For instance, a rule for activating a *Philips Hue* lamp shares functional similarity with a rule for opening *Hunter Douglas* blinds, as both aim to illuminate a space. Unlike *TAPrec*, which can be considered an extension of the IFTTT platform, *BlockComposer* (Mattioli & Paternò, 2020) is a recommendation-powered EUD interface with its own custom user interface. However, it adopts a similar recommendation approach. Specifically, the tool offers two recommendation policies assisting users during the definition of IF-THEN rules: i) step-by-step, through which the tool suggests the next element to be included in the rule being edited; and ii) full rule, through which complete rules are provided as suggestions.

As a second strategy to promote IF-THEN rules discovery (Table 6.3.3, second row), researchers have innovated conversational agents, typically in the form of chatbots, able to map a natural-language request of the user into the intended IF-THEN rule(s). Through this strategy, strongly linked to the interaction techniques strategy reported in Table 6.3.2 and the first strategy of Table 6.3.3, researchers take advantage of specialized recommender systems to move from an abstract intention of the user to a set of specific IF-THEN rules that can be executed in the user's environment. One of the first prototypes to compose rules via conversation and recommendations, named *InstructableCrowd*, was proposed by (Huang, Azaria, & Bigham, 2016). The tool is a crowd-sourcing system that empowers users to generate IF-THEN rules according to their specific requirements.

Through a dedicated user interface on their smartphones, end users can engage in conversations with crowd workers to articulate their issues, such as arriving late for a meeting. By utilizing a customized interface, crowd workers can effectively combine triggers and actions to address these problems and suggest proper IF-THEN rules to the end users. *HeyTAP* ((Corno, De Russis, & Monge Roffarello, HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation, 2020), instead, is a conversational platform able to map abstract users' needs to executable IF-THEN rules automatically, without involving human figures. Through a multimodal interface, users can engage with a chatbot by typing or using their voice to convey their personalization preferences for various contexts. Furthermore, users can provide supplementary details on how to implement their personalization intentions, which serve as guidance for rule suggestions. For instance, if the user selects "sustainability" as an additional intention, *HeyTAP* suggests rules considering the secondary goal of saving energy. *HeyTAP*^{2} (Corno, De Russis, & Monge Roffarello, From Users' Intentions to IF-THEN Rules in the Internet of Things, 2021) represents the evolution of *HeyTAP*, incorporating an upgraded recommender system that enhances the application's ability to comprehend user intentions through iterative refinements (Figure 6.3.5). In cases where the user cannot find a rule that aligns with their intention, *HeyTAP*^{2} implements a preference-based feedback approach. This iterative collaboration with the user allows for further feedback, leading to the refinement of recommendations. With each refinement cycle, the user can express their short-term preference by selecting a recommended rule that closely matches their intention. This information is then utilized to adjust the weighting of candidate rules, thereby promoting recommendations of items that better match the provided feedback.

Figure 6.3.5. HeyTAP[2] is a chatbot for defining IF-THEN rules that implements a preference-based feedback approach: it iteratively collaborates with the user to get a personalization intention and refine recommendations. Image taken from the original paper.

Defining Safe IF-THEN Rules. In addition to facilitating users in defining and discovering IF-THEN rules, another critical and pressing challenge must be addressed: preventing potential conflicts and evaluating the accuracy of IF-THEN rules (i.e., solving the run-time problems issue reported in Section 6.3.1). Indeed, problems and errors within trigger-action rules harm users' ability to anticipate program outcomes accurately (Brackenbury, et al., 2019). Such a mismatch, in turn, can lead to unforeseen and potentially hazardous behaviors (Brush, et al., 2011), e.g., unexpectedly unlocked doors. Regrettably, as we have seen for the previous challenges, existing trigger-action programming platforms often provide overwhelming functionality and employ technology-specific representation models that require extensive knowledge of all the smart devices and online services involved. Consequently, users frequently misinterpret the behavior of trigger-action rules deviating from their intended semantics (Brush, et al., 2011) and are prone to introducing errors (Huang & Cakmak, 2015).

Table 6.3.4 summarizes three main strategies that have been followed in the literature to define safe IF-THEN rules, i.e., rules that do not introduce problems at run-time.

Table 6.3.4. The three main strategies that researchers have followed to allow the definition of safe IF-THEN rules: problems standardization, offline verification, and debugging tools.

Strategy	**Description**	**Main References**
Problems standardization	Definition and standardization of the problems, e.g., loops and conflicts, that may arise at run-time from executing a set of concurring IF-THEN rules.	Brackenbury et al., 2019 Corno et al., 2019
Offline verification	Checking the behavior of a set of already-defined IF-THEN rules "offline," e.g., through formal verification, to identify possible run-time problems.	Liang et al., 2016 Surbatovich et al., 2017 Vannucchi et al., 2017 Zhang et al., 2019
Debugging Tools	Development of tools and platforms for end-user debugging tools to assist users in solving possible run-time problems during the definition process.	Corno et al., 2019 Manca et al., 2019

To this end, the first strategy is to define these problems and how they can influence the context in which they are introduced (Table 6.3.4, first row). Two recent works extended findings from other domains, e.g., active databases, to standardize a set of problems that apply to the context of trigger-action programming for IoT personalization. In their recent research, (Brackenbury, et al., 2019) thoroughly analyzed existing literature and bugs in other domains to identify ten programming bugs that can arise in IF-THEN rules. The study revealed several types of problems:

- *control-flow bugs*, i.e., run-time problems that may impair proper control flow, e.g., in case of conflicts between concurrent IF-THEN rules;
- *timing bugs*, i.e., problems influencing the concurrent behaviors of IF-THEN rules, e.g., the non-deterministic nature of how a system processes simultaneous triggers;
- *inaccurate user expectations*, i.e., problems that happen when novice programmers attribute system intelligence beyond their actual capabilities, leading to challenges in users' mental models.

Stemming from such an analysis, (Corno, De Russis, & Monge Roffarello, Empowering End Users in Debugging Trigger-Action Rules, 2019) specifically focused on control-flow bugs by defining three

classes of problems influencing the execution flow of IF-THEN rules. According to the authors, *loops* occur when a set of trigger-action rules are continuously activated without reaching a stable state. *Inconsistencies*, instead, occur when the execution order of concurrent rules may render different final states in the system, thus leading to inconsistent situations like a lamp that is turned on and off simultaneously. Similarly, *redundancies* are introduced when two or more concurrently activated rules have replicated functionality, e.g., a set of IF-THEN rules that share a post on a social network twice.

Previous work started to address the challenge of avoiding problems like those mentioned above at run-time leveraging software engineering techniques, e.g., formal verification (Liang, et al., 2016) and information flow control (Surbatovich, Aljuraidan, Bauer, Das, & Jia, 2017). The idea of such a strategy (Table 6.3.4, second row) is to check a set of already defined rules through an offline verification to allow experts (or the same end users) to fix potential problems afterward. (Liang, et al., 2016) developed *Salus,* a building automation service that leverages formal methods to locate faulty logic in IoT applications, transforming them into parameterized equations that can be solved through model-checking tools or Satisfiability Modulo Theories (SMT) solvers. The work by (Vannucchi, et al., 2017) demonstrates the usage of advanced SMT-based techniques for software verification in intelligent environments, optimizing the verification process and improving both the performance and expressivity of event-condition-action rules compared to previous approaches. *AutoTap*, instead, is a system introduced by (Zhang, et al., 2019) that enables novice users to define desired properties for devices and services, e.g., to specify that a window should remain closed. Then, the system translates these properties in a linear temporal logic (LTL) and is able to a) generate a set of IF-THEN rules that satisfy them and b) repair an existing set of IF-THEN rules to respect the defined properties.

Instead of checking rules "offline," i.e., after their definition, previous works highlighted the possibility of supporting users to identify programming bugs in IF-THEN rules and reason about how to fix them during the definition process. This strategy implies designing trigger-action programming

tools tailored explicitly for end users unfamiliar with programming, providing users with mechanisms to debug their IF-THEN rules (Table 6.3.4, third row). The process of debugging involves identifying and resolving the underlying cause of a misbehavior. With access to appropriate information, even end users can successfully create accurate applications and programs (Cao, et al., 2010). Following such a principle, HCI researchers have proposed and evaluated end-user debugging tools for trigger-action programming.

(Corno, De Russis, & Monge Roffarello, Empowering End Users in Debugging Trigger-Action Rules, 2019) developed *EUDebug (Figure 6.3.x)*, a tool that integrates different end-user debugging features on top of an IFTT-like interface. Specifically, the tool employs two main strategies that extend the definition process of IF-THEN rules: (i) assisting users in identifying rule conflicts by highlighting potential problems at run time (Figure 6.3.6, a) and (ii) facilitating the anticipation of rule behavior during run-time through step-by-step simulation (Figure 6.3.6, b).

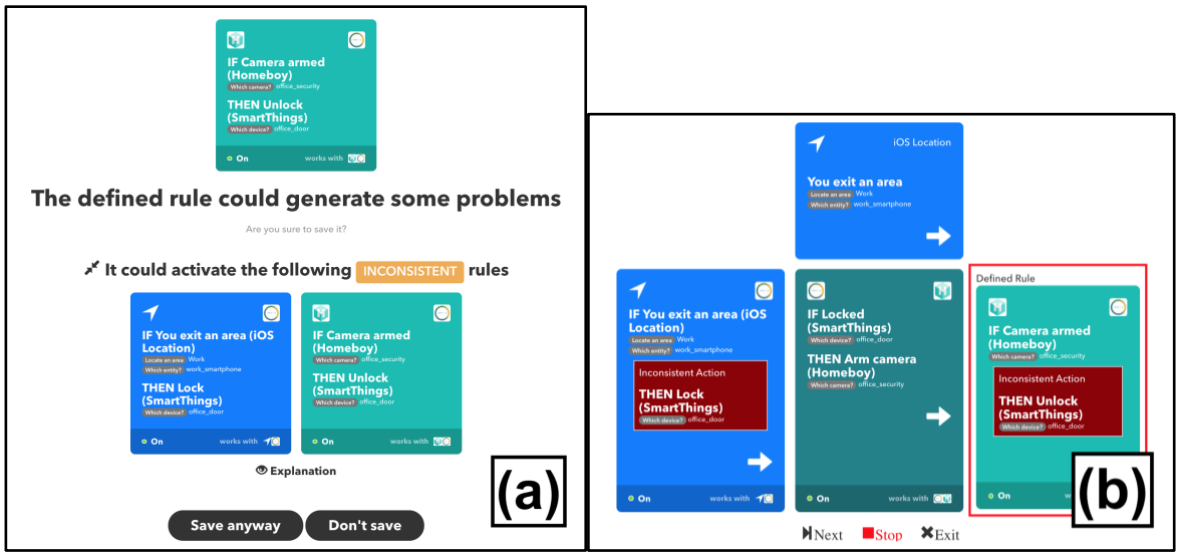

Figure 6.3.6. EUDebug integrates end-user debugging features, e.g., problem checking (a) and step-by-step explanations (b), on top of an IFTTT-like interface. Images taken from the original paper.

Under the hood, *EUDebug* exploits a Semantic Colored Petri Net (SCPN) approach to transform IF-THEN rules into a Petri network that can be executed and analyzed to find potential loops, inconsistencies, and redundancies. The same SCPN approach is also exploited by *My IoT Puzzle*

(Corno, De Russis, & Monge Roffarello, My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor, 2019). Besides adopting a novel Jigsaw visual metaphor to represent triggers and actions (see Figure 6.3.3), the tool also integrates end-user debugging features, assisting users in identifying control-flow bugs. As shown in Figure 6.3.7, in particular, puzzle pieces deteriorate over time according to their usage history. Employing the same trigger across multiple rules, for example, results in the simultaneous execution of those rules, thereby raising the probability of introducing conflicts, such as redundancies and inconsistencies. Furthermore, *My IoT Puzzle* also provides users with graphical and textual (see the "why it is not working" box in Figure 6.3.7) explanations of the detected bugs.

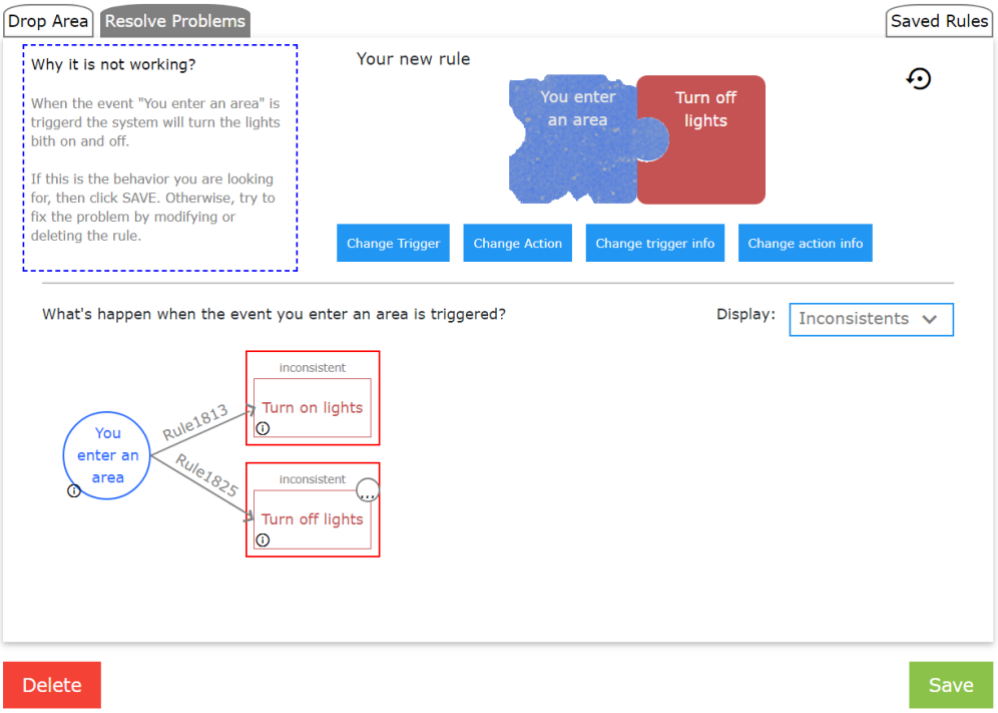

Figure 6.3.7. Besides letting users define rules through the Puzzle metaphor, My IoT Puzzle integrates end-user debugging features like puzzle pieces that deteriorate over time and textual and graphical explanations of potential run-time problems. Image taken from the original paper.

Finally, (Manca, Paternò, Santoro, & Corcella, 2019) proposed *ITAD* (Figure 6.3.8), another end-user debugging tool for trigger-action programming. The tool can identify rules that may conflict depending on the values their triggers can assume at run-time. Furthermore, it can indicate why or why not a rule can be triggered in a given context of use. As in *My IoT Puzzle*, also *ITAD* adopts

textual and graphical explanations. In Figure 6.3.8, for example, specific icons are used to differentiate instantaneous triggers and conditions visually.

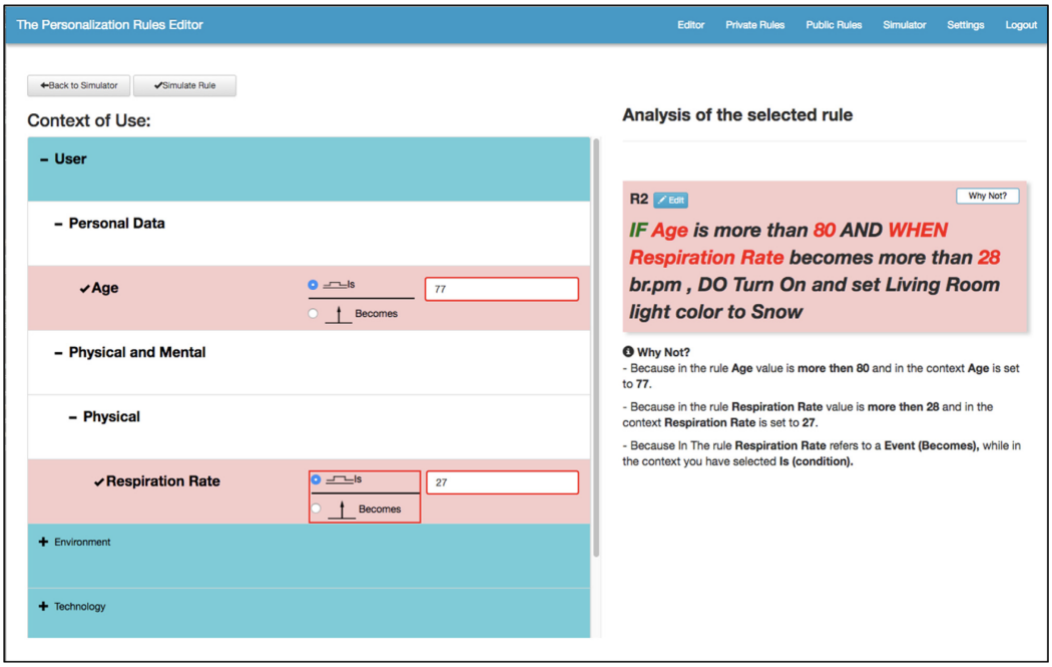

Figure 6.3.8. ITAD is an end-user debugging tool for trigger-action programming that can provide indications about why or why not a rule can be triggered in a given context of use (image taken from the original paper).

6.3.3 Discussion and Guidance for Future Research

Throughout Section 6.3, we reviewed issues, challenges, and solutions characterizing the domain of trigger-action programming for IoT personalization. Here, we discuss the most promising findings, trying to connect solutions to different challenges and highlighting opportunities for future research in such an evolving research field.

To advance the field of IoT personalization, we see particular value in further exploring the adoption of alternative representation models and interaction techniques (Table 6.3.2) for two main reasons. First, these strategies offer unique opportunities for improving the user experience in trigger-action programming platforms. By investigating and implementing alternative representation models, we can enhance the expressiveness and flexibility of trigger-action programming, enabling users to define more complex rules and personalize their IoT devices in a more nuanced manner.

Additionally, exploring novel interaction techniques can provide more intuitive and efficient ways for users to interact with and define their rules, further improving the user experience. Finally, the same strategies have not yet been fully explored – thus leaving ample room for further investigation and experimentation – and could be easily integrated with most of the other solutions targeting other challenges. Adopting representation models that support a higher level of abstraction, for example, inevitably poses trust, security, and privacy issues that may be worth exploring in future works. Furthermore, the work by (Corno, De Russis, & Monge Roffarello, A high-level semantic approach to End-User Development in the Internet of Things, 2019) suggests that abstract IF-THEN rules, e.g., those that do not refer to specific devices or manufacturers, are not always appreciated by end users: in some specific contexts, e.g., the smart home, people would prefer having more control to specify triggers and actions, e.g., to refer to specific technologies (Corno, De Russis, & Monge Roffarello, Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization, 2021). Consequently, it becomes fundamental to explore solutions that may provide end users with the right level of abstraction, e.g., through automatic AI-powered solutions or through multi-level interfaces exposing a hierarchy of possible triggers and actions at different abstraction levels. For what concerns the exploration of novel interaction techniques, instead, one of the objectives of using voice-based interaction for defining IF-THEN rules (Table 6.3.2, third row) is to convert a user's voice-based natural language request into the intended rule. While extensions of existing platforms like Amazon Alexa (Manca, Parvin, Paternò, & Santoro, 2020) or custom Intelligent Personal Assistants (Monge Roffarello & De Russis, 2023) may facilitate the vocal creation of flexible automation using the trigger-action format, the process of defining IF-THEN rules through voice input can be challenging due to the inherent ambiguities in natural language. Therefore, we emphasize the significance of advancements in Natural Language Processing (NLP) to address and eliminate potential ambiguities, enabling users to precisely indicate their desired effects. Previous studies have demonstrated the effectiveness of Artificial Intelligence methods in accurately mapping the user's abstract requirements to a lower level of abstraction that can be comprehended and

executed in real-time (see (Corno, De Russis, & Monge Roffarello, From Users' Intentions to IF-THEN Rules in the Internet of Things, 2021) and (Corno, De Russis, & Monge Roffarello, HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation, 2020)). Specifically, we see value in integrating vocal interaction techniques with recommendation algorithms that may assist users in the definition process (e.g., through the strategies described in Table 6.3.3). Many other opportunities need to be further explored to fully take advantage of vocal interaction in such a EUD domain. One of these opportunities is to understand how we can manage collections of existing rules in a voice-based system, e.g., by exploring how a voice assistant can effectively present the list of available rules without the help of a screen. While reading out all the rules with their comprehensive details may be impractical, offering only partial information (e.g., the title of a rule) could lead to errors due to limited context. Striking a balance between these extremes is the key challenge in this context. Another opportunity is to use vocal interaction to support users during the execution of rules, e.g., through proactive voice assistants that explain why a rule is being activated or why a conflict is happening. Such an approach may further complement the need to assist end users in defining safe IF-THEN rules and provide them with effective end-user debugging tools (Table 6.3.4). The challenge here is, again, at the presentation level, with open questions that concern determining when it is appropriate to alert users about a problem, to which user(s), and how to effectively communicate the conflicts.

6.4 The developer perspective

While the previous section focused on end users, their understanding of the IoT environments, and the related personalization options, this section focuses on actual programmers creating IoT systems and how developers characterize them.

Implementing IoT systems is particularly complex and differs from developing mainstream mobile and web applications. In fact, it involves a comprehensive spectrum of technologies **(Taivalsaari &**

Mikkonen, On the development of IoT systems, 2018) that involve multiple development and execution environments. Consequently, besides focusing exclusively on the code, IoT developers must also deal with hardware implementation and distributed computing concepts.

In this scenario, harnessing the programmable world's power requires understanding the peculiarities and the most challenging issues that the implementation of IoT systems poses to the developers and envisioning new software engineering and development technologies, processes, methodologies, and tools **(Taivalsaari & Mikkonen, A Roadmap to the Programmable World: Software Challenges in the IoT Era, 2017)**.

In addition to introducing IoT implications from the developers' perspective, we explore its associated issues and challenges, along with the emerging solutions. For a comprehensive overview of these issues, challenges, and solutions, please refer to Table 6.4.1

Table 6.4.1. A summary of the issues, challenges, and solutions reviewed in this Section.

Issue	**Challenge**	**Solutions**
Limited awareness of IoT peculiarities: An understanding of the methodology and its main challenges is lacking to support the entire IoT application development life-cycle.	***Characterizing IoT systems and identifying development issues faced by developers:*** Provide a common understanding of IoT defining characteristics.	● Reference architecture ● Development process analysis ● Custom Software Engineering development methodologies
Integration issues: Envisioning and successfully achieving the integration of IoT components is conceptually and technically challenging.	***Supporting developers in integrating IoT subsystems:*** Provide developers conceptual and technical resources to design and implement the integration of heterogeneous components**.**	● Programming frameworks ● Architectural styles ● Software development approaches

6.4.1 Context

The complexity of developing IoT systems arises from the specific computing resources, technologies, and communication protocols associated with each of the four architectural elements, i.e., devices, gateways, cloud services, and applications (see Figure 6.2.1). The coexistence of diverse

devices, protocols, architectures, and programming languages necessitates expertise in multiple disparate areas and the ability to orchestrate them effectively. As outlined in Table 6.4.1, two issues arise from this particular scenario:

- ***Limited awareness of IoT peculiarities:*** As previously outlined, the development process of IoT systems differs from traditional software and poses distinctive challenges to developers. Indeed, IoT developers are now required to consider several dimensions unfamiliar to most software developers (Taivalsaari & Mikkonen, A Roadmap to the Programmable World: Software Challenges in the IoT Era, 2017). However, there is a lack of awareness of IoT systems' specificities regarding their architecture and development process, resulting in highly difficult-to-implement and platform-dependent software. Little research to date uncovers the core development process lifecycle needed for IoT systems, and thus software engineers find themselves unprepared and unfamiliar with this new genre of system development (Fahmideh, Ahmad, Behnaz, Grundy, & Susilo, 2022). Furthermore, as stated by (Larrucea, Combelles, Favaro, & Teneja, 2017), linked to IoT systems development's inherent complexity is that no consolidated set of software engineering best practices for the IoT has emerged yet. In his words, "*IoT landscape resembles the Wild West, with programmers putting together IoT systems in ad hoc fashion."* To effectively leverage the power of the promised programmable world of IoT, it is essential to have a precise understanding of IoT intricacies. Without such understanding, it becomes impossible to envision appropriate software engineering and development technologies, processes, methodologies, and tools tailored for IoT applications.
- ***Integration issues***: From the developers' perspective, the high heterogeneity among software and hardware components present in IoT systems results in several integration challenges. This integration must ensure data exchange across diverse IoT components regarding protocols, connectivity, data formats, technologies, specifications, and communication mechanisms. Indeed, IoT systems development typically requires integrating

complex legacy systems, often conceived, developed, and deployed as monoliths (Fortino, Savaglio, Spezzano, & Zhou, 2021). Therefore, successfully achieving interoperability among these legacy systems demands careful design decisions and adopting precise technical resources. For this reason, available alternatives to deal with integration are required to consider conceptual and architectural approaches, commonly adapted from traditional software development paradigms to specific methodologies, frameworks, platforms, and tools.

6.4.2 Challenges

In general, the two identified issues correspond to specific challenges:

- Characterizing IoT systems and identifying development issues faced by developers.
- Overcoming the integration issues among heterogeneous subsystems.

In this section, we describe these challenges highlighting solutions that have been put forth to tackle each.

Characterizing IoT systems and identifying development issues faced by developers:

So far, IoT applications have been based on fragmented software implementations for specific systems and use cases (Weyrich, 2016). A unifying approach grounded on standard abstractions, models, and methodologies is still missing to aid IoT systems development (Zambonelli, 2017). Given this situation, as shown in Table 6.3.4, researchers have followed three main strategies to consolidate a common understanding of IoT systems' characteristics and development process: Reference architectures, IoT development process analysis, and custom Software Engineering development methodologies.

Table 6.3.4. Researchers have followed three main strategies to consolidate a common understanding of IoT systems' characteristics and development process.

Strategy	**Description**	**Main References**
Reference Architecture	Identification and representation of the	(Weyrich, 2016)

	main components present in a typical IoT system and the interactions among them.	(Fortino, Savaglio, Spezzano, & Zhou, 2021) (Zambonelli, 2017)
Development process analysis	Analyze quantitatively and qualitatively the characteristics of the code and the IoT developers' implementation process.	((Fahmideh, Ahmad, Behnaz, Grundy, & Susilo, 2022) (Corno, De Russis, & Sáenz, How is Open Source Software Development Different in Popular IoT Projects?, 2020) (Corno, De Russis, & Sáenz, On the challenges novice programmers experience in developing IoT systems: A Survey, 2019) (Makhshari & Mesbah, 2021)
Custom Software Engineering development methodologies	Revisit traditional software development methodologies to accommodate the distinctive needs imposed by IoT systems.	(Motta, de Oliveira, & Travassos, 2023) (Morin, Harrand, & Fleurey, 2017) (Patel, 2015) (Berrouyne, Adda, & Mottu, 2022) (Ferreira, et al., 2022)

Regarding the first strategy, the industry's need for reference architectures has become tangible with the fast-growing number of initiatives working toward standardized architectures (Weyrich, 2016). (Čolaković & Hadžialić, 2018) hold that IoT software architectures and frameworks are necessary to overcome IoT systems' inherent complexity and provide an environment for services composition. In this sense, two major architectures are available: the Internet of Things - Architecture (IoT-A) and Industrial Internet Reference Architecture (IIRA). IoT-A delivered a detailed architecture and model from the functional and information perspectives. In contrast, IIRA was delivered by the Industrial Internet Consortium (founded by AT&T, Cisco, General Electric, IBM, and Intel) for broad consideration and discussion. Such architectures can serve as an overall and generic guideline, and not all domain applications will require every component for real-life development.

Although these reference architectures may vary and a definitive “standard” architecture has not yet emerged, they encompass many functions, information structures, and mechanisms. They offer developers a more comprehensive perspective of the IoT system they intend to implement. Furthermore, they contribute to defining and elucidating the overall structure of the IoT. These reference architectures offer descriptive models that illustrate how IoT devices and humans interact and process data, incorporating patterns of machine-to-machine (M2M) communication standards (Weyrich, 2016). According to (Fortino, Savaglio, Spezzano, & Zhou, 2021), exploiting these established reference architectures is another common trait of IoT methodologies toward IoT system interoperability. Indeed, over the years, a broad and solid consensus on architectural IoT building blocks and their main design patterns has been reached.

Similarly, (Zambonelli, 2017) frames key general characteristics related to the engineering of IoT systems by synthesizing the common features of existing proposals and scenarios. The author identifies the key software engineering abstractions around which IoT systems and application development could revolve. Specifically, the common features were the things (physical objects, places, and persons), the software infrastructures, and services and applications. Meanwhile, the key abstractions for developing IoT systems were: the stakeholders and users, requirements (policies, goals, and functions), avatars and coalitions (individual things and a group of things that contribute to defining a unique functionality or service), and smart things.

As for the second strategy, research efforts have been dedicated to providing insights into the most critical development tasks and the challenges associated with implementing IoT systems. (Fahmideh, Ahmad, Behnaz, Grundy, & Susilo, 2022) conducted a research study that employed a mixed quantitative and qualitative approach. The objective was to derive a conceptual process framework for the development processes of IoT systems, wherein common tasks are organized into three distinct phases. The derived framework was validated through a survey involving 127 IoT practitioners. These participants provided justifications, examples, and recommendations on tasks' importance and associated challenges. The authors identified challenges in the analysis, design, and

implementation phases. Additionally, respondents offered general advice to software teams on effectively addressing these challenges.

Similarly, (Corno, De Russis, & Sáenz, On the challenges novice programmers experience in developing IoT systems: A Survey, 2019) surveyed 40 novice developers that worked in groups developing IoT systems during several years of a university course. Based on their own experiences, individually and as a group, the most challenging development tasks were identified and prioritized over a common architecture regarding difficulty level and effort. In line with this research, to gain a deeper understanding of how open-source IoT projects differ from non-IoT projects, (Corno, De Russis, & Sáenz, How is Open Source Software Development Different in Popular IoT Projects?, 2020) conducted a quantitative analysis of a broad set of the 60 most popular publicly available IoT and non-IoT projects on GitHub. The analysis provided insight into the purpose and characteristics of the code, the behavior of the contributors, and the maturity of the IoT software development ecosystem. The findings highlight significant differences between IoT and non-IoT application development regarding how applications are realized, the diversity of developers' specializations, and how code is reused.

Finally, regarding the third strategy, the lack of a software engineering methodology to support the entire IoT application development lifecycle results in highly difficult to maintain, reuse, and platform-dependent design (Patel, 2015). In this sense, (Motta, de Oliveira, & Travassos, 2023) present an evidence-based roadmap for IoT development to support developers in specifying, designing, and implementing IoT systems. The IoT Roadmap consists of 117 items grouped into 29 categories. An experimental study on a case study involving a healthcare IoT project was conducted. Moreover, the roadmap includes a convenient checklist that aids in identifying the most relevant recommendations for developing and engineering IoT software systems.

Likewise, according to (Morin, Harrand, & Fleurey, 2017), IoT applications have two main characteristics from a software engineering viewpoint: their distribution over an extensive range of processing nodes; and the high heterogeneity of the processing nodes and the protocols used. To

deal with these characteristics, they introduce a modeling language aligned with UML, an advanced multiplatform code generation framework, and a methodology specifying the development processes and tools used by IoT service developers and platform experts. Similarly, (Patel, 2015) introduce a development methodology for IoT application development based on model-driven development. Such methodology separates IoT application development into different concerns and provides a conceptual framework to develop an application. Additionally, the framework provides a set of modeling languages to specify each development concern.

Achieving the integration among heterogeneous subsystems:

As previously described, the diversity among IoT software and hardware components raises several interoperability issues. Specifically, in the IoT scenario, interoperability refers to the degree to which two or more components can exchange information and use the information that has been exchanged (Noura, Atiquzzaman, & Gaedke, 2019). Successfully achieving this interoperability impacts the widespread adoption of IoT technology on a large scale.

From the development point of view, alternatives to deal with the integration issues and achieve interoperability. Given the complexity and heterogeneity of the IoT ecosystem, these strategies go beyond technical aspects and involve Methodologies and architectural approaches. These strategies typically involve architectural approaches like microservices, event-driven architectures, and service-oriented architectures to IoT platforms and tools that connect different devices and access, manage, and process their data. Table 6.3.4 mentions some strategies to deal with the complexity of integrating IoT components from two perspectives: Methodologies and architectural approaches and Programming frameworks, platforms, and tools.

Table 6.3.4. Researchers have followed three main strategies to deal with the complexity that integrating IoT components poses.

Strategy	**Description**	**Main References**
Architectural approaches and methodologies	Architectural approaches adapted from traditional software development. Service Oriented Architectures, and Model Driven	(Asghari, 2018) (Urbieta, 2017) (Hamzei & Navimipour,

	Development are among the most prominent alternatives.	2018) (Cabrera, Cárdenas, Cedillo, & Pesántez-Cabrera, 2020) (Patel, 2015)
Programming frameworks, platforms and tools	IoT platforms, tools and programming frameworks aimed at easing the interoperability among IoT components. IoT Cloud Platforms, Middlewares, Software adapters, and Gateways are the most popular strategies.	(Avilés-López & García-Macías, 2012) (Barros, et al., 2022) (Fortino, Savaglio, Spezzano, & Zhou, 2021) (Razzaque, Milojevic-Jevric, Palade, & Clarke, 2016) (Beniwal & Singhrova, 2022)

At the **architectural approach**, software adapters, gateways, and service-oriented interfaces are standard solutions to ensure syntactic interoperability among diverse communication technologies or application layer protocols. In this regard, the most prevalent approaches employed by IoT frameworks to develop interoperable IoT systems are *Model-Driven Engineering* (MDE) and service-oriented development (Fortino, Savaglio, Spezzano, & Zhou, 2021). MDE relies on domain knowledge modeling standards and metalanguages to describe IoT system components and functionalities independently of their specific implementation details. For their part, *Service Oriented Architectures* (SOA) consider the definition of well-defined interfaces and protocols that allow IoT devices and applications to communicate and exchange data regardless of the underlying technologies or platforms. In this sense, exposing each component's functionalities as a standard service can significantly increase network and device interoperability (Noura, Atiquzzaman, & Gaedke, 2019). In particular, resource-oriented web services (REST web services) are widely adopted to achieve the potential of SOA, providing service sharing and reuse, and have been used to address syntactic interoperability.

Along the same line, *microservices* are another way to support protocol-aware heterogeneous interoperability. They allow loose-coupled service interfaces and integration, simplifying services' architectural patterns and related implementations (Fortino, Savaglio, Spezzano, & Zhou, 2021). At

the *software* layer, APIs are commonly utilized in SOA and microservices approaches to achieve syntactic interoperability.

Concerning the currently proposed **IoT methodologies**, (Fortino, et al., 2018) presented a full-fledged, general-purpose methodology supporting the integration process among IoT platforms. This methodology is based on a process with analysis, design, implementation, deployment, and maintenance phases. Similarly, (Geraldi, Reinehr, & Malucelli, 2020) provide a comprehensive overview of how the integration challenges can be faced by relying on Software Product Line methodologies tailored for the IoT. (Cabrera, Cárdenas, Cedillo, & Pesántez-Cabrera, 2020), for their part, propose an agile methodology to guide the development of IoT software components based on microservice architectures. Finally, (Patel, 2015) presents a methodology separating IoT application development into different concerns and providing a conceptual framework to develop an application.

From a technical perspective, *IoT middleware*, following SOA architectural approach, provides services and abstractions to simplify **IoT system development**. At the same time, IoT gateways focus more on easing connectivity, data processing, and protocol translation. By definition, middleware serves to abstract the complexities of the system or hardware. It enables the developer to concentrate solely on the business logic of the solution that is being implemented. Middleware in IoT eases development by integrating heterogeneous computing and communications devices and supporting interoperability within diverse applications and services (Razzaque, Milojevic-Jevric, Palade, & Clarke, 2016).

Whether software or hardware components, *IoT Gateways* are bridges between IoT devices and the cloud, data centers, or central management systems connected through the Internet (Beniwal & Singhrova, 2022), they facilitate protocol conversion between the sending and receiving devices. They use plug-and-play software adapters to connect components and platforms with communication technologies, application layer protocols, or data formats (Noura, Atiquzzaman, & Gaedke, 2019). These gateways can communicate using protocols like Bluetooth, ZigBee, and

Ethernet and interfaces like Wi-Fi, MQTT, and CoAP (Dizdarević, Carpio, Jukan, & Masip-Bruin, 2019). At a platform level, currently, *IoT Cloud platforms* provide tools and protocols for device registration, authentication, and secure communication. They offer various services like device management, data storage, analytics, and application development tools. In particular, they provide a scalable and secure infrastructure for connecting, monitoring, and controlling a wide range of IoT devices and applications. Furthermore, Today's IoT platforms almost all provide a public API to assist developers in connecting with other systems and services and accessing their services. Popular IoT Cloud platforms include Amazon Web Services (AWS) IoT Core, Microsoft Azure IoT Hub, Google Cloud IoT Core, and IBM Watson IoT. These platforms offer comprehensive features and services to support the entire IoT ecosystem, from device connectivity to data analytics.

6.4.3 Discussion and Guidance for Future Research

We want to draw attention to the fact that as the IoT continues to position itself as a well-established paradigm, the ecosystem grows, expands to several domains, and research efforts are sufficiently addressing the emerging challenges. There are plenty of academic and industry efforts to identify and address the requirements that IoT development poses. Therefore, the upcoming issues do not arise from the shortcoming of methodologies, frameworks, platforms, and tools. On the contrary, they paradoxically emerge from the overabundance of resources, continuously fueled by innovative and evolutionary development approaches and technologies (Fortino, Savaglio, Spezzano, & Zhou, 2021). In this scenario, choosing the best solution for designing and implementing an IoT system has become a challenging entry barrier, especially for novice IoT developers. Indeed, preparing future IoT developers has become particularly important given the definite IoT consolidation and expansion in several domains and its ever-growing ecosystem complexity (Corno & De Russis, Training Engineers for the Ambient Intelligence Challenge, 2017).

In this scenario, documentation resources play a fundamental role. Specifically in terms of providing comprehensive documentation that accounts for the IoT development peculiarities. For instance,

previous research has determined that it is hard for programmers to understand low-quality documentation of certain device manufacturers and interpret complex response payloads from particular devices (Makhshari & Mesbah, 2021). In the context of novice programmers getting started with IoT development, (Corno, De Russis, & Sáenz, On the challenges novice programmers experience in developing IoT systems: A Survey, 2019) identified that integration among IoT subsystems was one of the most painful issues not-experienced developers faced due to the lack of appropriate documentation. Therefore, they proposed a documentation mechanism called Code Recipes, aimed at enabling the integration of several software components through code fragments that might belong to different programming languages and might be deployed across various runtime environments, as it is common in IoT systems. Code Recipes (Corno, De Russis, & Saenz Moreno, Easing IoT development for novice programmers through code recipes, 2018), therefore, are defined as summarized and well-defined documentation modules independent from programming languages or runtime environments. Precisely, these recipes are specified through metadata and consist of multiple code fragments, documentation, and links to ease the understanding of such code to implement a given integration between subsystems of an IoT system. As a further step in this research direction, and given the prominence that computational notebooks have been gaining due to their capability to consolidate text, executable code, and visualizations, (Corno, De Russis, & Saenz Moreno, Computational notebooks to support developers in prototyping IoT systems, 2022) have explored to what degree these notebooks are appropriate for facilitating the prototyping of IoT systems, even when the process entails multiple steps and encompasses various development and execution environments. The authors designed, implemented, and assessed an IoT-tailored notebook aimed at helping developers to build and share a computational narrative around the prototyping of IoT systems. The rationale behind the proposal stemmed from the belief that by documenting the reasoning and subsequent steps in a Computational notebook, within a meticulously configured execution environment, the prototyping of IoT systems can be enhanced in terms of reproducibility and comprehensibility.

In a broader sense, several research efforts have been devoted to envisioning IoT learning strategies. (Burd, et al., 2018) have highlighted the challenges for computer science educators in conducting IoT courses: connecting and integrating hardware and software; finding adequate physical space and infrastructure; and preparing instructors and teaching assistants for the content.

Furthermore, the authors identify core and specialized topics that educators might integrate into their curricula depending on the student's existing knowledge and learning objectives. Among these identified topics are: cloud computing, human-computer interaction, embedded computing, platform-specific development, and web and mobile development, among others. Naturally, these IoT curriculums must continuously evolve as the IoT industry evolves and new tools and components appear in the ecosystem.

Due to the above, a significant challenge that, in our opinion, requires attention is envisioning strategies and tools to enclose into a typical academic period the topics and skills that enable students to face a generic IoT system implementation. This goal requires finding a proper balance in the hardware platforms, networking protocols, programming languages, and cloud services to include. Similarly, to better cover these topics and technologies and to set up a realistic development scenario, it is imperative to promote teamwork and project-based learning approaches.

6.5 Bridging the gap between perspectives

In the IoT, end users and developers have different needs and perspectives. They are, however, two faces of the same coin: users can personalize their IoT environments if developers provide them with proper tools and options; developers can work better if they know what the users' expectations and needs are. While end users focus on personalizing the behavior of their environments through dedicated tools, developers are more concerned with integration issues due to the variety of architectural elements that constitute an IoT system.

In addition, both have their understanding of IoT environments that stems from their specific

context. End users want to create automations to increase their serendipity and extend the capabilities of the existing system, within their understanding of how the IoT-enabled environment operates. Conversely, developers want to create robust systems, extendable and with suitable user interfaces, within their understanding of what can constitute an IoT environment. The user interface is one of the many components they have to consider, and often not the most critical to make the system work.

They also have different struggles. End users need to understand how to reach their automation goal, the implications of their actions, and what to do when things are not working as expected. Developers need to handle the various components, written in different languages and using many technologies, and integrate them in a coherent system.

From this short summary, it should be clear what is the gap, what are the motivations behind it, and what is the commonality between the two perspectives. The commonality rises from the need of both actors to understand the IoT: they need to know how the IoT environment might work. However, the different purpose in this understanding is what generates the gap: end users focus on the results derived from the run of the IoT system, while developers are still focusing on how to make an heterogeneous system work in a homogenous way, while keeping the pace with the latest IoT technologies and software methodologies. In other words, end users have a more focused and utilitarian view of the system, which is seen as a way to fulfill some intentions, while developers must have a wider perspective, considering the operational aspects and the use cases as well as the maintainability, security, and integration challenges.

To our knowledge, this gap has not yet been tackled in the literature. The research field focusing on end-user development is informed by software engineering methods and applies development concepts to their research output, still within the specific goal of creating, maintaining and debugging automations. The software engineering field, instead, considers users and interfaces for

them as one of the many elements of an IoT system, often considering the immediate use more than end-users development options.

Hereafter, the chapter proposes two possible recommendations to bridge the perspectives and the respective needs. Such recommendations stem from the literature discussed above and from the authors' personal experience in creating EUD tools and supporting IoT developers. Both asks for a stronger collaboration between the research communities focusing on end-user development and software engineering for the IoT:

IoT developers need to consider personalization options for end users: Developing an IoT system focuses on various components, one of which is the user interface within the "application" component (Figure 6.2.1). Current research is mostly focused on technical aspects, where one of the main challenges is about the integration of components into a coherent system. However, end users have varied needs and benefit from the possibility to tailor their environment accordingly. An explicit focus on end-user development, within the "application" or a dedicated component (e.g., a "personalization component"), can have two advantages for end users and developers: end users can have more options, developers can receive dedicated feedback and collect information on the actual use of their IoT system in real settings. This feedback can guide developers to not only refine the offered personalization options, but also to understand the specific needs of their users and direct the development process accordingly.

EUD tools and methodologies should be applied to IoT development: Since integration is one of the main challenges in creating IoT systems, a viable option to minimize the struggle can be to adopt EUD methodologies to ease the integration efforts, in addition to personalize the behavior of the various "things" in the environment. Such techniques can, similarly to what conversational agents and recommenders are doing in the EUD field, help with the integration and creation of various

components and documentation, thus lowering the entry barrier. The goal, here, can be to support developers so that they can personalize their IoT *development* environment with the same ease they can customize the IoT-enabled environment. In this scenario, technologies like *digital twins* can provide further support to developers, carefully replicating physical devices in their virtual clones.

6.6 Conclusion

The chapter discussed how people characterize and interact with the Internet of Things and, particularly, with IoT-enabled environments. It tackles two perspectives: end users wanting to personalize their environments according to their specific and contextual desires; developers who need to build such systems. Indeed, developers' challenges and adopted solutions can be useful to understand the options they offer to end users; conversely, end users' expectations for personalizing IoT systems can inform developers' practices. The chapter describes these two perspectives and approaches to the IoT with respect to the literature. For each of them, challenges and possible solutions are reported, as well as gaps between the perspectives. In the end, the chapter proposes two possible recommendations to try to close the gap between these two worlds with the aim to ease IoT system creation and the subsequent interaction with them.

6.6.1 Future Trends

As discussed by Sharma et al. (Sharma, Shamkuwar, & Singh , 2018), a number of interesting possibilities and future trends characterize the topic of interacting with the IoT. The first promising and needed direction involves developing energy-efficient IoT technology. Examples include the "Green IoT" discussed by Alsharif et al. (Alsharif, Jahid, Kelechi, & Kannadasan, 2023), as well as the sustainable IoT ecosystems envisioned for the agricultural (Dhanaraju, Chenniappan , Ramalingam, Pazhanivelan, & Kaliaperumal, 2022) and railway transportation (Singh, Elmi, Meriga, Pasha, & Dulebenets, 2022) domains. Other future trends worth to be explored include integrating blockchain technologies in smart cities (Majeed, et al., 2021), supporting students in learning about emerging

technologies (Van Mechelen & Smith, 2023), offering appropriate resources for End-User Development in the workplace (Barricelli, Fogli, & Locoro, 2023), and addressing security challenges (Yugha & Chithra, 2020). In conclusion, fostering and improving the interaction between end users, developers, and the IoT, has an important role in making these future trends possible and driving digital transformation, enhancing efficiency and fostering sustainability.

References

Paternò, F., Lieberman, H., & Wulf, V. (2006). End-User Development: An Emerging Paradigm. In *End User Development* (pp. 1-8). Springer Netherlands.

Cerf, V., & Senges, M. (2016). Taking the Internet to the Next Physical Level. *Computer, 49*(2), 80-86.

Cook, D., & Das, S. K. (2005). *Smart environments.* (D. Cook, & S. K. Das, Eds.) Wiley.

Petrosyan, A. (2023, February 23). *Number of internet users worldwide 2022.* Retrieved May 12, 2023, from Statista: https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/

Dey, A. K., Sohn, T., Streng, S., & Kodama, J. (2006). iCAP: Interactive Prototyping of Context-aware Applications. *Proceedings of the 4th International Conference on Pervasive Computing*, 254-271.

Namoun, A., Daskalopoulou, A., Mehandjiev, N., & Xun, Z. (2016). Exploring Mobile End User Development: Existing Use and Design Factors. *IEEE Transactions on Software Engineering*, 960–976.

Brich, J., Walch, M., Rietzler, M., Weber, M., & Schaub, F. (2017). Exploring End User Programming Needs in Home Automation. *ACM Transaction on Computer-Human Interaction*, 11:1–11:35.

Daniel, F., & Matera, M. (2014). *Mashups: Concepts, Models and Architectures.* Springer Berlin Heidelberg.

Ur, B., McManus, E., Pak Yong Ho, M., & Littman, M. L. (2014). "Practical Trigger-action Programming in the Smart Home. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 803–812.

Desolda, G., Ardito, C., & Matera, M. (2017). Empowering End Users to Customize Their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools. *ACM Transactions on Computer- Human Interaction, 24*(2), 1-52.

Brackenbury, W., Deora, A., Ritchey, J., Vallee, J., He, W., Wang, G., . . . Ur, B. (2019). How Users Interpret Bugs in Trigger-Action Programming. *Proceedings of the 2019 CHI Conference on*

Human Factors in Computing Systems, 552:1–552:12.

Barricelli, B., & Valtolina, S. (2015). Designing for End-User Development in the Internet of Things. *End-User Development, 9083*.

Zaslavsky, A., & Jayaraman, P. (2015). Discovery in the Internet of Things: The Internet of Things (Ubiquity Symposium). *Ubiquity*, 2:1–2:10.

Corno, F., De Russis, L., & Monge Roffarello, A. (2019). A high-level semantic approach to End-User Development in the Internet of Things. *International Journal of Human-Computer Studies, 125*, 41-54.

Ur, B., Yong Ho, M., Brawner, S., Lee, J., Mennicken, S., Picard, N., . . . Littman, M. L. (2016). Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. *Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems*, 3227– 3231.

Caivano, D., Fogli, D., Lanzilotti, R., Piccinno, A., & Cassano, F. (2018). Supporting end users to control their smart home: design implications from a literature review and an empirical investigation. *Journal of Systems and Software, 144*, 295–313.

Brush, A., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., & Dixon, C. (2011). Home Automation in the Wild: Challenges and Opportunities. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2115–2124.

Huang, J., & Cakmak, M. (2015). Supporting Mental Model Accuracy in Trigger-action Programming. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 215–225.

Danado, J., & Paternò, F. (2014). Puzzle: A mobile application development environment using a jigsaw metaphor. *Journal of Visual Languages & Computing, 25*(4), 297-315.

Corno, F., De Russis, L., & Monge Roffarello, A. (2019). My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor. *End-User Development*, 18–33.

Corno, F., De Russis, L., & Monge Roffarello, A. (2021). Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization. *End-User Development*, 71–86.

Ghiani, G., Manca, M., Paternò, F., & Santoro, C. (2017). Personalization of Context-Dependent Applications Through Trigger-Action Rules. *ACM Transactions on Computer-Human Interaction, 24*(2), 14:1–14:33.

Akiki, P. A., Bandara, A. K., & Yu, Y. (2017). Visual Simple Transformations: Empowering End-Users to Wire Internet of Things Objects. *ACM Transactions on Computer-Human Interaction, 24*(2), 1-43.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67.

Liang, C.-J., Bu, L., Li, Z., Zhang, J., Han, S., Karlsson, B. F., . . . Zhao, F. (2016). Systematically Debugging IoT Control System Correctness for Building Automation. *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, 133–142.

Zhang, L., He, W., Martinez, J., Brackenbury, N., Lu, S., & Ur, B. (2019). AutoTap: Synthesizing and Repairing Trigger-Action Programs Using LTL Properties. *Proceedings of the 41st International Conference on Software Engineering*, 281-291.

Surbatovich, M., Aljuraidan, J., Bauer, L., Das, A., & Jia, L. (2017). Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. *Proceedings of the 26th International Conference on World Wide Web*, 1501-1510.

Corno, F., De Russis, L., & Monge Roffarello, A. (2019). Empowering End Users in Debugging Trigger-Action Rules. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1-13.

Vannucchi, C., Diamanti, M., Mazzante, G., Cacciagrano, D., Culmone, R., Gorogiannis, N., . . . Raimondi, F. (2017). Symbolic verification of event–condition–action rules in intelligent environments. *Journal of Reliable Intelligent Environments, 3*(2), 117-130.

Cao, J., Rector, K., Park, T. H., Fleming, S. D., Burnett, M. M., & Wiedenbeck, S. (2010). A Debugging Perspective on End-User Mashup Programming. *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, 149–156.

Manca, M., Paternò, F., Santoro, C., & Corcella, L. (2019). Supporting end-user debugging of trigger-action rules for IoT applications. *International Journal of Human-Computer Studies, 123*, 56-69.

Corno, F., De Russis, L., & Monge Roffarello, A. (2019). RecRules: Recommending IF-THEN Rules for End-User Development. *ACM Transactions on Intelligent Systems and Technology, 10*(5), 1-27.

Corno, F., De Russis, L., & Monge Roffarello, A. (2020). TAPrec: supporting the composition of trigger-action rules through dynamic recommendations. *Proceedings of the 25th International Conference on Intelligent User Interfaces*, 579-588.

Mattioli, A., & Paternò, F. (2020). A visual environment for enduser creation of IoT customization rules with recommendation support. *Proceedings of the International Conference on Advanced Visual Interfaces*, 1-5.

Srinivasan, V., Koehler, C., & Jin, H. (2018). RuleSelector: Selecting Conditional Action Rules from User Behavior Patterns. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2*(1), 1-34.

Gallo, S., Manca, M., Mattioli, A., Paternò, F., & Santoro, C. (2021). Comparative Analysis of

Composition Paradigms for Personalization Rules in IoT Settings. *End-User Development: 8th International Symposium*, 53-70.

Manca, M., Parvin, P., Paternò, F., & Santoro, C. (2020). Integrating Alexa in a Rulebased Personalization Platform. *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*, 108–113.

Corno, F., De Russis, L., & Monge Roffarello, A. (2020). HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation. *Proceedings of the International Conference on Advanced Visual Interfaces*, 1-9.

Corno, F., De Russis, L., & Monge Roffarello, A. (2021). From Users' Intentions to IF-THEN Rules in the Internet of Things. *ACM Transactions on Information Systems, 39*(4), 1-33.

Huang, T.-H. K., Azaria, A., & Bigham, J. P. (2016). InstructableCrowd: Creating IF-THEN Rules via Conversations with the Crowd. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 1555–1562.

Barricelli, B., Fogli, D., Iemmolo, L., & Locoro, A. (2022). A Multi-Modal Approach to Creating Routines for Smart Speakers. *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, 1-5.

Monge Roffarello, A., & De Russis, L. (2023). Defining Trigger-Action Rules via Voice: a Novel Approach for End-User Development in the IoT. *End-User Development*.

Taivalsaari, A., & Mikkonen, T. (2018). On the development of IoT systems. *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, 13-19.

Taivalsaari, A., & Mikkonen, T. (2017). Beyond the next 700 Iot platforms. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3529-3534.

Stankovic, J. A. (2014, February). Research Directions for the Internet of Things. *IEEE Internet of Things Journal, 1*, 3-9.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012, September). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks, 10*(7), 1497-1516.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials, 17*(4), 2347-2376.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014, February). Internet of Things for Smart Cities. *IEEE Internet of Things Journal, 1*(1), 22-32.

Islam, S. M., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K.-S. (2015). The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access, 3*(678-708).

Atzori, L., Iera, A., & Morabito, G. (2010, October 28). The Internet of Things: A survey. *Computer*

Networks, 54(15), 2787-2805.

Taivalsaari, A., & Mikkonen, T. (2017, January). A Roadmap to the Programmable World: Software Challenges in the IoT Era. *IEEE Software, 34*(1), 72-80.

Fahmideh, M., Ahmad, A., Behnaz, A., Grundy, J., & Susilo, W. (2022, August 1). Software Engineering for Internet of Things: The Practitioners' Perspective. *IEEE Transactions on Software Engineering, 48*(8), 2857-2878.

Larrucea, X., Combelles, A., Favaro, J., & Teneja, K. (2017, January). Software Engineering for the Internet of Things. *IEEE Software, 34*(1), 24-28.

Fortino, G., Savaglio, C., Spezzano, G., & Zhou, M. (2021, January). Internet of Things as System of Systems: A Review of Methodologies, Frameworks, Platforms, and Tools. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51*(1), 223-236.

Weyrich, M. (2016, January). Reference Architectures for the Internet of Things. *IEEE Software, 33*(1), 112-116.

Zambonelli, F. (2017, January). Key Abstractions for IoT-Oriented Software Engineering. *IEEE Software, 34*(1), 38-45.

Corno, F., De Russis, L., & Sáenz, J. (2020). How is Open Source Software Development Different in Popular IoT Projects? *IEEE Access, 8*, 28337-28348.

Corno, F., De Russis, L., & Sáenz, J. (2019). On the challenges novice programmers experience in developing IoT systems: A Survey. *Journal of Systems and Software, 157*, 110389.

Makhshari, A., & Mesbah, A. (2021). IoT Bugs and Development Challenges. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 460-472.

Motta, R. C., de Oliveira, K. M., & Travassos, G. H. (2023, July). An evidence-based roadmap for IoT software systems engineering. *Journal of Systems and Software, 201*, 111680.

Morin, B., Harrand, N., & Fleurey, F. (2017). Model-Based Software Engineering to Tame the IoT Jungle. *IEEE Computer Society Press, 34*(1), 30–36.

Patel, P. (2015, May). Enabling high-level application development for the Internet of Things. *Journal of Systems and Software, 103*, 62-84.

Berrouyne, I., Adda, M., & Mottu, J.-M. (2022, October 15). A Model-Driven Methodology to Accelerate Software Engineering in the Internet of Things. *IEEE Internet of Things Journal, 9*(20), 19757-19772.

Ferreira, L. B., Chaves, P. R., Assumpção, R. M., Branquinho, O. C., Fruett, F., & Cardieri, P. (2022, November). The Three-Phase Methodology for IoT Project Development. *Internet of Things, 20*, 100624.

Čolaković, A., & Hadžialić, M. (2018, October 24). Internet of Things (IoT): A review of enabling

technologies, challenges, and open research issues. *Computer Networks, 144*, 17-39.

Noura, M., Atiquzzaman, M., & Gaedke, M. (2019). Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mobile Networks and Applications, 24*, 796–809.

Asghari, P. (2018, October 15). Service composition approaches in IoT: A systematic review. *Journal of Network and Computer Applications, 120*, 61-77.

Urbieta, A. (2017, November). Adaptive and context-aware service composition for IoT-based smart cities. *Future Generation Computer Systems, 76*.

Hamzei, M., & Navimipour, N. (2018, October). Toward Efficient Service Composition Techniques in the Internet of Things. *IEEE Internet of Things Journal, 5*(5), 3774-3787.

Cabrera, E., Cárdenas, P., Cedillo, P., & Pesántez-Cabrera, P. (2020). Towards a Methodology for creating Internet of Things (IoT) Applications based on Microservices. *2020 IEEE International Conference on Services Computing (SCC)*, 472-474.

Avilés-López, E., & García-Macías, J. (2012). Mashing up the Internet of Things: a framework for smart environments. *EURASIP Journal on Wireless Communications and Networking, 2012*(1), 1687-1499.

Barros, T. G., Da Silva Neto, E. F., Da Silva Neto, J., De Souza, A. G., Aquino, V. B., & Teixeira, E. S. (2022). The Anatomy of IoT Platforms—A Systematic Multivocal Mapping Study. *IEEE Access, 10*, 72758-72772.

Razzaque, M., Milojevic-Jevric, M., Palade, A., & Clarke, S. (2016). Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal, 3*(1), 70-95.

Beniwal, G., & Singhrova, A. (2022). A systematic literature review on IoT gateways. *Journal of King Saud University - Computer and Information Sciences, 34*(10), 9541-9563.

Geraldi, R., Reinehr, S., & Malucelli, A. (2020, August). Software product line applied to the internet of things: A systematic literature review. *Information and Software Technology, 124*, 106293.

Fortino, G., Savaglio, C., Palau, C. E., Suarez de Puga, J., Ganzha, M., Paprzycki, M., . . . Llop, M. (2018). Towards Multi-layer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach. *Integration, Interconnection, and Interoperability of IoT Systems*, 199–232.

Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Computing Surveys, 51*(6), 1–29.

Corno, F., & De Russis, L. (2017). Training Engineers for the Ambient Intelligence Challenge. *IEEE Transactions on Education, 60*(1), 40-49.

Burd, B., Barker, L., Fermín Pérez, F., Russell, I., Siever, B., Tudor, L., . . . Pollock, I. (2018, July). The internet of things in undergraduate computer and information science education: exploring

curricula and pedagogy. *ITiCSE 2018 Companion: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 200–216.

Corno, F., De Russis, L., & Saenz Moreno, J. (2018). Easing IoT development for novice programmers through code recipes. *ICSE-SEET '18: Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*.

Corno, F., De Russis, L., & Saenz Moreno, J. (2022). Computational notebooks to support developers in prototyping IoT systems. *International Journal of Human-Computer Studies, 162*, 102850.

Angelini, L., Couture, N., Abou Khaled, O., & Mugellini, E. (2018). Internet of Tangible Things (IoTT): Challenges and Opportunities for Tangible Interaction with IoT. *Informatics, 5*(1), 1-28.

Kim, K. (2016). Interacting Socially with the Internet of Things (IoT): Effects of Source Attribution and Specialization in Human–IoT Interaction. *Journal of Computer-Mediated Communication*, 420-435.

Shirehjini, A., & Semsar, A. (2016). Human interaction with IoT-based smart environments. *Multimedia Tools and Applications* , 13343–13365.

Kranz, M., Holleis, P., & Schmidt, A. (2009). Embedded Interaction: Interacting with the Internet of Things. *IEEE Internet Computing*, 46-53.

Phupattanasilp, P., & Sheau-Ru, T. (2019). Augmented Reality in the Integrative Internet of Things (AR-IoT): Application for Precision Farming. *Sustainability*.

Shahab, S., Agarwal, P., Mufti, T., & Obaid, A. (2022). SIoT (Social Internet of Things): A Review. *ICT Analysis and Applications*, 289-297.

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., & Gidlund, M. (2018). Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics*, 4724-4734.

Miraz, M., Ali, M., Picking, R., & Excell, P. (2015). A Review on Internet of Things (loT), Internet of Everything (IoE) and Internet ofNano Things (IoNT). *2015 Internet Technologies and Applications (ITA)*, (pp. 219-224). Wrexham, UK .

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 36-52.

Alaa, M., Zaidan, A., Zaidan, B., Talal, M., & Kiah, M. (2017). A review of smart home applications based on Internet of Things . *Journal of Network and Computer Applications*, 48-65.

Laplante, P., & Laplante, N. (2016). The Internet of Things in Healthcare Potential Applications and Challenges. *IT Trends*, 2-4.

Darshan, K., & Anandakumar, K. (2016). A Comprehensive Review on Usage of Internet of Things (IoT) in Healthcare System. *2015 International Conference on Emerging Research in*

Electronics, Computer Science and Technology (ICERECT), (pp. 132-136). Mandya, India.

Jaidka, H., Sharma, N., & Singh, R. (2020). Evolution of IoT to IIoT: Applications & Challenges. *International Conference on Innovative Computing & Communications (ICICC) 2020*, (pp. 1-6).

Kim, W.-S., Lee, W.-S., & Kim, Y.-J. (2020). A Review of the Applications of the Internet of Things (IoT) for Agricultural Automation. *Journal of Biosystems Engineering*, 385-400.

Bellini, P., Nesi, P., & Pantaleo, G. (2022). IoT-Enabled Smart Cities: A Review of Concepts, Frameworks and Key Technologies . *Applied Sciences*, 1-21.

Sharma, N., Shamkuwar, M., & Singh , I. (2018). The History, Present and Future with IoT . *Internet of Things and Big Data Analytics for Smart Generation*, 27-51.

Alsharif, M., Jahid, A., Kelechi, A., & Kannadasan, R. (2023). Green IoT: A Review and Future Research Directions. *Simmetry*, 1-37.

Majeed, U., Khan, L., Yaqoob, I., Kazmi, S., Salah, K., & Hong, C. (2021). Blockchain for IoT-based smart cities: Recent advances, requirements, and future challenges. *Journal of Network and Computer Applications*, 1-33.

Dhanaraju, M., Chenniappan , P., Ramalingam, K., Pazhanivelan, S., & Kaliaperumal, R. (2022). Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture. *Agriculture*.

Singh, P., Elmi, Z., Meriga, V., Pasha, J., & Dulebenets, M. (2022). Internet of Things for sustainable railway transportation: Past, present, and future. *Cleaner Logistics and Supply Chain* , 1-34.

Yugha, R., & Chithra, S. (2020). A survey on technologies and security protocols: Reference for future generation IoT . *Journal of Network and Computer Applications*.

Barricelli, B., Fogli, D., & Locoro, A. (2023). EUDability: A new construct at the intersection of End-User Development and Computational Thinking. *Journal of Systems and Software*.

Van Mechelen, M., & Smith, R. (2023). Emerging Technologies in K–12 Education: A Future HCI Research Agenda. *ACM Transactions on Computer-Human Interaction*, 1-40.