A generic and modular architecture for self-explainable smart homes

Étienne Houzé*†, Ada Diaconescu†, Jean-Louis Dessalles† and David Menga*
*EDF R&D, Palaiseau, France
Email: {first}.{last}@edf.fr
†Télécom Paris, Palaiseau, France
Email: {first}.{last}@telecom-paris.fr

***Abstract*—Explainable AI (XAI) has become a major topic in Artificial Intelligence since the mid 2010s. While smart home explainability promises to improve user experience and trust, it is mostly left outside the scope of current AI research. We identify three main challenges that may cause this delay. First, smart device *heterogeneity* hinders the development of a system-wide vocabulary and communication medium required for end-to-end explanation. Second, smart homes undergo *runtime changes* – e.g. dynamic component additions, deletions and updates – that require corresponding explanatory updates. Third, the explanation *context* may undergo runtime changes, where word meanings may vary with the end-user and the passing seasons. To tackle these challenges, we propose a generic, modular XAI architecture featuring: i) Local Explanatory Components (LECs) that provide resource-specific explanatory expertise and support runtime extensions; ii) mapping capabilities that allow LECs to translate resource-specific monitoring variables into resource-independent abstractions – *predicates* and *events* – which can then be used for generic inter-LEC communication; iii) a generic central component, called *Spotlight*, that coordinates LECs to generate system-wide explanations. We validate our proposal via a cyber-physical prototype of self-explainable smart home, implemented via a physical home maquette equipped with GrovePi sensors. We show how our prototype can handle several realistic scenarios highlighting the main issues identified above. This provides an initial stepping-stone towards a fully self-explanatory smart home solution. The genericity of our proposal opens the way for transferring it to similar application domains.**

***Index Terms*—Smart Home, Autonomic System, Explainable AI**

I. Introduction

In 2016, DARPA identified Explainable Artificial Intelligence (XAI) as an upcoming challenge for AI systems [1]. This call for projects is the consequence of several challenges identified within the AI community. First, the opacity of black box models and their predominance in recent AI techniques raise concerns, especially for critical applications (e.g., autonomous vehicles, medical diagnostics). Second, as most AI techniques rely on Big Data, several concerns have been raised regarding personal data usage. In this respect, authorities enforced regulations, such as the European General Data Protection Regulation (GDPR), which require companies to justify their use of personal data. Going further, the GDPR evokes a "right to explanation" for any AI system's end users [2].

Despite this recent interest, many end applications are unable to provide explanations to users. For instance, consider a room equipped with smart devices [3], such as a temperature-controlling thermostat. In case the temperature crosses the expected range, the user may ask the system to explain this situation. To the best of our knowledge, no solution exists to propose this kind of service. Most consumer devices, such as Google Home or Amazon's Alexa, will treat such specific request as any other generic question (i.e. that can be answered based on common knowledge from the Internet). Even though this approach may generate satisfactory answers, it provides no guarantee of pertinence, as its explanation is not generated based on an understanding of the specific problem. In addition, relying on distant knowledge makes the system dependent to network connection and may raise concerns regarding reasoning opacity and data usage.

This lack of appropriate solutions is brought to the fore in the context of smart homes, or more generally of Cyber-Physical Systems (CPS) – e.g. smart buildings and power grids. The high specificity, dynamism and context sensitivity of such CPS render generic solutions based on similar cases inappropriate. We identify the following key characteristics of smart homes (and CPS) that XAI solutions should address:

a) Resource heterogeneity: : smart homes consist of a collection of smart devices that collectively achieve high-level goals, such as comfort, power consumption or security [4], [5]. Each device can present its own interface: think, for instance, of how a window controller and an electric appliance differ. An explanatory system compliant with smart homes must handle this heterogeneity, while most of current XAI research focuses on monolithic, uniform systems. In addition, as devices may be provided by different manufacturers, internal knowledge may not be available for house-wide applications, as certain critical variables may be kept private.

b) Runtime adaptation: : most smart home solutions feature modular, flexible architectures, allowing for "plug-and-play" capabilities and runtime adaptation to context and goal changes [6]. While these properties contribute to maintainability and performance, they may also introduce surprising or unexpected behaviors that require explanations for users. For instance, if a system self-adapts its behavior following the

This work is part of a PhD thesis funded by the Association Nationale de la Recherche et de la Technologie (ANRT), grant XXXXX, and co-hosted at EDF&D and Télécom Paris.

addition of a new device, the house occupant may inquire the reason for the perceived change. However, the impact of these properties in XAI are, to the best of our knowledge, out of the scope of current XAI research.

c) Context-dependence: : rather than based on a fixed ontology, words employed in natural language may change their meaning to users depending on the context in which they are used – e.g. the definition of the concept of "cold" differs depending on the user, the time of the year or the country. This introduces new difficulties when explaining situations within smart homes, as such vocabulary variations must be handled and integrated within the system's explanatory reasoning.

Based on these challenges, we identify several architectural principles that we believe self-explaining smart home systems should follow. Inter-device communication is mandatory for generating system-wide explanations, which must integrate specific knowledge from several system components. This communication must be agnostic to devices types (Cf. challenge 1. above), hence necessitating *generic*, standard communications. To be able to handle runtime changes and adaptation (Cf. challenge 2), the explanatory system's architecture must be *modular*: local explanatory components (LECs) must contain the specific knowledge needed to explain the behavior of dynamic system components; these LECs can be added, removed and managed at runtime so as to follow corresponding adaptations in the underlying system components to be explained. To consider context as part of the explanatory reasoning (Cf. challenge 3), the system must be able to self-observe and re-evaluate its assumptions – e.g., the meaning of words to users, such as the temperature range corresponding to the word "cold"; or, the accuracy of its temperature sensors in time. It can then propose as explanations the differences between previous and updated assumptions.

In this paper, we propose a self-explanatory system that satisfies the above criteria – i.e., implementing a generic and modular architecture, and taking context into account (Figure 1). We consider that a smart home contains various kinds of Observable Components, either Managed Resources or Autonomic Managers (AM), according to the generic architecture of Autonomic Computing systems [6]. Each component is observable via a resource-specific interface, which allows to plug-in a corresponding Local Explainable Component (LEC) that analyzes, reasons about and explains this component. LECs then translate their resource-specific observations into higher-level Boolean propositions using a set of generic *predicates* and recorded *events*. This enables standard inter-LEC communication, using the high-level Boolean propositions. A central component, the Spotlight, coordinates the LECs to generate system-wide explanatory reasoning, following a process introduced in a previous work [7]: Decentralized Conflict-Abduction-Simulation (D-CAS).

The rest of this paper is organized as follows. In Section II, we review existing works on autonomic systems and AI explainability to better understand the problem of generating explanations for smart homes. We detail the proposed architecture and its features in Section III. We present the implementation of a proof-of-concept demonstrator that mimics a realistic setup on embedded devices in Section IV. In Section V, we draw conclusions based on the first results and propose future research axes towards explainable autonomic systems.

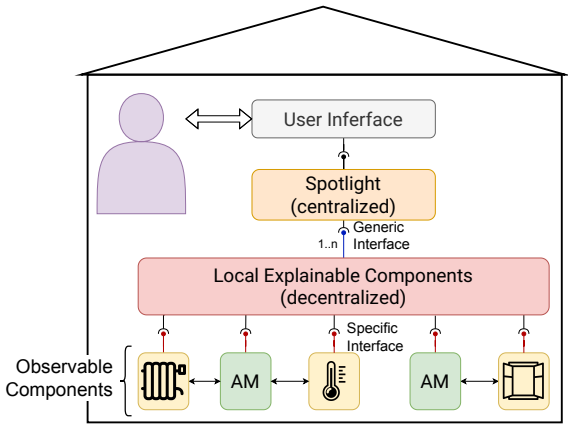

Fig. 1: Architectural overview of the proposed Explanatory System. Smart devices (yellow) and Autonomic Managers (AM, green) are observed by a layer of Local Explainable Components (LECs, pink) via resource-specific monitoring interfaces (red). The LECs then present generic interfaces to a central "Spotlight" coordinator.

II. Background and Related Works

Autonomic Computing (AC) systems maintain their operations while requiring minimal user intervention when dynamic changes occur [6], [8]. Namely, AC systems can self-monitor, self-diagnose and self-modify in response to changes in their execution environment, internal resources or targeted objectives. This self-adaptive behavior can be implemented in various manners depending on the system requirements and constraints. We focus here on AC systems that feature modular, decentralised architectures, as needed to self-adapt to structural changes in the managed system [**?**](e.g., dynamic addition or removal of managed resources). This mostly suits the targeted application domain of Cyber-Physical Systems (CPS), e.g. smart homes in particular.

Smart homes are defined as collections of various connected devices and controllers that interact with the shared physical environment of an individual house to achieve different high-level goals such as comfort, security and power efficiency [4]. In many aspects, smart homes are examples of the Autonomic Computing paradigm, as their goal is to automatize mundane house management and minimize user intervention.

In recent years, XAI has put the focus on "opening the black box" of opaque AI models. Notably, methods such as LIME [9] or SHAP [10] propose to explain the output of classifiers in terms of feature relevance: they identify which feature of the input data points was the most discriminating in the classification result. For instance, they may determine that a person's age was a bank algorithm's prime decision factor in a

loan demand rebuttal. A variety of approaches and techniques exist to handle various AI models and applications [11].

However, we argue that existing XAI methods are not satisfactory regarding the challenges presented in Section I. First, as previously evoked, most XAI techniques aim to explain a single monolithic opaque AI model, while smart homes are built as collections of numerous heterogeneous devices. Second, as some criticism highlights, most XAI methods are not targeting end-users. Rather, they focus on synthetic benchmarks which do not reflect user needs [12]. For instance, the output of visualization-based methods, such as GradCam [13] or LIME [9], can be useful for an AI expert, but are far from the intuitive notion of explanation that would satisfy non-expert users.

To the best of our knowledge, no XAI solution focuses on generating explanations for AC systems at runtime, while addressing all highlighted challenges. E.g., in [14], the MAB-Ex (Monitor - Analyze - Build - Explain) framework proposes to tweak the popular MAPE (Monitor - Analyze - Plan - Execute) computing scheme [6] into an explanatory system. Instead of planning system adaptations, the Build process updates a causal system model that can then be used to produce user-specific explanations (via the Explain process). While pertinent, the proposal is rather preliminary and offers few reusable artifacts. Moreover, maintaining a centralized system model may raise scalability and maintainability issues. In our proposal, each LEC maintains a local model of the system resource that it is supposed to explain; and the system-wide model relevant to each explanation is generated at runtime via the Spotlight's coordination of relevant LECs. In previous works, we have also studied the development of causality models that could serve as a base for self-explanation [15].

III. Proposed Generic Architecture

A. General principles and organization

The proposed architecture is depicted in Figure 2. We consider that a smart home's devices and autonomic managers (or controllers) are observable (i.e. can be monitored via Specific Interfaces); and we refer to them generically as Observable Components (OCs). This assumption is hardly restrictive, as component observation is a common feature of Autonomic Systems [8] and smart devices [3].

Additionally, we assume that each OC-provider will offer additional OC-specific modules (e.g. for abduction, interpretation and simulation – Cf. Section III-E) that allow reasoning about and explaining that particular OC. These modules must implement standard interfaces to be integrated into the explanatory system, as detailed below. This addresses the heterogeneity issue, as each OC can come with its own implementation of explanatory functions, appropriate for its specific design and purpose. Each OC provides a standard specification (Cf. Section III-E), indicating its specific modules and its observable interfaces.

Local Explanatory Component (LECs) are generic components that are created and customized at runtime based on OC specifications. That is, when an OC is added into the system, an associated Local Explanatory Component (LEC) is created and customized based on the specifications exposed by the OC. First, the LEC downloads and integrates the OC-specific modules into its generic framework (i.e. plug-and-play). This endows the LEC with inference knowledge (e.g. inference rules, decision or machine learning techniques), allowing it to determine the causes and consequences of its attached OC's state. Then, the LEC can interpret the data observed from its attached OC so as to produce higher-level objects: *events* and *predicates*.

To allow for inter-LEC reasoning and explanations, all LECs expose the same Generic Interface and communicate based on high-level representations of predicates and events. A central component, the *Spotlight*, keeps an updated record of LECs present in the system, integrating them via their Generic Interfaces and coordinating them to generate system-wide explanatory reasoning. The Spotlight itself is observable, hence allowing a LEC to be associated to it and to enable self-observation.

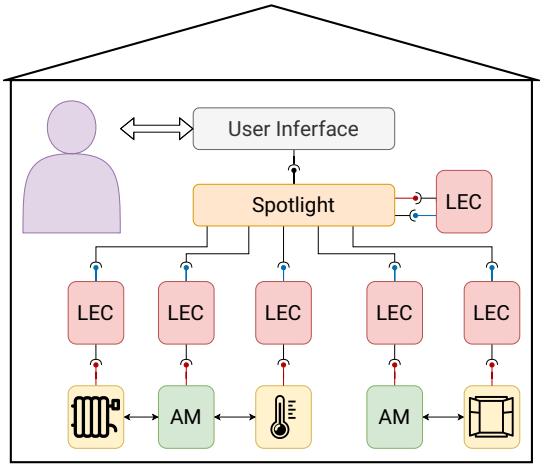

Fig. 2: Local Explanatory Components (LECs) handle observations from associated Observable Components, via Specific Interfaces (in red). All LECs expose to the Spotlight the same standard Generic Interface

B. D-CAS overview: integrating LECs

In previous work, we proposed a generic process for generating explanatory reasoning, named Decentralized Conflict-Abduction-Simulation (D-CAS) [7]. This method is implemented within the Spotlight, allowing it to select relevant LECs for each explanation, to query these LECs for partial OC-specific explanations and to integrate these local explanations into a coherent end-to-end explanation for the user.

More precisely, each explanation query is triggered by a discrepancy between an observation and an expectation. Formally, this is conveyed by a *conflict* (P, N) where P is a Boolean proposition and N a number. P encodes the object of the explanation, while N conveys the intensity of the request. For instance, if the user inquires the reasons for a window being open, his/her request is considered as the conflict

$(\texttt{open}(\texttt{window}), -30)$. As the Spotlight has no knowledge of the system's state and logic, it forwards the request to the most relevant LEC, asking it to further investigate the conflict. In our example, the relevant LEC would be the one attached to the window, so the Spotlight would send the request to it.

Depending on the query's context and the LEC's capabilities, the LEC's reply may be one of the following:

- `ABDUCTION`, (C, N_C): a possible cause for the window being open (determined via abduction). C is the proposition defining the identified cause; N_C its necessity.
- `ACTION`, (A, N_A): an action that can revert the window being open (determine via simulation). A is the proposition defining the identified action; N_A its necessity.
- `GIVE` $-$ `UP`: meaning that no other possibility was found, or that the conflict cannot be confirmed.

In case the LEC's response is based on abduction or action (i.e. cause (C, N_C) or action (A, N_A)) the Spotlight considers this response as a new conflict and repeats the process, as above, by contacting the next relevant LEC that may handle it. This recursive line-of-reasoning develops progressively, from one LEC to the next, until a LEC returns a `GIVE-UP` response. This may occur either because the LEC proposed an action that was executed and resolved the conflict (hence the LEC indicates that the conflict no longer exists); or, because the questioned LEC has ran out of abduction and action options to explore. The Spotlight interrupts the line of reasoning that lead to the giving-up LEC and starts exploring an alternative line-of-reasoning by re-questioning a previous LEC (which may propose alternative causes or actions). Exploring such alternative corresponds to branching a new line-of-reasoning from one of the LECs interrogated before.

Hence, the overall explanation process that the Spotlight coordinates produces a tree-like structure: starting from the initial conflict that generated the query (root) and spawning various lines-of-reasoning (branches) depending on the alternative responses of interrogated LECs (e.g. Figs. 10 and 11). This explanatory tree provides the generic basis for producing a final explanation to users – translating such formal explanation into natural language is out of the paper's scope.

In the rest of this section, we propose the architecture design that enables D-CAS while presenting the desired properties of knowledge locality and modularity.

C. Using Events and Predicates

Figure 3 depicts the internal LEC architecture. A Central Unit interfaces the LEC with external components (i.e. handles monitoring data from the associated OC; and D-CAS requests from the Spotlight); and coordinates the execution of the other LEC sub-components (i.e. Interpretation and Abduction Units; Events and Predicates repositories).

The LEC uses its Interpretation Unit to analyse data observed from its attached OC and then to translate this data into Boolean propositions, which D-CAN uses. This analysis is done in two steps. First, the Interpretation Unit records high-level *Event* objects from the incoming data streams. Events are static representations of noticeable trends or patterns in the

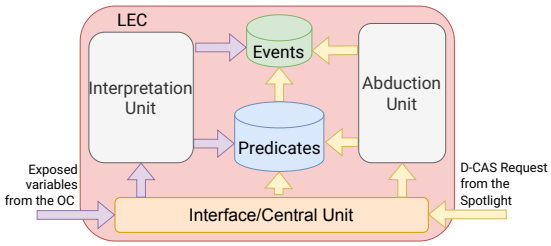

Fig. 3: Internal architecture of Local Explanatory Component (LEC). A LEC interprets data observed from its associated OC and creates high-level events and predicates (purple arrows); it also processes D-CAS requests from the Spotlight by performing abduction and simulation (not shown) (yellow arrows).

data. For instance, an event can correspond to a temperature being higher than a given threshold, or a perceived movement in the room. Formally, events are defined as triples (t, l, X):

$$e = \begin{cases} t \in \mathbb{N} \\ l \in Lab \\ X \in \mathbb{R}^{d_l} \end{cases} . \tag{1}$$

Here, t is the *timestamp* of the event, l a *label* indicating its nature, Lab is the set of possible labels and X is a vector of d_l characteristics. For instance, the system would consider a high temperature as an event e_h with the time of occurrence t_h, the label $l_h = \texttt{high\_temp}$, and characteristics $X_h = \{\text{device} : \texttt{thermo19}, \text{max\_temp} : 27°\texttt{C}, \text{duration} : \texttt{2930s}\}$.

As events are of different natures, they do not share the same characteristic dimensions – it is hence impossible to compare them directly. To allow for inter-LEC communication, we introduce *predicates*. Predicates are Boolean functions over events that characterize them. Formally, a predicate p is defined as:

$$p : \begin{cases} \mathcal{E} \times \mathbb{N}^k \mapsto Props \times \{0, 1\} \\ (e, a_1, \ldots, a_k) \mapsto (prop, val) \end{cases} , \tag{2}$$

where k is the *arity* of the predicate p, i.e. the number of additional arguments $a_1, \ldots, a_k$ used by the predicate. The result of the predicate's application on an event and arguments is a Boolean proposition and its value. In our vision, Boolean propositions are constructed as Strings. To illustrate this, consider a predicate p_{in} indicating whether an event occurred in a given location. The arity of this predicate is 1, as it takes the desired location loc as an extra argument. Given the previously considered high temperature event e_h, $p_{in}(e_h, \texttt{room1})$ will return the tuple $(\texttt{in}(\texttt{e}, \texttt{room1}), true)$ if the device `thermo19`, which reported e_h, is in `room1`.

The Interpretation Unit identifies events and predicates from OC observations and stores them in distinct memories. Here, we operate the following distinction. On the one hand, events correspond to structures identified in the data and to physical measures that cannot be denied – hence, they are *immutable* (i.e. they represent facts that once recorded can no longer be altered). On the other hand, predicates correspond to the

LEC's interpretation of recorded events – as such, they are *mutable* (i.e. the predicate's definition may change, as the LEC's and the user's understanding may vary in time). For instance, consider the predicate p_{hot} indicating whether a recorded event corresponds to a hot feeling. Depending on the context, this notion varies: $20°C$ can be considered hot in the winter and a normal temperature during summer. Hence, the LEC's Interpretation Unit can update predicate definitions so as to adapt them to the current context.

Figure 4 models Predicates, Events and their relation via a class diagram. A Predicate object contains: a *signature* attribute defining the number and types of received arguments; and two methods, *evaluate* and *toProposition*, that correspond to the definition in Equation 2. Similarly, an event contains three constant attributes, *timestamp*, *label* and *characteristics*, corresponding to the event definition in Eq. 1.

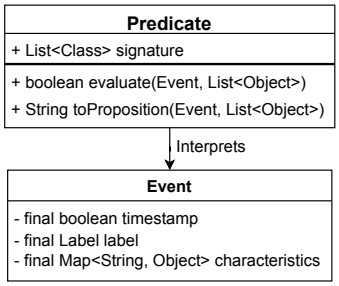

Fig. 4: Class diagram of Predicates and Events. Note that Event objects are immutable, with only final attributes.

D. LEC processing of a request query

During the explanation process described in subsection III-B a LEC contributes to an explanation when requested by the Spotlight to further investigate a query: (P, N), where P is a Boolean proposition and N a number transcribing the intensity of the request. In this subsection, we detail how the LEC processes this request to generate one of the three possible answers: ABDUCTION, ACTION or GIVE-UP (Cf. III-B).

Figure 5 depicts the LEC's request-handing process as a sequence diagram. The process is triggered when the LEC receives a request from the Spotlight conveying the questioned proposition and the associated necessity (P, N) (01). The LEC then checks whether it can find a predicate p, arguments $a_1, \ldots, a_k$ and an event e such that

$$p(e, a_1, \ldots, a_k) = (P, true)\,. \tag{3}$$

This corresponds to calls (02) and (03) in the sequence diagram. In case this search is successful, the LEC's Central Unit transfers the request to the Simulation Unit (not depicted in Figure 3 for clarity) and the Abduction Unit (08, 04). These units will identify if, to the best of their knowledge, there exists a causal hypothesis C that can provoke P or an action A that could revert P, respectively (calls 05 to 07, and 09 to 11). For each proposition, a necessity score is computed to account for the estimated strength of the causal relation or mutability of the

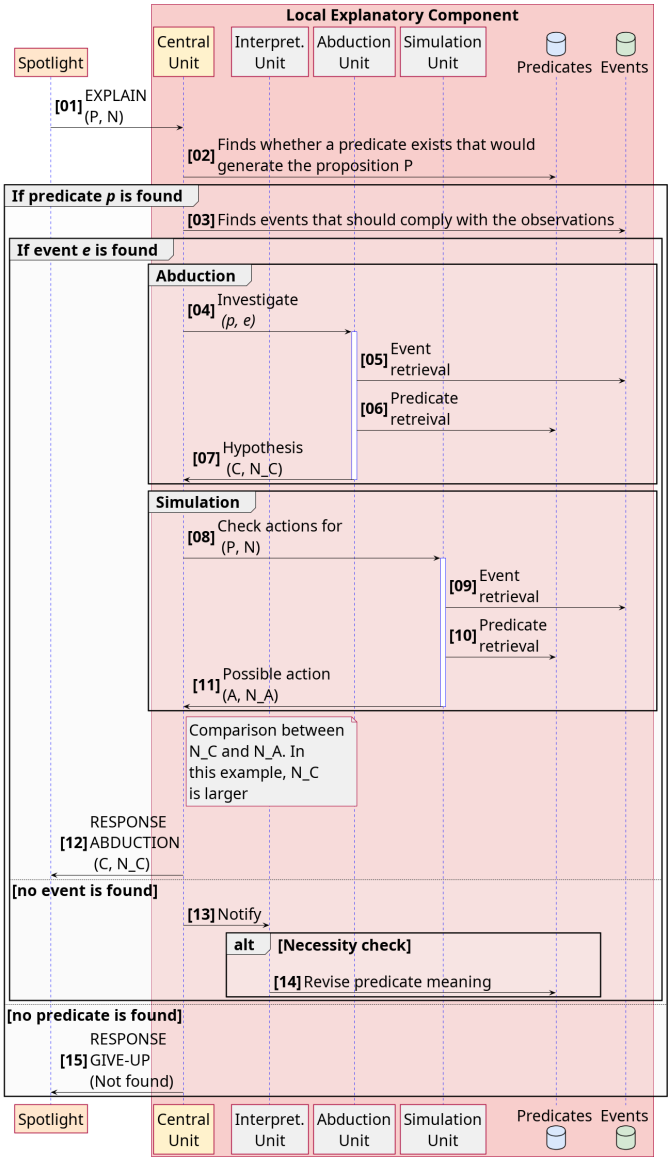

Fig. 5: Sequence of a request processing by the LEC.

proposed action. These necessities are denoted N_C and N_A, respectively. Both results are returned to the LEC's Central Unit, which compares both necessity scores and returns the highest one to the Spotlight, with the corresponding flag. In Figure 5, this corresponds to the result of the abduction unit (12). In case both the Abduction and the Simulation Units fail to find a solution, the LEC returns `GIVE-UP` instead.

In case the LEC fails to find a solution to Equation 3, two options exist. If no predicate exists in memory that corresponds to the requested proposition P, the LEC returns a `GIVE-UP` response to the Spotlight to notify this mismatch (15). Conversely, if a predicate p in the LEC's memory corresponds to the proposition P, but no recorded event can be identified, the Central Unit notifies the Interpretation Unit of the situation. Then, depending on the intensity $|N|$ of the request and of the beliefs of the Interpretation Unit, predicate

revision is possible (14): the Interpretation Unit may revise the meaning of the predicate p so that a recorded event e matches Equation 3 (14). For instance, if the LEC associated to the temperature controller is requested to explain `cold`, but finds no recorded event that matches its definition of cold, the LEC may revise this definition so that some recent event can be identified as cold. This mechanism takes advantage of the mutability of predicates.

E. A modular architecture

As smart homes are highly heterogeneous and dynamic (Cf. challenges 1 and 2 in sec. I), we aimed to provide a "plug-and-play" feature that enables the runtime discovery and integration of new system components. While this feature already exists for autonomic devices [6], [16], nothing comparable has been designed for explanatory systems. In this subsection, we detail the adaptation features enabled by our architecture.

As the system's Observable Components change, the associated LECs must adapt their reasoning and explanation capacities accordingly. Moreover, as new OCs are added to the system dynamically, corresponding LECs must be added to be able to reason about and explain them. We ensure this flexibility by providing a modular LEC architecture 6. Namely, the proposed LEC provides a generic framework, which is specialized for understanding its attached OC by the addition of dedicated modules. That is, the Abduction, Interpretation and Simulation Units are composed of various modules that can be added, removed or updated at runtime.

A local manager in the LEC's Central Unit supervises LEC modifications and records them as events, which can then be characterized by predicates, to be integrated into the explanatory reasoning as any other event. This manager also maintains an up-to-date internal context that can be used by some modules to integrate self-observation in the explanatory reasoning. The internal manager also listens to predicate changes and can record corresponding events.

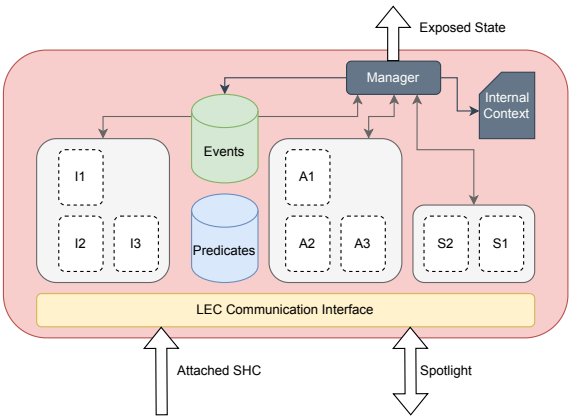

Fig. 6: The modular architecture of a LEC. Each unit is composed of different modules: I1, I2 and I3 are Interpretation Modules; A1, A2 and A3 are Abduction Modules; S1 and S2 are Simulation Modules.

As device manufacturers and controller providers detain specific knowledge and experience regarding the internal logic and interpretation of data recorded by their Observable Components' sensors, it is logical to allow them to define the characteristics and implementations of the modules that specialize the LEC's reasoning and explanation based on these data. Hence, each OC comes with a specification that defines its explanation-related artefacts, to be integrated by LECs. The OC specification is formatted as shown in Figure 7 (similar to the universal IoT spec in [17]). Namely, the OC specification, contains: i) an OC id; ii) information for integrating the OC within the smart home (e.g. sensor and actuator specs based on a standard description – out of scope); iii) the LEC-specific modules (i.e. available classes implementing the LEC's Abduction, Interpretation and Simulation modules); and, iv) OC characteristics (e.g. manufacturer, version, type). The LEC-specific modules (iii) are the most relevant to our proposal as they allow customizing a LEC for the OC.

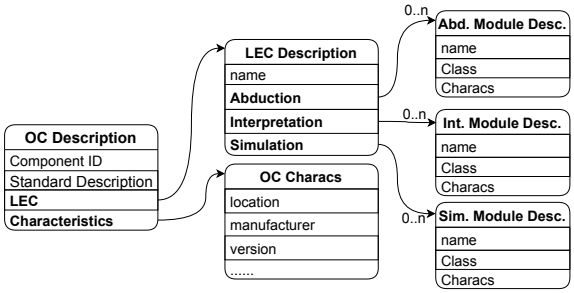

Fig. 7: The OC description is enriched with the addition of the LEC description, containing the characteristics and classes of its specialized modules.

Subsequently, an OC's integration process occurs as shown in Figure 8. When the user deploys a new device, the smart home discovers it automatically, e.g. as supported by many service-oriented management solutions. A typical solution, which we adopt here, is to employ a Naming and Directory service, or registry, that keeps track of all system components (i.e. identities and properties) and that any component can access to learn about new additions. The Spotlight observes this registry periodically, hence noticing an OC addition and fetching its exposed properties (based on the OC specification, Fig. Figure 7). The Spotlight then creates a new LEC and instantiates the specific modules for Abduction, Interpretation and Simulation. Upon its creation, the LEC connects to the newly added OC. Once the connection established, the LEC notifies the Spotlight, which adds the LEC to its own registry.

OC updates and removals follow the same process: changes are noticed via the Naming and Directory service and propagated into the associated LEC; the Spotlight registry may be updated accordingly.

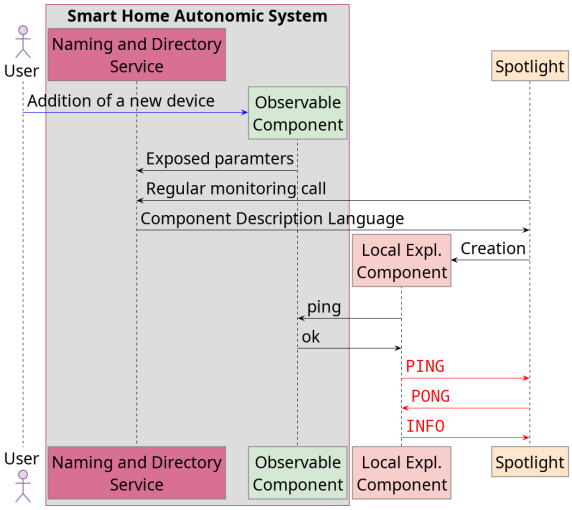

Fig. 8: Adding a LEC

IV. Implementation and Examples

A. Implementation Description

To demonstrate the feasibility of the proposed explanatory system architecture and to provide illustrative examples, we developed a proof-of-concept demonstrator based on a cyber-physical smart home model (Fig. 9. It consists of a physical house model equipped with various sensors and actuators (i.e. GrovePi thermometers, heaters, windows, lights), which are connected to different Raspberry Pi (i.e. one per room). All Raspberry Pi are connected to a central NUC station.

Plugged-in devices and corresponding OCs are administered via a Local Control Manager, on each Rasberry Pi (RPi). Within the central NUC, a Central Manager keeps track of all connected RPi and exposes a Naming and Directory service, i.e. a registry that keeps track of all known devices [6]. Communication between the Central and Local Managers relies on REST requests. This constitutes the autonomic system of the house model (e.g. that monitors and controls room temperatures, windows and lights).

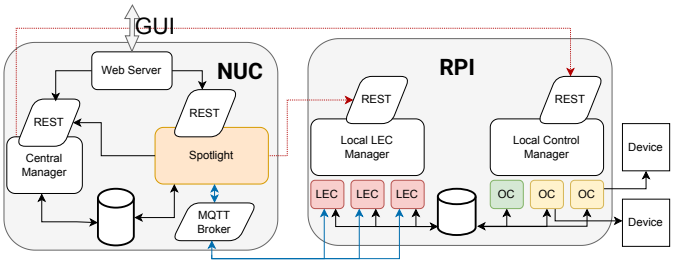

Fig. 9: Overview of prototype demonstrator. Inter-host communications are performed either via REST (orange arrows) or MQTT (blue arrows). Each RPi hosts its own database, as shared memory for exposing OC data to the LECs.

On top of the autonomic system, we implemented the explanatory components described in Section III: the Spotlight is deployed on the central NUC and LECs are hosted on the RPi, where the corresponding OCs are located. On each RPi, a local *LEC Manager* is responsible for the LECs' lifecycles (i.e. allowing the Spotlight to create and customise them, via a REST interface). After the initial LEC creation and registration, communications between the Spotlight and the LECs relies on the MQTT protocol [18], with each component listening on a specific channel named after its unique ID.

LECs obtain monitoring data from OCs via a shared memory (i.e. Redis database). On each RPi, hosted OCs write their observations into a dedicated stream field in the database; corresponding LECs listen to this stream and process the data. A similar construct is implemented in the central unit (NUC), where the Spotlight records its state on a stream field in a Redis Database and access the exposed naming and directory service of the Central Manager. This usage of a database as a shared memory for observation allows to conveniently unify the observable interfaces required by the explanatory system.

To illustrate the defining features of our architecture, i.e. its runtime adaptation, knowledge revision and self-awareness integration into the explanatory reasoning, we provide two scenarios representative of typical smart home situations.

B. Example 1: runtime integration

The first scenario demonstrates how our explanatory system manages the integration of a new device and updates its explanatory capabilities accordingly. In the initial situation, the user feels that a room is colder than expected and queries the system about this situation. From this starting point, we develop three possible outcomes, as shown in Figure 10. For more details, step-by-step versions of the presented tree outputs of our explanatory system are available online at https://explainableai.fr/ACSOS22.

1) Figure 10a: In the first outcome, the incoming explanation request is transcribed as $(\texttt{cold}(\texttt{room1}), -10)$, then forwarded by the Spotlight to the LEC attached to the Temperature Controller in $room1$. The questioned LEC uses its abductive module to propose that the cause may be that a window is open – this is transcribed by the proposition $\texttt{deviceType}(\texttt{window}) \wedge \texttt{open}()$. However, in this case, there is no smart window component in the system. Therefore, this hypothesis cannot be further explored, resulting in a "give-up" LEC response and corresponding node in the tree output. As the Temperature Controller's LEC can find no other hypothesis, the conflict is discarded. The final result of this situation is that the explanatory system can propose a possible cause, while possessing no further competence to explore it. In this case, it is possible to integrate the user into the loop by asking him/her directly to confirm whether a window is open: the tree output can be transcribed as "It is cold because a window may be open; the system cannot confirm whether this hypothesis is true."

2) Figure 10b: The initial situation is identical to the one in the previous example. However, in the meantime, the user installed a smart window into the system. Hence, the hypothesis that the Temperature Controller's LEC proposes,

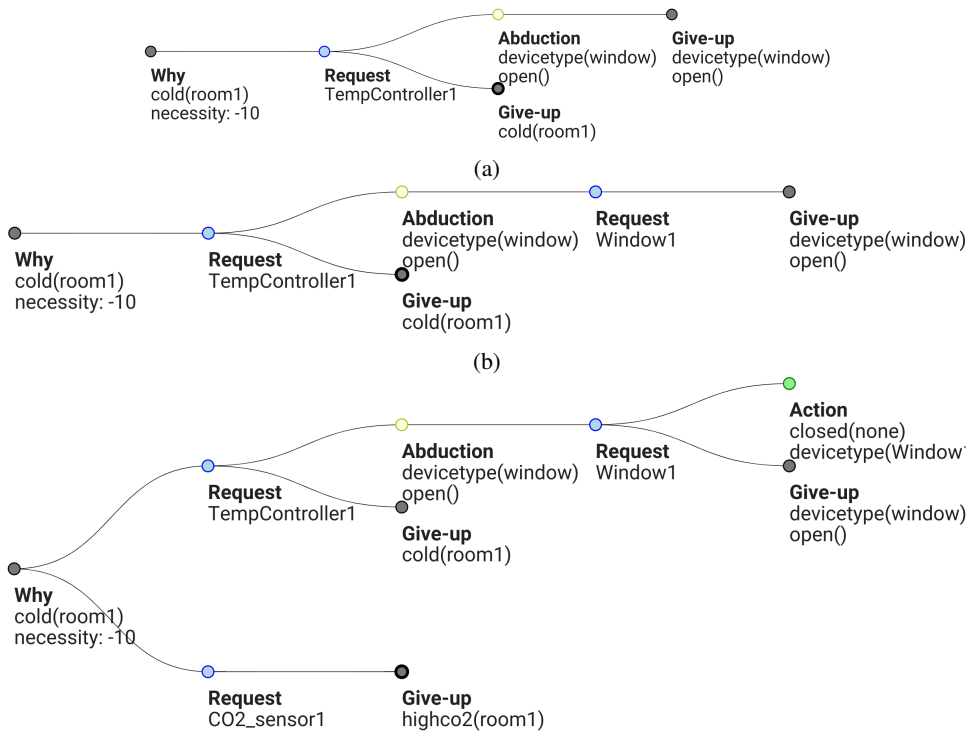

Fig. 10: Different explanatory tree outputs of D-CAS. The outcomes depend on system capabilities: (a) no smart window is present. (b) a smart window is added, allowing the system to confirm its hypothesis. (c) a simulation module is added to the window, enabling to discover consequences of the window's closing.

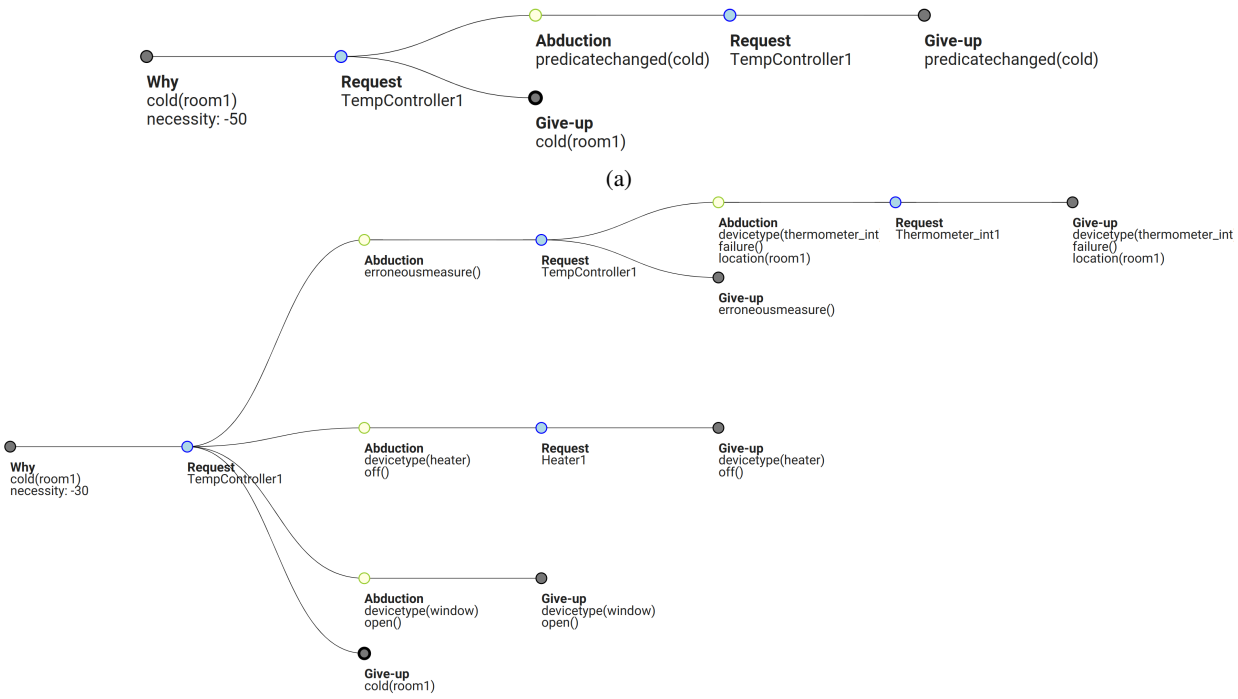

Fig. 11: Two scenarios illustrating knowledge revision. (a) the LEC revises its definition of *cold*, and is able to include this change in the subsequent reasoning. (b) the LEC changes its belief and assumes that the reported measures are erroneous, leading it to consider that the thermometer may be failing.

$\texttt{deviceType}(\texttt{window}) \wedge \texttt{open}()$, fits within the knowledge of the newly added window's LEC, and can be further investigated. The new LEC confirms that the window is open, but has no module for determining a causal hypothesis for this state, or an action that could revert it. Therefore, this line-of-reasoning is, in the end, discarded. This time, with the addition of the smart window, the explanatory system is able to confirm to the user that the proposed hypothesis of the open window is coherent with its observations, but can go no further. The transcribed output is: "It is cold because a window is open; the system cannot further investigate this hypothesis."

3) Figure 10c: The initial situation is identical to the previous example. This time, the window's LEC has been updated with a new simulation module. This addition allows the window's LEC to propose the idea of closing the window to effectively end the conflict that it had been questioned about, i.e. the window being open. The simulation of this action can be operated by a digital twin of the house, or directly by the simulator module of the window, depending on their capabilities. In case no module can handle the simulation of the proposed action, it is possible that the system asks the user to perform the action instead. By simulating the window's closing, the system realizes that the room is no longer cold. However, another conflict is now raised by the CO_2 sensor, as the concentration becomes higher than the desired limit. In the end, the addition of the simulation capabilities to the window's LEC allows the system to improve its reasoning output and confirm that the window was indeed the cause for the low temperature. At the same time, the system is able to identify that the window was open to prevent the CO_2 concentration from being too high. Here, the output is transcribed as: "It is cold because the window is open. The window is open because closing it would provoke high CO_2 concentration."

C. Example 2: Knowledge revision

Self-observation and its interpretation is a key element to design an explainable smart home system: we argue that it is necessary that the system integrates its own changes and modifications into its reasoning to achieve explainability. In our proposed architecture, this is achieved by including within LECs local managers that record observed changes as events, which can later-on be processed and integrated into the D-CAS algorithm without additional input from the user. To demonstrate the benefits of this approach, we present in Figure 11 the tree outputs of two situations where knowledge revision is integrated into the explanatory output.

1) Figure 11a: In a first situation, the user feels uncomfortably cold in the room. He/she inquires the system for the reason, which corresponds to the request $(\texttt{cold}(\texttt{room1}), -50)$. The Spotlight identifies that the relevant component to process this request is the LEC attached to the Temperature Controller and hands the inquiry to it. However, the LEC cannot confirm that the conflict is occurring: its `cold` predicate compares temperatures to a given threshold, e.g. $18°C$, while recent recorded events show a temperature of $18.2°C$. As the difference is small, the intensity of the request, -50, suffices to make the Interpretation module to revise its knowledge and modify the threshold, increasing it to $18.5°C$; this change is recorded by the LEC's manager in a dedicated event. As the conflict is now confirmed, the LEC continues to process the request. An abduction module proposes that a recent predicate change is the reason for the current situation. This hypothesis is considered, and confirmed by the temperature controller's LEC: the recent change of the meaning of `cold` matches this hypothesis. The final output of the explanatory system is that it is cold in the room because the notion of cold had to be changed to correspond to the user's perception of the situation.

2) Figure 11b: In this second situation, the initial setup is similar to the previous example. Again, the Temperature Controller's LEC does not confirm that it is cold in the room: its measures indicate that the room is at $28°C$. This time, the difference between the measures and the threshold ($18.5°C$) is too large to consider increasing the `cold` predicate's threshold. However, given that the reported temperature is abnormally high, the LEC may revise its definition of the `erroneous` predicate, to consider that the currently recorded event is based on false reports. By doing so, the LEC can now assume that the temperature is indeed cold, and proceed with the rest of the process. Its abduction module first proposes that the erroneous measure can be the cause of the perceived cold. In the ensuing call, it proposes that the erroneous measure originates from a failure in the reporting thermometer. Hence, the request is forwarded to the thermometer's LEC, which confirms that it may be failing to report the correct temperature, but cannot propose further causal hypotheses or actions for this state. Hence, this line-of-reasoning is now discarded. Following the D-CAS process order (Cf. subsec. III-B, detailed in [7]), the request goes back to the Temperature Controller's LEC to find other hypotheses. The abduction module then proposes that the heater being turned off may be a cause for the room's low temperature, but this lead is discarded as the heater's LEC finds that the heater is switched on. Back to the Temperature Controller's LEC, the abduction module now proposes that a window may be open, but this lead is discarded as the system does not contain a smart window to further investigate this hypothesis. In the end, the system's output corresponds to the following text output: "The room may be cold because of an erroneous measure, which originates from the thermometer failure. Other hypotheses, such as the heater being turned off or the window being open, could not be confirmed."

V. Conclusions and Perspectives

This paper proposed a generic, modular architecture for self-explanatory systems. Here, the knowledge necessary for causal reasoning and explanation is decentralized: Local Explanatory Components (LECs) monitor each observable component (OC) in the smart home, interpret their variables and formulate causal or simulation hypotheses. A central coordinator, the Spotlight, integrates their local reasoning into a system-wide explanatory process and explanation to users. While LECs are OC-specific, the Spotlight is generic and has no access to

knowledge of the system's state. This renders our proposal applicable across widely heterogeneous explanatory systems.

By introducing the proxy of events and predicates, from which Boolean propositions are constructed, our architecture preserves knowledge locality: components communicate Boolean propositions without having to disclose their inner logic or their proposition meanings. This allows to preserve privacy of the components' functioning while enabling system-wide coordination and explanation generation.

Our system is able to handle runtime adaptation, which is a prominent feature of Autonomic Computing: it is possible to add/remove or update devices or LEC modules at runtime, so as to extend or evolve the system's explanatory capabilities. Also, the LECs' self-observation features are directly integrated into the explanatory process and treated equally as observations from the OC. This allows the system to generate explanations from self-awareness considerations, for example understanding that some measures are erroneous or do not correspond to the user's perception of the situation.

The viability of our architecture was illustrated via a prototype implementation and several realistic scenarios. Results highlighted the principal advantages of the proposed architecture: ability to explain widely heterogeneous system components; ability to extend the system's explanatory capabilities when new system components are added; and ability to evolve its explanation vocabulary when the context changes (e.g. user preferences or defective sensors).

Future work may focus on developing and improving interpretation, abduction and simulation methods. Interpretation Modules may incorporate outlier analysis methods [19] to detect events; or implement online learning of predicates by using contrast-based operations [20]. Similarly, advances in XAI such as feature relevance can be considered to provide hypotheses for abductive inference; while dedicated twin home simulators [21] can be used as simulators and added to the explanatory system.

Another area of future research is the integration of user(s) into the explanatory reasoning, by adding a LEC "attached" to the user (one to each user) which would interpret the user's behavior and reactions as events and predicates.This feature would allow the system to formulate explanations such as "this month's power consumption is low because you were away for two weeks" or "the room is cold because another user changed the setting yesterday".

Overall, our architecture provides an initial basis for developing more advanced explanatory systems, its genericity allowing to consider applications them to further CPS domains.

REFERENCES

[1] "Explainable Artificial Intelligence," DARPA, Broad Agency Anouncement DARPA-BAA-16-53, Aug. 2016.

[2] B. Goodman and S. Flaxman, "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"," *AI Magazine*, vol. 38, no. 3, pp. 50–57, Oct. 2017. [Online]. Available: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2741

[3] M. Silverio-Fernández, S. Renukappa, and S. Suresh, "What is a smart device?-a conceptualisation within the paradigm of the internet of things," *Visualization in Engineering*, vol. 6, no. 1, pp. 1–10, 2018, publisher: Springer.

[4] D. Marikyan, S. Papagiannidis, and E. Alamanos, "A systematic review of the smart home literature: A user perspective," *Technological Forecasting and Social Change*, vol. 138, pp. 139–154, 2019, publisher: Elsevier.

[5] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, "The smart home concept: our immediate future," in *2006 1st IEEE international conference on e-learning in industrial electronics*. IEEE, 2006, pp. 23–28.

[6] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003, publisher: IEEE.

[7] E. Houzé, J.-L. Dessalles, A. Diaconescu, D. Menga, and M. Schumann, "A Decentralized Explanatory System for Intelligent Cyber-Physical Systems," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2021, pp. 719–738.

[8] P. Lalanda, J. A. McCann, and A. Diaconescu, *Autonomic computing: principles, design and implementation*. Springer Science & Business Media, 2013.

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[10] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[11] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, and others, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020, publisher: Elsevier.

[12] T. Miller, P. Howe, and L. Sonenberg, "Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences," *arXiv preprint arXiv:1712.00547*, 2017.

[13] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[14] M. Blumreiter, J. Greenyer, F. J. C. Garcia, V. Klös, M. Schwammberger, C. Sommer, A. Vogelsang, and A. Wortmann, "Towards self-explainable cyber-physical systems," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019, pp. 543–548.

[15] K. Fadiga, E. Houzé, A. Diaconescu, and J.-L. Dessalles, "To do or not to do: finding causal relations in smart homes," in *Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. Online: IEEE, 2021, pp. 110–119, _eprint: 2105.10058.

[16] B. Kang, D. Kim, and H. Choo, "Internet of everything: A large-scale autonomic iot gateway," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 206–214, 2017.

[17] D. Burmeister, F. Burmann, and A. Schrader, "The smart object description language: modeling interaction capabilities for self-reflection," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, pp. 503–508.

[18] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.

[19] C. C. Aggarwal, *Outlier Analysis*. Springer International Publishing, 2017.

[20] P. Gärdenfors, *Conceptual spaces: The geometry of thought*. MIT press, 2004.

[21] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.