FORTNIoT: Intelligible Predictions to Improve User Understanding of Smart Home Behavior

SVEN COPPERS, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media
DAVY VANACKEN, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media
KRIS LUYTEN, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media

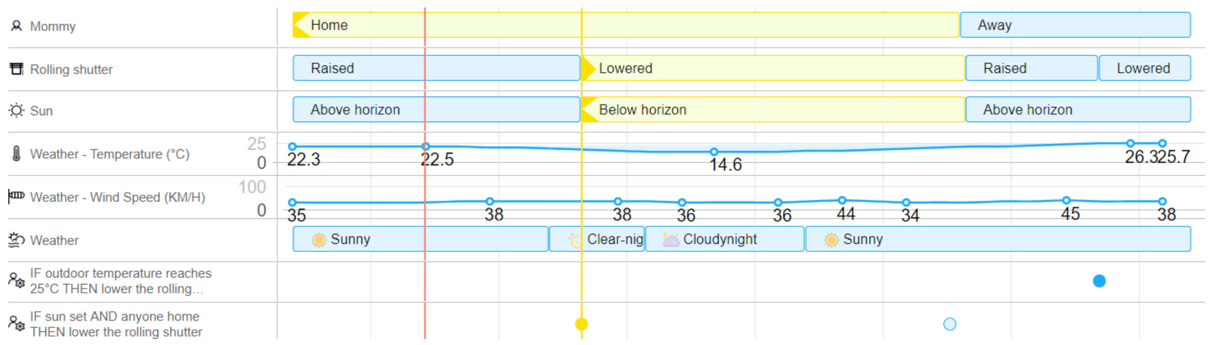


Fig. 1. Based on self-sustaining predictions (e.g. the sun will set), FORTNIoT can deduce when trigger-condition-action rules (e.g. `IF sun set AND anyone home THEN lower the rolling shutter`) will trigger in the near future and what effects they will cause (e.g. the rolling shutter will lower).

Ubiquitous environments, such as smart homes, are becoming more intelligent and autonomous. As a result, their behavior becomes harder to grasp and unintended behavior becomes more likely. Researchers have contributed tools to better understand and validate an environments' past behavior (e.g. logs, end-user debugging), and to prevent unintended behavior. There is, however, a lack of tools that help users understand the future behavior of such an environment. Information about the actions it will perform, and why it will perform them, remains concealed. In this paper, we contribute FORTNIoT, a well-defined approach that combines self-sustaining predictions (e.g. weather forecasts) and simulations of trigger-condition-action rules to deduce when these rules will trigger in the future and what state changes they will cause to connected smart home entities. We implemented a proof-of-concept of this approach, as well as a visual demonstrator that shows such predictions, including causes and effects, in an overview of a smart home's behavior. A between-subject evaluation with 42 participants indicates that FORTNIoT predictions lead to a more accurate understanding of the future behavior, more confidence in that understanding, and more appropriate trust in what the system will (not) do. We envision a wide variety of situations where predictions about the future are beneficial to inhabitants of smart homes, such as debugging unintended behavior and managing conflicts by exception, and hope to spark a new generation of intelligible tools for ubiquitous environments.

Authors' addresses: Sven Coppers, sven.coppers@uhasselt.be, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media, Diepenbeek, Belgium; Davy Vanacken, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media, Diepenbeek, Belgium, davy.vanacken@uhasselt.be; Kris Luyten, Hasselt University - tUL - Flanders Make, Expertise Centre for Digital Media, Diepenbeek, Belgium, kris.luyten@uhasselt.be.

2474-9567/2020/12-ART124 $15.00
https://doi.org/10.1145/3432225

CCS Concepts: • **Human-centered computing → Graphical user interfaces**; **Interaction paradigms**; *HCI theory, concepts and models*; *Interaction design theory, concepts and paradigms*; User interface toolkits.

Additional Key Words and Phrases: Intelligibility, Scrutability, Internet-of-Things, Smart Homes, Simulations, Predictions

ACM Reference Format:
Sven Coppers, Davy Vanacken, and Kris Luyten. 2020. FORTNIoT: Intelligible Predictions to Improve User Understanding of Smart Home Behavior. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 124 (December 2020), 24 pages. https://doi.org/10.1145/3432225

1 INTRODUCTION

Devices are becoming smarter [52], and in combination with online services they constitute networked applications often labeled Internet of Things (IoT) applications [1, 19, 30]. In the context of smart homes, such ubiquitous technologies are often meant to automatically respond to our presence and actions [28], for example for automation of day-to-day household actions, to achieve peace of mind [10, 49], or to optimise the usage of resources such as electricity [6, 25]. Besides major barriers such as high cost of ownership and a lack of flexibility, widespread adoption of home automation technology is slowed down by poor manageability [10, 28].

In domestic applications of the IoT, *entities* such as smart devices (e.g. light bulbs, smart sockets) and services (e.g. weather forecasts, social media) are often connected by a *middleware*. This is a platform that, among others, can host *trigger-condition-action rules* that are commonly used to automate smart home behavior [24, 33, 46], even by non-experienced users [64]. A *trigger* defines which event needs to occur before the condition is evaluated (e.g. `sun elevation changed`). An optional *condition* is a boolean expression that evaluates to true or false (e.g. `sun == below horizon`). An *action* defines what should happen when a rule is triggered and the condition is true (e.g. `turn on the light`). In this paper, we refer to the invocation of a rule in three stages: a rule is (1) *triggered* when an incoming event matches one of its triggers, (2) *evaluated* when its boolean expression is assessed, and (3) *executed* when its actions are performed (which implies that the rule was triggered and evaluated to true).

Despite the simplicity of trigger-condition-action rules, non-technical inhabitants have difficulties in understanding their smart home [36, 38, 43] and may not be willing to invest time in learning this [10, 46, 73]. As the number of rules grows, it becomes increasingly hard to understand how the smart home is behaving [11, 28, 37, 69, 70], users frequently misinterpret the behavior (e.g. they assume that rules are executed when their condition evaluates to true, without getting triggered first), and are prone to introduce errors [10, 30, 35]. All this can cause unintentional actions and security vulnerabilities [70], such as unlocking a door at the wrong moment [19, 28, 46], and results in a lack of trust of the user.

To counter such challenges, previous work focused, among others, on explaining a system's actions while they happen [44, 67, 68] and on debugging unwanted behavior [3, 19, 46]. These tools, however, mainly support debugging past behavior [38], or require users to commit to a solution when composing the rules (i.e. *at rule creation time*). While informing users about future events is already proven to be useful [17, 26, 62], no tools currently exist to help users anticipate future actions of their smart home and unexpected system actions. To address this gap and help users understand what their smart home will do in the future and why [7, 28, 42], we present three main contributions in this paper:

(1) FORTNIoT, a well-defined approach that uses predictions (e.g. weather forecasts) to simulate when trigger-condition-action rules will be executed in the future and to deduce what state changes these will cause to other connected entities (Section 3);
(2) a visual demonstrator that presents these predictions in an intelligible overview of smart home behavior, without requiring inhabitants to understand how trigger-condition-action rules work (Section 4);
(3) and a between-subject study with 42 participants to evaluate the impact of the predictions on the user's understanding of, and trust in, the smart home behavior (Section 5 and 6).

The remainder of this paper is organised as follows: we first highlight the importance of intelligibility and describe the state of the art regarding trigger-condition-action rules (Section 2). We then present the core principles of FORTNIoT predictions (Section 3) and a visual demonstrator that includes these predictions in a graphical user interface (Section 4). Next, we describe our methodology to clarify the impact of FORTNIoT predictions on the user experience (Section 5) and report on both quantitative and qualitative results from our study (Section 6). Finally, we discuss the key findings and opportunities for future work (Section 7).

2 RELATED WORK

In this section, we highlight the importance of intelligibility, provide an overview of academic visualizations and debuggers for the IoT, and describe earlier research regarding simulations and semantics.

2.1 Intelligibility

Despite their apparent simplicity, composing trigger-condition-action rules is often a complex task for non-programmers [36]. They lack knowledge about key concepts such as boolean expressions and operators [46], and have difficulties in correctly understanding the distinction between events and conditions [30]. Changing user requirements also cause smart homes to be reconfigured over time, for instance by adding entities (e.g. devices or services) or editing rules [46]. As the number of rules grows, it becomes increasingly hard to fully grasp their behavior [28, 37], and as the behavior is often automated, it can leave users feeling out of control [5], especially when appropriate feedback is lacking [56]. To asses whether the environment is safe, secure, and operating as intended at all times, users must understand what it is doing [38, 71]. Having a limited understanding can be frustrating [5, 41], and can cause users to lose trust [2, 53]. More trust is, however, not always better, since users might trust the system when it is not behaving as intended [72]. It is crucial to help users to ensure that the system they created meets their expectation [11].

Intelligibility and scrutability have been identified as crucial properties to help users build a mental model about how the environment is behaving [4, 7, 45]. Amongst others, intelligibility can help users understand system states [38], and when and why automatic actions are performed [44, 67]. There are a few concerns that developers of intelligible interfaces need to keep in mind: users are generally not interested in learning how the underlying technology works [73], can easily be overwhelmed by large amounts of well-intended information [38, 51], and their needs for intelligibility shifted over time from in-depth awareness to management by exception [38].

In the context of intelligibility in context-aware applications, Lim et al. describe different types of explanations, including *why*, *what*, and *what if* [42]. They also put forward guidelines for explanations, such as providing explanations automatically and supporting combinations of explanations [42, 57]. Edwards et al. translated these guidelines to the context of smart homes and argue that inhabitants need to know what devices are doing and how they can be controlled [28], but Jakobi et al. found that existing systems typically have limited self-explication capabilities, which are often restricted to showing a list of triggered rules or sensed events in the past [38]. No systems currently exist that explain how a smart home will behave in the near future.

2.2 Visualizations and Interaction

The common method to improve intelligibility is visualizing additional information in a meaningful way. In the context of smart homes, however, the huge variety of entities, and thus information, needs to be considered carefully to avoid overwhelming the user to the point where she simply decides to ignore the available information [51]. If not handled mindfully, the solution of presenting additional information might actually become worse than the problem of lacking intelligibility, as it comes with the trade-off of taking more time to interpret [14, 18]. It is also interesting to note that interactive explanations can be more effective than static explanations [14].

Jakobi et al. recommend explaining behavior by relating it to routines [38]. With *Casalendar*, Mennicken et al. explored this approach by integrating past and future entity states and system events into a shared calendar [51]. In their evaluation, users expressed a lack of relevant information about the future, such as weather forecasts or the weekly cleaning schedule of the robot vacuum, as Casalendar included dummy data instead of actual predictions. Others focused on monitoring and explaining actions, while they are being performed by the system, for instance in traditional GUIs [39], virtual reality [29] and augmented reality [47, 65, 66]. Castelli et al.'s tool combines predefined visualizations and a visualization creation tool to inspect logged smart home data [13]. Such tool could be useful to visualize predicted smart home data as well. Visualizations developed to enhance the intelligibility can often also be used to debug (unintended) behavior [22, 48]. In summary, previous efforts aim to visualize interconnected behavior in real-time or in retrospect, and no tools currently exist to visualize or explain the future behavior of smart homes.

2.3 Simulations and Semantics

An effective solution to ensure interoperability between entities is using semantics [27, 55], which describe the capabilities of entities and the meaning of their connections [69]. To this end, the W3C created the Semantic Sensor Networks ontology, which is currently used in a number of research projects [16]. By means of semantic descriptions, virtual counterparts for real entities, connections and calculations can be created [55, 60]. Besides quick deployment [54] and a decreased dependency on hardware during development [59], virtual counterparts enable possibilities such as debugging virtual situations and running simulations. While emulators and simulators are useful for basic testing, they are often insufficient or infeasible (e.g. no simulator available, or too many dependencies) and must be complemented by tests on real devices [59].

Besides virtual entities, smart home behavior can be simulated by using models such as hybrid automata [11]. AppsGate, for example, can run programs using virtual data and time [21]. Similarly, the simulators of SAPIENS [61] and the TARE editors [30] can validate which rules would be triggered in a simulated context entered by the user. A newer version of the latter tool can also detect conflicts based on these simulations [46]. While such simulations are useful to understand if and how a set of rules will work in a specific test context, it remains cumbersome to validate whether the rules will work in all contexts. Corno et al.'s EUDebug is capable of detecting the same conflicts, but in contrast to the TARE editor, this tool uses hybrid semantic colored Petri nets to simulate rules and their interactions, without requiring the user to define a test context [19, 20]. In summary, previous work has used simulators to help users understand how a set of rules will behave in a user-defined context, and how rules can conflict in general. Using simulators to generate the outcomes of rules in the context of real entities after those rules have been deployed, however, remains unexplored.

3 FORTNIOT: DEDUCING PREDICTIONS OF SMART HOME BEHAVIOR

Although substantial efforts have been made to accurately model the holistic behavior of a smart home [11, 70], its (incomplete) behavior is typically described as a set of trigger-condition-action rules [24, 36, 38, 46]. Reasoning about such a set of rules requires a substantial mental effort from users and depends on many external factors, such as user input and data generated by sensors. In this section, we introduce FORTNIoT, an approach that reuses these rules to predict future smart home behavior to help users understand how their smart home will behave in the future. Compared to reasoning about an abstract set of rules, it is much easier to interpret concrete predictions that demonstrate the behavior of the smart home.

The core principle behind FORTNIoT is presented in Figure 2: by simulating trigger-condition-actions rules on self-sustaining predictions, we can deduce additional predictions about the future of the smart home. In the context of Figure 1, for example, FORTNIoT uses self-sustaining predictions for the sun (e.g. it will set during the evening), and three trigger-condition-action rules (e.g. `IF sun set and anyone home THEN lower the rolling`

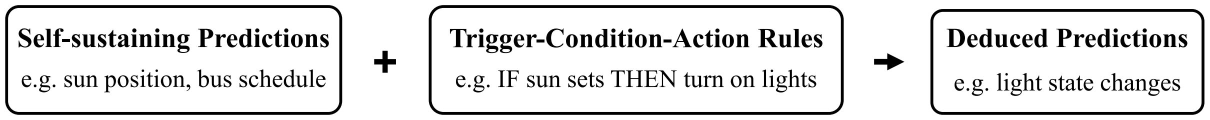

Self-sustaining Predictions
e.g. sun position, bus schedule
+
Trigger-Condition-Action Rules
e.g. IF sun sets THEN turn on lights
Deduced Predictions
e.g. light state changes

Fig. 2. The core principle of FORTNIoT. Based on self-sustaining predictions and trigger-condition-action rules, additional predictions about the future can be deduced.

`shutter`). By simulating these rules using the predictions for the sun, FORTNIoT can deduce that the rolling shutter will change state during the evening (e.g. it will be lowered).

3.1 Self-sustaining Predictions

Predictions such as weather forecasts, the position of the sun, or real-time public transit information are already a commodity. We can adapt information that is already available in existing IoT middleware to basic predictions about the future (e.g. the sun will set at 21:30). These predictions stand on their own and thus *sustain themselves.* In this overview, we discuss their most important properties.

Type of predictions. We define two types of predictions: *fixed* and *continuous.* Based on the available information, fixed predictions about certain entities can be made (e.g. tomorrow the sun will rise at 9:30). These predictions are about specific moments in the future, and are not affected by rule executions or other entities. Such predictions can be done once, at the start of the simulation. By contrast, continuous predictions are context-sensitive and can be affected by other rule executions or entity changes (e.g. the room temperature will increase if the heating is on, and will cool down when the air conditioning is turned on). The prediction of their state at time t_n depends on the predicted context at time t_{n-1}. As a consequence, their states need to be updated throughout the simulation, with a minimum *interval.* The smaller this interval, the more accurate the predictions (e.g. less overshooting).

Predictability and accuracy. Some self-sustaining predictions are more *predictable* and thus more certain to happen, whereas others are nearly impossible to predict. Predictions based on user interaction, for example, are highly uncertain. Using pattern recognition on logged events can mitigate this uncertainty to some degree, but will never reach complete accuracy in most cases. Furthermore, predictions about the near future tend to be more *accurate* compared to predictions about the distant future: when context information changes, predictions might no longer be correct. For example, a user's estimated time of arrival might shift when a traffic accident occurs, so any rules that trigger when the user arrives at home need to be predicted again. We mitigate this problem by updating predictions when the context changes [51]. The predictability and accuracy of self-sustaining predictions is inherited by other predictions that are deduced from them. In this paper, we assume that self-sustaining predictions are complete and accurate, and we discuss the implications in Section 7.

Availability of existing services. Can an external service already predict certain information (e.g. transit information, weather forecasts)? If yes, users probably have access to it (e.g. through a website, or an app). To avoid information scattering, these predictions can be reused and integrated into a single overview. If not, an internal model can be used to approximate the behavior of an entity. Depending on the desired accuracy and the complexity of the behavior of the entity at hand, the most appropriate type of model must be selected. We can use, for instance, heuristic models (e.g. a linear model for battery depletion rate), a mathematical formula (e.g. to predict the elevation of the sun at a specific moment in time based on the coordinates of the observer), or a machine learning model (e.g. pattern recognition for a user's preferred temperature throughout a week). As smart homes embed a heterogeneous set of entities, so can a heterogeneous set of models be used to approximate each of their respective behaviors.

3.2 Simulating Trigger-Condition-Action Rules

Besides self-sustaining predictions, FORTNIoT requires the trigger-condition-action rules that determine the behavior of a smart home. Although some behavior might be built into an entity by its manufacturer (e.g. unlock fail safe locks when the power is lost), we assume that such behavior can be modeled with rules as well.

Similar to Home Assistant, we assume all trigger-condition-action rules consist of three parts [33]. The *trigger* of a rule defines which event needs to occur before the condition is evaluated (e.g. `button press`, `indoor temperature change`, or `location update`). The *condition* is a boolean expression that evaluates to true or false, depending on whether or not its constraints about the entities are satisfied (e.g. `sun == below horizon`, `temperature < 20 degrees`, or `number of people at home > 0`). Rules usually contain a trigger for the entity that is referred to in the condition (e.g. when the condition is `IF sun below horizon`, there is a trigger `sun position changed`). When multiple entities are referred to in the condition (e.g. when comparing two entities, or in AND conditions), rules usually contain triggers for each entity (e.g. in `IF sun below horizon AND somebody home`, there are triggers for `sun position changed` and `person location updated`). Only one incoming event at a given moment needs to match one of the triggers to cause the entire condition to be re-evaluated, but only when all constraints of the condition hold, the actions are executed. An *action* defines what should happen when the rule is triggered and the condition evaluates to true. An action can be a request to change an entity's state (e.g. turn on the living spots) or a service call (e.g. post status update on social media). One rule can include multiple actions.

Similar to other tools that analyse existing behavioral models [17, 19, 46, 61], FORTNIoT reuses those models by running them in a sandbox. To use trigger-condition-action rules for predictions, their components need to be decoupled and adapted, so rules can be triggered, evaluated and executed in a simulation mode. First, triggers need to be able to detect 'virtual' events for simulation purposes. Second, a condition needs to be able to evaluate whether its constraints hold, based on a snapshot of all entity states at a specified moment in time. Last, the actions of the rule cannot be executed directly on connected entities, but instead need be saved virtually as part of the prediction process.

3.3 Deducing Predictions

Figure 3 presents the algorithm to deduce predictions about the future, by combining model-based predictions and simulations of trigger-condition-action rules. The algorithm uses three main data structures: queue is a chronologically ascending list that keeps track of all entity state changes that still need to be considered by the algorithm, snapshot is a collection of all entity states at a specified moment in time, and future saves all predictions about rules and which state changes their actions will cause.

Initialisation. The algorithm starts by initialising an empty queue and future, and then predicts fixed future states using the adapters of each entity (e.g. the sun entity knows when the sun will set, the transit information entity knows the future bus arrivals). All fixed self-sustaining predictions are collected in the queue. The algorithm also takes a snapshot with all current entity states as a starting point. To maintain performance and accuracy, the algorithm only runs until a predefined moment in the future (the *end time* in Figure 3).

Updating self-sustaining predictions. Since simulating every second of the future would be too time consuming, the algorithm only simulates a discrete set of future moments. As next moment to be simulated, the algorithm uses the time of the next state change in the queue, unless a minimum *tick time* is reached. This tick time assures regular updates of the continuous predictions. At each *tick* (i.e. each moment that needs to be simulated, Figure 4), all self-sustaining predictions are updated and more predictions can be deduced.

Deducing predictions. To deduce predictions, states are popped from the queue and passed to a rule manager that checks which rules will be triggered. Each of these rules then evaluates its own condition, based on the latest snapshot. The algorithm yields virtual states, based on the actions of all rules that evaluate to true. The newly

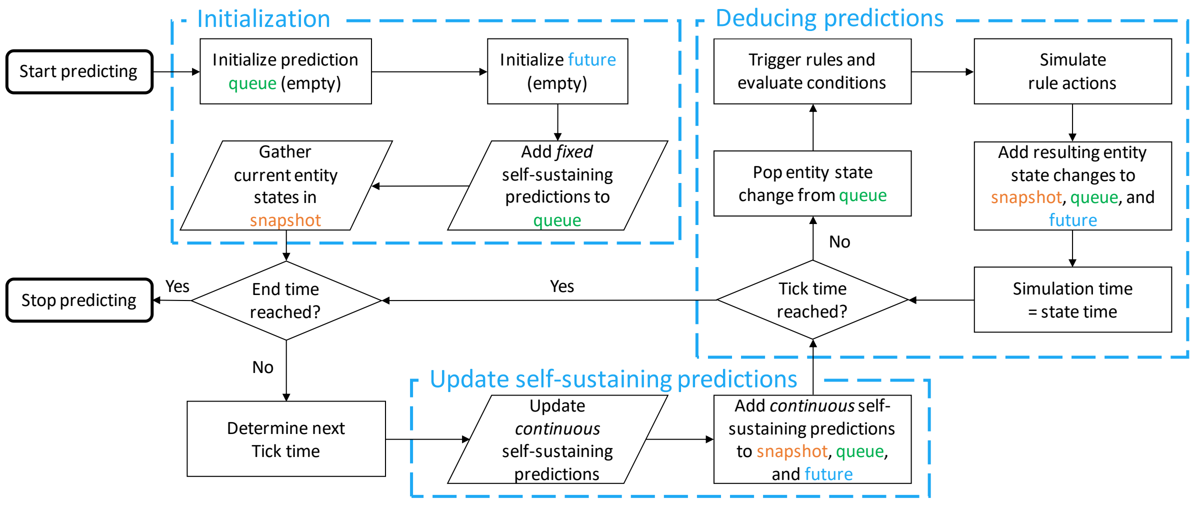

Initialization
Deducing predictions
Update self-sustaining predictions
Start predicting
Initialize prediction queue (empty)
Initialize future (empty)
Add *fixed* self-sustaining predictions to queue
Gather current entity states in snapshot
Trigger rules and evaluate conditions
Simulate rule actions
Add resulting entity state changes to snapshot, queue, and future
Simulation time = state time
Tick time reached?
Pop entity state change from queue
No
Yes
End time reached?
Yes
Stop predicting
No
Determine next Tick time
Update *continuous* self-sustaining predictions
Add *continuous* self-sustaining predictions to snapshot, queue, and future

Fig. 3. The prediction algorithm in FORTNIoT uses self-sustaining predictions and simulations of trigger-condition-action rules to deduce which rules will be executed in the future and which state changes they will cause to the connected entities.

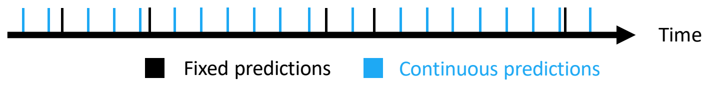


Fig. 4. Every vertical line is a *tick* or moment in time that needs to be simulated. In contrast to fixed predictions that will happen at a specific moment (e.g. the sun will set), continuous predictions require to be updated at a minimum interval (e.g. indoor temperature throughout the day).

predicted states are added to the future predictions, as well as to the queue and snapshot, as each of these new states can result in other rules being triggered. If the queue contains multiple states that happen exactly at the same time (e.g. when a rule caused two actions), the algorithm processes each of these states in no particular order before moving on to the next tick, until the end time is reached.

Tracking causes and effects. Besides predicting what state changes will happen to the entities in the future, it is also important to explain why these changes will happen [46]. To keep track of causes and effects, the algorithm uses a separate data structure that stores the simulation time and the rule that was executed, as well as the IDs of causing states and resulting states.

3.4 Proof-of-Concept Implementation of FORTNIoT

In order to perform an evaluation that is representative for the capabilities of FORTNIoT and to facilitate future research, we implemented the algorithm described in the previous section on top of Home Assistant [32], an open source middleware that integrates many kinds of IoT entities. The architecture is presented in Figure 5 and consists of three major components that correspond with the main principles of FORTNIoT: the `Rule Manager`, the `Adapter Manager`, and the `Prediction Engine`.

The `Rule Manager` keeps track of all trigger-condition-action rules, both for user-defined rules and rules that replicate the behavior embedded into some devices by the manufacturer. We opted to not use the rule manager

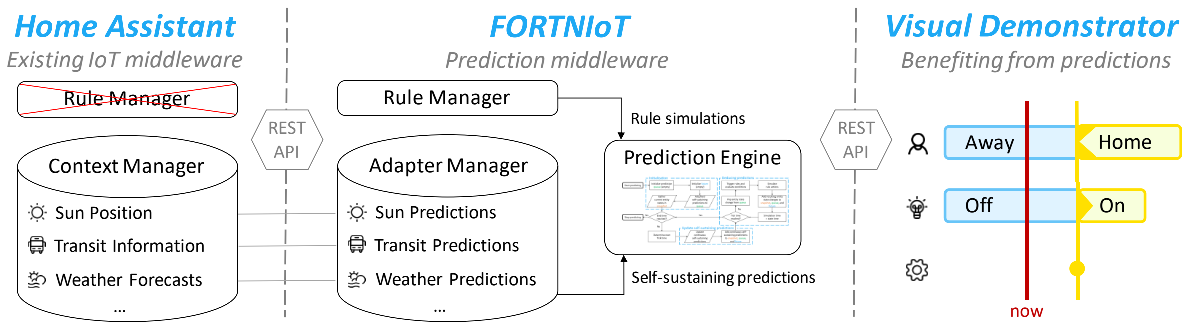


Fig. 5. Our proof-of-concept implementation is built on top of Home Assistant [32], an existing middleware to manage IoT entities in a smart home. We implement a custom rule manager to allow the simulation of trigger-condition-action rules.

of Home Assistant because it lacks the required "simulation mode", and implemented our own `Rule Manager` instead. In addition, our implementation of the `Rule Manager` keeps track of *why* the condition of a rule evaluates to true and which entity states result from the actions of the rule, for the purpose of intelligibility.

The `Adapter Manager` keeps track of all adapters that collect information from Home Assistant to generate new self-sustaining predictions. These predictions can be supplied in two distinct ways, depending on the type of prediction. Adapters can provide *fixed* predictions once at the start of the simulation (e.g. at 14:35 it will start to rain). Alternatively, at the start of every tick, an update method is called to give every adapter the chance to provide a *continuous* update of the future states of the entity it represents, based on the states of all entities at the end of the previous tick (e.g. the room temperature will decrease by 0.1 °C because the air conditioning turned on at the end of the previous tick).

The `Prediction Engine` accesses both aforementioned managers to implement the prediction algorithm presented in the previous section. The algorithm is executed for every incoming event or context change to prevent the predictions from becoming outdated [50].

Currently, only the IoT middleware of Home Assistant is used, but other middleware can be supported by implementing new adapters that collect information and use it to generate new self-sustaining predictions. Similar to the current implementation, the trigger-condition-action rules need to be imported or replicated in the `Rule Manager` of FORTNIoT. The `Predication Engine` does not require any changes and exposes all predictions by means of a RESTful API (Figure 5), which can be accessed by any user interface that requires predictions, such as our visual demonstrator, for example.

4 VISUAL DEMONSTRATOR

To demonstrate how FORTNIoT predictions can be used, we designed and implemented an overview that embeds FORTNIoT predictions into an interactive timeline. Such an interface can be a powerful tool for users who need to understand what will happen in their smart home and why, without the need for knowledge about the inner workings of the system [10, 28]. Since intelligible systems should explain *what* they will do and *why* [28, 42] to help users understand whether the system meets their expectations [11], and because current systems do not explain how a smart home will behave in the near future [38], we decided on two main goals. The first goal, ***G1.*** *Display what will happen to entities in the future*, helps inhabitants to answer questions such as "What are their predicted future states?", "When will they change state?" and "How long will they remain in that state?". The second goal, ***G2.*** *Display why the entities will change*, helps inhabitants to understand which rules will cause these changes, why those rules will be triggered, and why the conditions of those rules are satisfied.

To accomplish these goals, we used an incremental and iterative design process: we started from the history visualization inside the dashboard of Home Assistant [34], which includes horizontal rows with blocks that represent historical states for each connected entity. We extended it by also including predicted states, by incorporating information about when rules will be executed, and by highlighting causes and effects. We only added one feature at a time and gathered feedback from two HCI researchers who were not involved in this project during every iteration. In the remainder of this section, we highlight our main design choices and rationale.

Entities and Changes. Our visualization in Figure 6 is based on a timeline, since time is generally considered a fitting structure for understanding smart home behavior [38, 50]. It closely resembles the history view of Home Assistant itself [34]. In contrast to Mennicken et al. [50], who integrated information about the system's state in a calendar, we opted to combine all information (including calendar data) in a timeline. The rows at the top each depict an entity, such as the sun or the living room spots. Depending on the resolution of the display, the visualization scales to around 15-20 rows of entities and rules that are visible without the need to scroll. We envision this to be enough for deployments in real smart homes, as rows can be filtered dynamically by relevance, for example based on the time or the user's location. The blue bars inside a row depict the states of the entity, so it is easy to discover upcoming changes to entity states. The zoom level and viewport of the time can be manipulated by scrolling and panning respectively. A vertical red line | denotes the current time and moves slowly to the right as time passes. The visualization seamlessly combines logged information with predictions from FORTNIoT. The visualization meets goal **G1** by making it easy to view the holistic state of the smart home at any point in time, which is otherwise difficult to grasp [38].

Rules and Executions. We add rows to the visualization for each individual trigger-condition-action rule to provide a clear overview of when they will be executed. These rows are grouped together at the bottom. Two types of rules are supported: rules that are provided by the user (e.g. `IF sun rises THEN turn off the living spots`) and rules that model the behavior built into an entity by the manufacturer (e.g. `IF power loss THEN unlock the door`). The blue dots inside a trigger-condition-action row represent the moments when the corresponding rule will be executed: solid dots ● mean that the actions of the rule will be executed and that those actions affect the state of some entities (e.g. the living room spots will turn off), while empty dots ○ mean that the actions of the rule will not cause any changes to entity states (e.g. as the living room spots will already be turned on, no actual changes happen when the sun sets). While empty dots do not contribute to the overall behavior of the smart home, we opted to include them to avoid confusion about absent rule executions.

Causes and Effects. To visually explain causes and effects and meet goal **G2**, we rely on the visualization principle of providing an overview first and details on demand [63]: by selecting a future state of an entity ▢, the user activates a visualization that explains how and why this state will come to be (Figure 7). To avoid visual clutter by adding extra graphical elements, the visualization instead reuses existing graphical elements

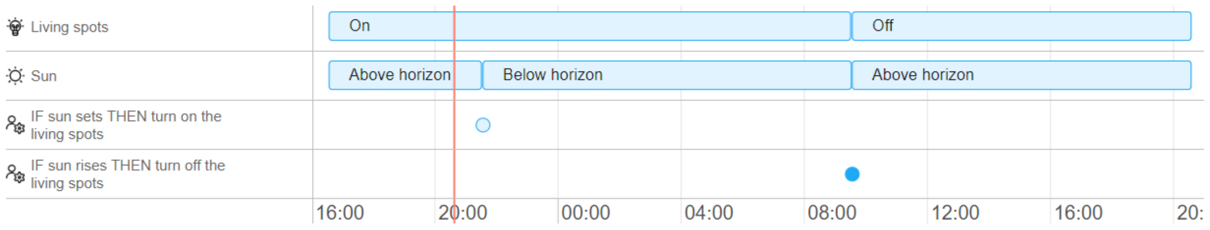


Fig. 6. The visualization seamlessly combines logged information (before the red line, e.g. the living room spots are on) and predictions from FORTNIoT (after the the red line, e.g. the spots will turn off around 9:30).

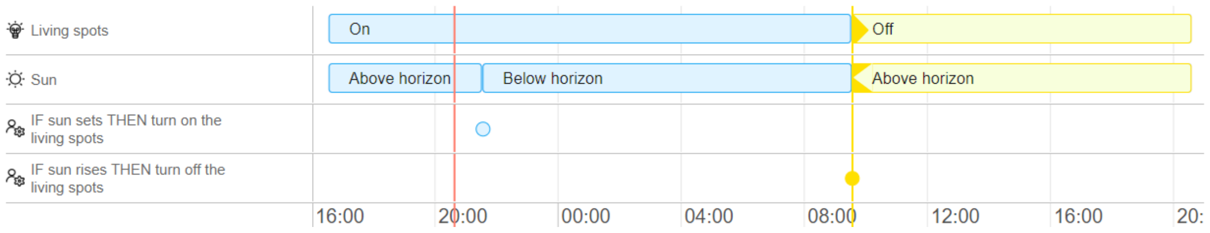


Fig. 7. When selecting an entity state in the future, the rule that will cause this state is highlighted, as well as the states that cause the condition of this rule to evaluate to true.

by highlighting all related entity states ▢ and the corresponding rule execution ● in yellow. Arrows inside the highlighted bars distinguish between causes and effects. The selected state, as well as any other states that represent consequences from executing the actions of this rule, are depicted with an arrow to the right ▶ (i.e. the effects). The states that will contribute to satisfying the condition of the rule, and thus will cause its execution, are depicted with an arrow to the left ◀ (i.e. the causes). The visualization assumes that all entities that trigger the rule also occur in its condition and does not visually distinguish them. All highlighted elements are connected by means of a yellow vertical line |, which also pinpoints the time at which the rule execution will be triggered.

Only a single mapping from causes to their effects is shown at a time (e.g. a single rule execution). When a causing state of one rule is also the effect of another rule, the user can click it to activate a new WHY-visualization for that state. These steps can be repeated to navigate through the causality chain until the cause of a state is unknown by the system (e.g. the system does not explain why the sun's position changes throughout the day).

5 STUDY DESIGN

We performed a between-subject study to evaluate the impact of FORTNIoT predictions on the user's understanding of, and trust in, the smart home behavior and to evaluate whether the visual demonstrator reaches the goals for which it was designed. Based on the design rationale listed in Section 4, we postulated three hypotheses:

H1. *FORTNIoT predictions lead to a* ***more accurate understanding*** *of the smart home behavior.* In particular, we expect participants to better understand *what* actions will happen and *why* those actions will happen.

H2. *FORTNIoT predictions lead to a* ***more appropriate trust*** *in the smart home behavior.* We expect participants to have *reduced undertrust* (i.e. they trust the system more when it is behaving correctly) and *reduced overtrust* (i.e. they trust the system less when it is behaving incorrectly).

H3. *FORTNIoT predictions lead to* ***more confidence*** *in participants' understanding of the smart home behavior.*

5.1 Experimental Design and Measurements

To reduce the impact of design as a variable, our between-subject study compares the visual demonstrator of Section 4 (the FORTNIoT condition, shown in Figure 8) to a version of the same interface that does not include the predictions generated by FORTNIoT (the baseline condition, shown in Figure 9). The baseline interface offers all information currently found in state-of-the-art tools (e.g. overview of entities and rules, and self-sustaining predictions such as calendar information), and even improves upon state-of-the-art tools by integrating all those pieces of information into one coherent visualization.

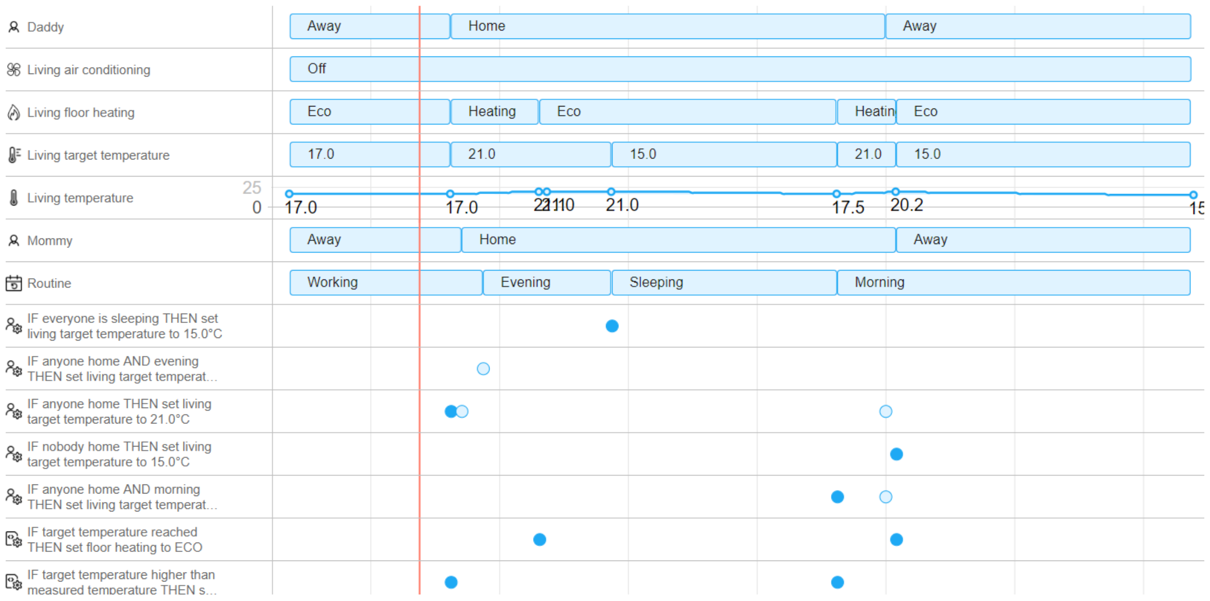

Daddy
Away
Home
Away
Living air conditioning
Off
Living floor heating
Eco
Heating
Eco
Heatin
Eco
Living target temperature
17.0
21.0
15.0
21.0
15.0
Living temperature
25
0
17.0
17.0
21.0
21.0
17.5
20.2
15
Mommy
Away
Home
Away
Routine
Working
Evening
Sleeping
Morning
IF everyone is sleeping THEN set living target temperature to 15.0°C
IF anyone home AND evening THEN set living target temperat...
IF anyone home THEN set living target temperature to 21.0°C
IF nobody home THEN set living target temperature to 15.0°C
IF anyone home AND morning THEN set living target temperat...
IF target temperature reached THEN set floor heating to ECO
IF target temperature higher than measured temperature THEN s...

Fig. 8. The FORTNIoT condition includes self-sustaining predictions from external sources (e.g. dad will come home), as well as when and why the rules will be executed and which actions they will cause (e.g. the heating will turn on).

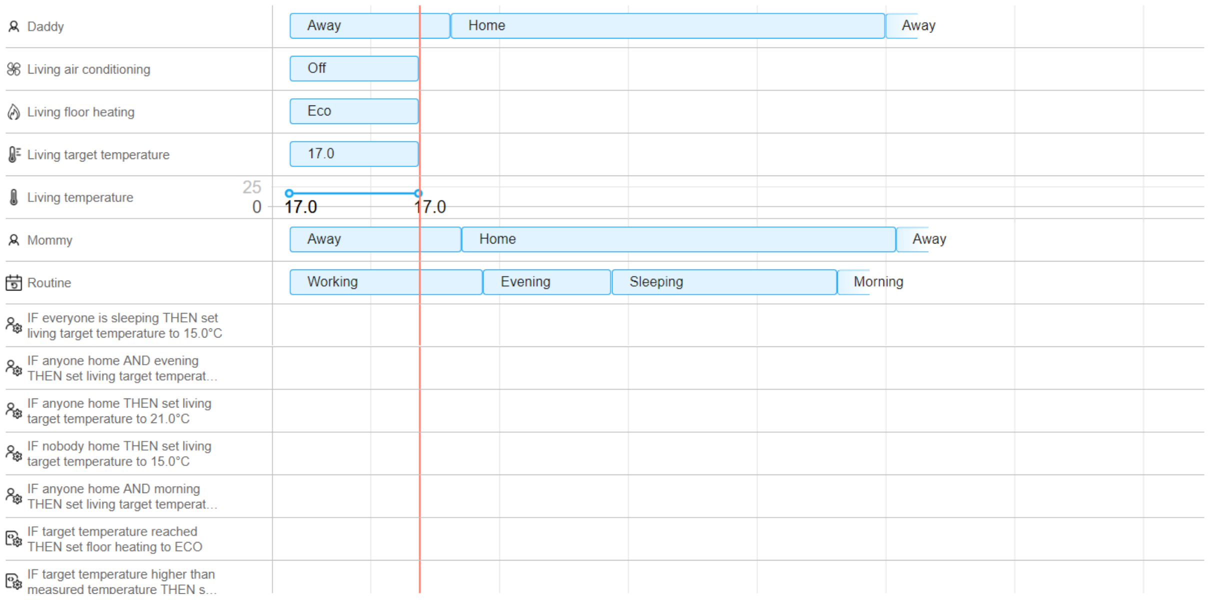

Daddy
Away
Home
Away
Living air conditioning
Off
Living floor heating
Eco
Living target temperature
17.0
Living temperature
25
0
17.0
17.0
Mommy
Away
Home
Away
Routine
Working
Evening
Sleeping
Morning
IF everyone is sleeping THEN set living target temperature to 15.0°C
IF anyone home AND evening THEN set living target temperat...
IF anyone home THEN set living target temperature to 21.0°C
IF nobody home THEN set living target temperature to 15.0°C
IF anyone home AND morning THEN set living target temperat...
IF target temperature reached THEN set floor heating to ECO
IF target temperature higher than measured temperature THEN s...

Fig. 9. The baseline condition only includes self-sustaining predictions from external sources (e.g. dad will come home) and a list of all the rules, but does not predict which rules will be executed or which actions will be performed.

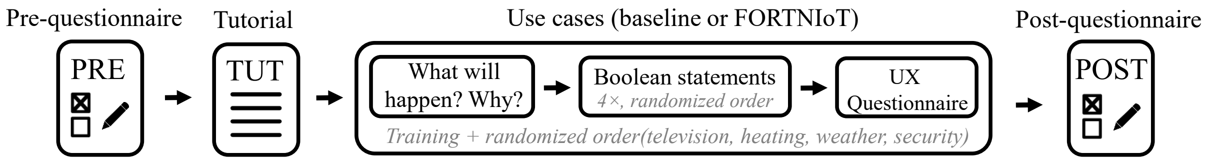


Fig. 10. The procedure of the between-subject study, with a baseline and FORTNIoT condition.

Due to the COVID-19 pandemic [58], we shifted our study to one-on-one remote video calls between the participant and facilitator. Each participant was asked to share her/his screen to ensure that the experiment was being executed appropriately and consistently, and we had the participant's undivided attention [15]. The procedure of the study is outlined in Figure 10: after an informed consent, participants filled in a *pre-questionnaire* about their demographics and experience. During the *tutorial*, participants read a single-page manual about the basic visualization of entities and trigger-condition-action rules. Participants using FORTNIoT had to read an extra page about the additional elements (e.g. the visualization of triggered rules, and causes and effects).

To compare understanding and trust between conditions, participants answered questions related to smart home scenarios presented in five use cases. The first use case was always a simplistic `training` scenario with three entities and rules. The following four use cases (Table 1) were presented in random order. The use cases were created by replicating the entities in our direct environment and by organising them by functionality (`television`, `temperature`, `weather`, and `security`), which resulted in $7-11$ entities per use case. We aimed to automate the behavior of these entities in a realistic but non-trivial manner by adding at least 5 relevant trigger-condition-action rules. During the evaluation, we loaded the set of virtual entities and rules into FORTNIoT, and prepopulated the logs and self-sustaining predictions with fixed entity states to ensure consistent smart home behavior across all participants. Only for participants in the FORTNIoT condition, rule executions were simulated by FORTNIoT to deduce additional consequences. Participants could explore all information using the visual demonstrator. Screenshots of the demonstrator for all use cases in both conditions are available in our supplementary material. We did not provide participants any additional information, such as descriptions of the smart home, to drive the need for exploration and to validate whether our visualizations are truly intelligible.

To measure understanding, we asked participants to think aloud and report *what* will happen in a scenario and *why*. Participants were asked to notify the facilitator when they believed to have found all answers (often not all future entity changes were actually found). Next, participants expressed the confidence in their answers on a 5-point Likert scale. Exploring the visual demonstrator helped participants to build a mental model of connected entities and rules, which was required for the next steps in the experiment.

To measure appropriate trust [72], participants had to validate four statements in slight variations of the use cases. These variations used the same entities and rules, but different entity states and self-sustaining predictions. In other words, the variations of the smart homes behaved identically, but in different contexts. Each statement was always combined with the same variation of the use case, and the order of the statements was random. To avoid bias, participants were not told that half of the statements were true (Table 3), and the others were false (Table 4). For each statement, participants again expressed their confidence on a 5-point Likert scale.

We measured the time that participants needed to decide on their final answer for completeness, and decided against imposing any time limit. The facilitator continuously monitored the screen and took notes about the participants' think-aloud reasoning. At the end of each use case, participants completed a short questionnaire about the task difficulty and their perceived understanding. In a final *post-questionnaire*, we queried participants about their strategy to find answers, and they completed the System Causality Scale [31], which is a more specialized version of the generic System Usability Scale [9]. It aims to quickly measure the usefulness of (visual) explanations and to what extent they are suitable for explaining a system.

Table 1. The complexity of the use cases, measured by the number of rules, number of times those rules are executed, number of entities, and number of times entities will change state. Some rule executions do not cause any entity change, when the entity already has the state specified by the actions of the rule.

Use case	# Rules	# Rule executions	# Entities	# Entity changes
`television`	6	6	8	4
`temperature`	7	12	7	8
`weather`	7	7	11	7
`security`	5	5	9	4

5.2 Participants

Via social media, we recruited 42 participants with varying backgrounds (6 business, 2 communications, 6 computer science, 3 education, 6 engineering, 3 government, 4 legal, 1 logistics, 3 health, 3 research, 4 students, and 1 unemployed). No prior knowledge about coding, trigger-condition-action programming or IoT was required. Although 7 participants were experienced to very experienced with coding, only 2 were experienced to very experienced with trigger-condition-action programming or IoT. We randomly assigned each participant to one of two groups for our between-subject experiment. The first group had 10 female and 11 male participants (11 aged 21-29, 4 aged 30-39, 5 aged 50-59, and 1 aged 60 or older). The second group had 13 female and 8 male participants (12 aged 21-29, 5 aged 30-39, 3 aged 40-49, and 1 aged 50-59). Each participant received a voucher of €10 as a token of gratitude for participating in the study, which took 90 minutes on average.

6 ANALYSIS AND RESULTS

For binomial data (e.g. whether or not participants found an entity change, or whether they thought a statement was true or false), we used Fisher's exact test to determine statistical significance, since our sample size is small (i.e. for some data points smaller than ten). Given the exploratory nature of our study, we applied the Benjamini-Hochberg procedure to control the False Discovery Rate [8]. We used Mann-Whitney U tests to evaluate differences for Likert-scale ratings about the user experience and confidence. Written qualitative remarks were grouped by the facilitator using thematic analysis.

6.1 Awareness and Understanding

When asked *what is going to happen* in the smart home, baseline participants missed 118 entity changes (out of 21 participants × 23 entity changes per participant = 483 entity changes in total). In contrast, FORTNIoT participants were significantly more aware and missed only 7 entity changes ($N = 966, p < 0.001, odds\ ratio = 21.9$). Those participants also understood significantly more often *why* changes will happen ($N = 966, p < 0.001, odds\ ratio = 7.5$). A breakdown of all entity changes is presented in Table 2.

Across all use cases, six baseline participants mentioned that the amount of entities and rules made it '*hard to keep an overview*", and eight commented that it was "*easy to miss something*. Moreover, multiple participants remained unsure about details such as "*does cloudy weather mean it is going to rain?*" (six participants), '*does the room cool down when the floor heating is in ECO mode?*", and "*is this a movie or TV show?*" (P10). Such details are important, because wrong assumptions can lead to incorrect expectations about the smart home behavior. Some participants found the antecedence between rules to be unclear, and incorrectly assumed that some rules were "*more important than others*" (P14), or "*override each other*" (P05, P28 and P37).

In the `television` case, 71.4% of the baseline participants missed `E3`. We believe this was caused by some participants considering each rule only once, while in fact the `IF news and somebody is home THEN turn on TV`

Table 2. The number of participants who found the entity state change and understood why it will happen. Significance determined using Fisher's exact test with Benjamini-Hochberg corrections (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

Use case		Entity state change	Found (%)	Why (%)
`television`	E1	TV turns on when the evening news starts	81.0 / 100.0	81.0 / 100.0
	E2	TV turns off when sleeping	71.4 / 100.0	71.4 / 100.0
	E3	TV turns on when the morning news starts	28.6 / 100.0 ***	28.6 / 100.0 ***
	E4	TV turns off when mom leaves	76.2 / 95.2	76.2 / 95.2
`temperature`	E5	Target temperature ⟶ 21 when dad comes	95.2 / 100.0	95.2 / 100.0
	E6	Target temperature ⟶ 15 when routine = sleeping	95.2 / 100.0	95.2 / 100.0
	E7	Target temperature ⟶ 21 when morning	95.2 / 100.0	90.5 / 100.0
	E8	Target temperature ⟶ 15 when mom leaves	100.0 / 100.0	100.0 / 100.0
	E9	Floor heating turns on when dad comes home	66.7 / 95.2	71.4 / 90.5
	E10	Floor heating turns to eco when temp reached	71.4 / 100.0	76.2 / 100.0
	E11	Floor heating turns on in the morning	52.4 / 95.2 *	52.4 / 90.5
	E12	Floor heating turns to eco when mom leaves	38.1 / 100.0 ***	23.8 / 95.2 ***
`weather`	E13	Chandelier turns on when the sun sets	85.7 / 100.0	81.0 / 100.0
	E14	Floor lamp turns on when the sun sets	85.7 / 100.0	81.0 / 100.0
	E15	Rolling shutters lower when the sun sets	76.2 / 100.0 *	71.4 / 100.0 *
	E16	Chandelier turns off when the sun rises	85.7 / 100.0	81.0 / 90.5
	E17	Floor lamp turns off when the sun rises	81.0 / 100.0	76.2 / 90.5
	E18	Rolling shutters raises when the sun rises	85.7 / 100.0	81.0 / 100.0
	E19	Rolling shutters lower when outdoor temp > 25°C	90.5 / 90.5	85.7 / 90.5
`security`	E20	Front door will unlock when dad comes home	100.0 / 100.0	100.0 / 100.0
	E21	Front door will lock when routine = sleeping	90.5 / 95.2	90.5 / 100.0
	E22	Front door will unlock when dad leaves	9.5 / 95.2 ***	9.5 / 52.4 *
	E23	Front door will lock when mom leaves	76.2 / 100.0	76.2 / 100.0

■ Baseline ■ FORTNIoT

rule triggered twice (E1 and E3). In the `temperature` case shown in Figure 8 and Figure 9, 47.6% and 61.9% of baseline participants missed E11 and E12 respectively, which are caused by E7 and E8. We believe they did not realise the action of one rule can trigger another rule (e.g. executing `IF somebody home THEN target temperature = 21` will also trigger `IF target temperature > measured temperature THEN turn on floor heating`).

Most of the time, participants who identified that an entity state change will happen also understood why it will happen. This makes sense, as baseline participants had to reason correctly to find the event in the first place, and FORTNIoT participants had an interactive feature to explore cause and effect. In the `security` case, however, only 52.4% of the FORTNIoT participants who found that the front door will unlock when dad leaves (E22) were able to explain why: when dad leaves, the `IF somebody is home THEN unlock the front door` rule is triggered and its condition is evaluated. This condition (counter-intuitively) evaluates to true because mom is still home, and thus the front door will open again. FORTNIoT participants were still drastically better informed than the baseline participants, where only 9.5% understood why. Furthermore, 95.2% of the FORTNIoT participants were successfully informed about this (dangerous) event, whereas only 9.5% of the baseline participants were aware. To remedy the confusion experienced by the FORTNIoT participants, we recommend highlighting not only the states that satisfy the condition of the rule, but also states that trigger the condition to be evaluated.

Table 3. The percentage of participants with undertrust when validating true statements (i.e. who wrongfully claim that a statement is false; lower is better). The remaining participants trust the behavior appropriately. No significant differences found after the Benjamini-Hochberg corrections.

Use case		True statement	Undertrust (%)
`television`	T1	TV will turn on when the news starts and somebody is home	9.5 / 4.8
	T2	Led strip turns on when the Superbowl starts	42.9 / 4.8
`temperature`	T3	Floor heating will start heating in the morning	9.5 / 4.8
	T4	Floor heating → *ECO* when target temperature reached	0.0 / 0.0
`weather`	T5	Rolling shutter will raise at run rise, even when nobody is home	9.5 / 0.0
	T6	Rolling shutter lowers when the wind speed reaches 55KM/H	4.8 / 0.0
`security`	T7	The front door unlocks when somebody leaves who is not last	23.8 / 19.1
	T8	All doors unlock when smoke is detected	0.0 / 0.0

■ Baseline ■ FORTNIoT

6.2 Appropriate Trust and Confidence

Having more appropriate trust means that participants have less overtrust (e.g. they agree less often with the false statements in Table 4), and have less undertrust (e.g. they agree more often with the true statements in Table 3). Fisher's exact test revealed a significant difference in undertrust between FORTNIoT and the baseline ($N = 336, p < 0.01, odds\ ratio = 0.3$). A breakdown of the number of participants with undertrust for each true statement is shown in Table 3. The only notable difference for undertrust, although no longer significant after the Benjamini-Hochberg correction, occurs for statement T2. In that situation, mom and dad are not home, but the TV is on (for some reason). The Super Bowl is starting soon and because of the rule `IF TV is on AND playing sports THEN turn on led strip`, the led strip will turn on. 42.9% of the baseline participants, however, thought T2 was false. Some participants (incorrectly) assumed that the TV will turn off because of the rule `IF nobody home THEN turn off TV`, but this rule was overridden by the household by turning the TV on again. Since there are no related entity state changes before the Super Bowl starts, the rule will not be triggered again.

Fisher's exact test also revealed a significant difference in overtrust between FORTNIoT and the baseline ($N = 336, p < 0.001, odds\ ratio = 0.2$). A breakdown of the number of participants with overtrust for each false statement is shown in Table 4.

One overarching mistake was that baseline participants often incorrectly assumed that when a rule exists, the opposite of that rule is implied to automatically revert the actions when the condition no longer holds (e.g. `IF sun sets THEN turn on the lights` implies `IF sun rise THEN turn off the lights`). This might be attributable to the fact that our participants are non-experts and do not yet have a proper understanding of how trigger-condition-action rules work. A significant difference in overtrust was revealed for F8, where 71.4% of the baseline participants incorrectly assumed that all doors are locked when everyone is sleeping. Participants received seemingly conflicting information about the future: mom and dad's routine will become 'sleeping' because they usually go to bed at that time, but they actually arrive home later. The rule `IF everyone is sleeping THEN lock the front door` will be triggered based on their routine, and afterwards the rule `IF anyone home THEN unlock the front door` will be triggered by them coming home late. Some baseline participants incorrectly assumed that the door will lock again: six participants assumed the rule to lock the door will trigger again, while P19 thought the rule had a "*higher priority*". P03 and P39 failed to notice the conflicting information and

Table 4. The percentage of participants with overtrust when validating false statements (i.e. who wrongfully claim that a statement is true; lower is better). The remaining participants trust the behavior appropriately. We determined significance using Fisher's exact test with Benjamini-Hochberg corrections (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

Use case		False statement	Overtrust (%)
`television`	F1	All lights turn off when Star Wars starts	23.8 / 4.8
	F2	Led strip turns on when the superbowl starts	28.6 / 0.0
`temperature`	F3	Airco will turn on when target temperature < measured temperature.	14.3 / 9.5
	F4	Floor heating will turn on earlier to anticipate target temperature	19.0 / 4.8
`weather`	F5	All lights will turn off when the sun rises	4.8 / 4.8
	F6	Rolling shutter will raise when the wind speed drops below 55KM/H	14.3 / 0.0
`security`	F7	All doors unlock when routine = sleeping ends	28.6 / 4.8
	F8	All doors are locked as long as everyone is sleeping	71.4 / 14.3 **

■ Baseline ■ FORTNIoT

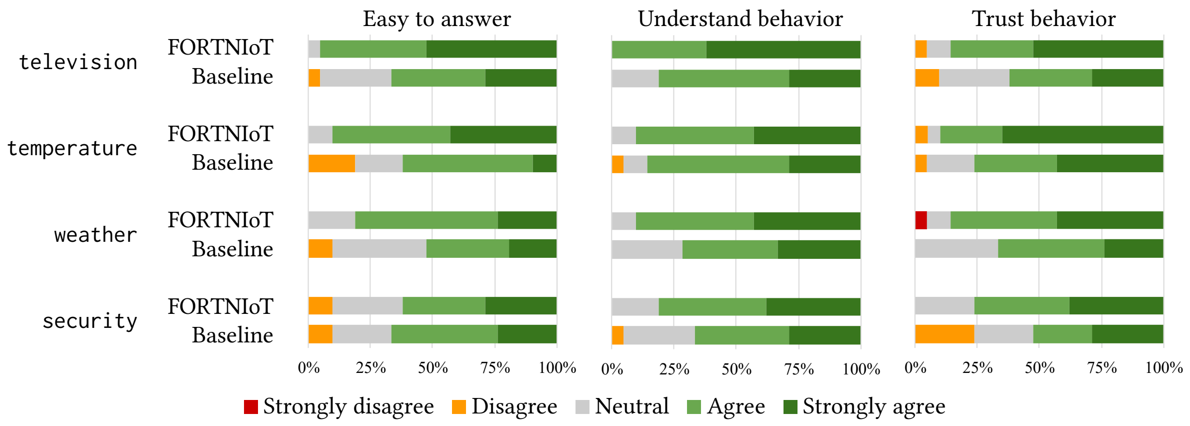
overlooked that mom and dad will be home late. 28.6% of the baseline participants thought `F2` was true because the TV is currently on and the Super Bowl will start soon, which will trigger the rule `IF tv is on AND playing sports THEN turn on led strip`. They overlooked, however, that mom and dad will leave and the TV will turn off because of the rule `IF nobody home THEN turn off tv`, and as a result, the rule that turns on the led strip will no longer be triggered.

Compared to baseline participants, FORTNIoT participants were significantly more confident in their answers when explaining what will happen ($p < 0.001, U = 4542.5, Z = 3.50, r = 0.27$) and when validating statements ($p < 0.001, U = 65118.5, Z = 4.56, r = 0.18$). More specifically, they were more confident when they had appropriate trust ($p < 0.05, U = 13178.5, Z = 2.54, r = 0.14$) and appropriate distrust ($p < 0.001, U = 11704.5, Z = 3.38, r = 0.19$). Our data do not show a significant difference in confidence when participants had undertrust or overtrust. Eight baseline participants were not completely confident in their interpretations, and eight baseline participants were not fully confident when they could not find information (e.g. rules) about the statement at hand. P05 and P06 even stated that they gave the smart home the benefit of the doubt when they could not find this information. Furthermore, six baseline participants reported that it took a lot of effort to check all information and they feared mistakes. Another source of doubt were conflicting states (six baseline participants) and not knowing which rule "*has priority*" (six baseline participants). FORTNIoT participants reported the aforementioned reasons much less often, but five of them mentioned that they lost confidence when the deduced predictions did not match their expectations.

6.3 Solution Strategy

Based on participants' self-reported strategies and notes taken by the facilitator, we observed clear differences between conditions in the strategies to find answers on what and why questions. All FORTNIoT participants except one used the dots as a starting point to understand *when* something will happen: the solid dots ● "*gave a good overview*" (P12) and "*can solve everything*" (P18). P40 stated that the empty dots ○ "*are not important because they don't have any actions*", while P07 considered them "*a second chance ... in case the first rule won't be executed*". Everyone then clicked the dots to better understand *why* the rule at hand will be executed. This feature

Table 5. After completing each use case, participants responded to 5-point Likert scale questions about perceived ease the find the answer, understanding, and trust in the smart home behavior.



proved particularly useful when multiple rules trigger in a short time span. Seventeen FORTNIoT participants also *validated the dots* by reading the rules or by looking at the predicted states.

By contrast, baseline participants had to *assemble answers* by combining written rules with external predictions using one of two strategies: *rule-driven* or *state-driven*. Those who employed a rule-driven strategy went over the list of rules only once and evaluated for each rule whether or not it would be executed given the context at hand. This strategy was inadequate, since actions of one rule can affect another rule (e.g. $E11$ and $E12$), so some rules needed to be revisited multiple times. A state-driven strategy, on the other hand, required more mental computation, because it required a pass over the list of rules for every future entity state. This approach resembles our proposed algorithm, but the order in which entity states are evaluated makes or breaks this approach. Five baseline participants used a random order, while only a chronologically ascending order guaranteed correct answers. At least four participants across both strategies mentioned the need to start over when they encountered a new fact that might invalidate their previous response, which was very time consuming.

Welch's t-tests showed that, compared to baseline participants, FORTNIoT participants required significantly less time to find out *what will happen*($t(146) = -4.82, p < 0.001, Cohen's\ d = -0.77$), and decide whether or not to trust the behavior ($t(602) = -2.47, p < 0.05, Cohen's\ d = -0.20$)". Since the experiment was carried out remotely, however, we refrain from drawing any conclusions based on this data.

6.4 User Experience and System Causability Scale

After completing each use case, participants responded to 5-point Likert scale questions about perceived ease to find the answer, perceived understanding, and self-reported trust in the smart home behavior of that use case. Their answers are depicted in Table 5. Overall, FORTNIoT participants agreed significantly more with the statement "*It was easy to answer questions about the smart home behavior*", compared to baseline participants ($p < 0.01, U = 4497, Z = 3.27, r = 0.25$). FORTNIoT provided "*sufficient details*" (P05) and presented them clearly (five participants). Interestingly, five participants thought the baseline presented a clear overview as well, while eight participants were overwhelmed, and found it hard to interpret (two participants), time consuming to apply the rules to the states (four participants) and hard because some rules have "*AND conditions*" (two participants).

Regarding the statement "*I understand the smart home behavior*", FORTNIoT participants agreed significantly more than baseline participants ($p < 0.01, U = 4338, Z = 2.79, r = 0.22$). FORTNIoT participants commented that they understood causes and effects well (P17, P40), and attributed it to their visualization (P01, P04, P09) and the clear presentation of rules (P17). In contrast, baseline participants doubted their own interpretation (P19, P22, P26), for instance because they assumed some kind of "*priority*" between rules (P03, P11, P14).

Although we measured a significant difference in the levels of agreement with the statement "*I trust the smart home behavior*" ($p < 0.01, U = 4429, Z = 3.03, r = 0.23$), the qualitative results are similar for both conditions: some smart homes are "*badly programmed*" (baseline: P08; FORTNIoT: P17 and P35) and "*will not work in all situation* (baseline: three participants; FORTNIoT: four participants). Some comments suggest that trust also depended on what kind of entities the smart home behavior was dealing with: P34 (baseline) said "*I don't care when the lights or TV are [behaving unexpectedly]*", while several participants expressed that incorrect programming of the door's lock could lead to dangerous situations (baseline: one participant; FORTNIoT: three participants).

Participants rated the baseline version 73.0 on the System Causability Scale, whereas FORTNIoT was rated 82.3. The Mann-Whitney U tests showed significant improvements regarding the statements "*I understood the explanations within the context of my tasks*" ($p < 0.05, U = 295.5, Z = 2.11, r = 0.33$), "*I could change the level of detail on demand*" ($p < 0.05, U = 312, Z = 2.43, r = 0.37$), and "*I did not need more references in the explanations*" ($p < 0.05, U = 312.5, Z = 2.41, r = 0.37$).

7 DISCUSSION

Our between-subject study with non-experienced users yielded interesting results for both the visual demonstrator of FORTNIoT and the baseline version. Both quantitative and qualitative results confirm that baseline participants still experienced difficulties in understanding what will happen in the smart home. In particular, they struggled when seemingly conflicting entity states are presented, when rules "*conflict with each other*", or when rules unexpectedly (not) trigger.

7.1 Summary and Discussion of Key Findings

Based on the results of the study, we revisit the three hypotheses that we put forward in Section 5:

H1. *FORTNIoT predictions lead to a* ***more accurate understanding*** *of the smart home behavior.* The FORTNIoT condition shows greatly improved scores on the System Causability Scale, which participants attributed to having more information available: they were able to answer the questions by interpreting the visualization, without predicting the future themselves (i.e. the baseline). Usage of FORTNIoT, however, was not limited to merely reading the visualization, as most participants also interacted with the dots to trigger the WHY-visualization and read the rule descriptions to double-check their interpretation. FORTNIoT participants missed significantly less entity states changes (1.4%) compared to baseline participants (24.4%), so the extensive predictions helped participants to be more aware of what will happen in the smart home. In 95.4% of the cases, FORTNIoT participants also understood why something will happen, whereas baseline participants only understood this in 73.7% of the cases. These results confirm hypothesis **H1**.

H2. *FORTNIoT predictions lead to a* ***more appropriate trust*** *in the smart home behavior.* A more accurate understanding should result in a better assessment of the statements about the smart home behavior. Indeed, FORTNIoT participants show a reduced overtrust. Surprisingly, the data do not show a difference in undertrust, which might be attributable to our between-subject design: participants of both groups believed that they were using the *outcome* of a research project, which might have increased trust in general. Nevertheless, we have evidence that confirms hypothesis **H2**.

H3. *FORTNIoT predictions lead to* ***more confidence*** *in participants' understanding of the smart home behavior.* Regardless of whether their answers were correct or not, FORTNIoT participants were significantly more confident about them compared to the baseline participants. This is true for the open-ended *what* and *why* questions, as well as the binary assessments of our statements. These results confirm hypothesis **H3**.

We are encouraged by the high numbers of participants who understood what will happen and why in our controlled lab study with 5-7 rules and 7-11 entities per use case. Accuracy will inevitably decrease when FORTNIoT is deployed in real smart homes with more connected entities with self-sustaining predictions, and larger rule sets that lead to more deduced predictions. Notwithstanding, as the complexity grows, tools like FORTNIoT become indispensable to help users to better understand and trust their smart home. For such intricate cases, the scalability of the visualization can be improved by, for example, automatically hiding rows for entities and rules that are not involved in the current rule execution, or employing a tree-like hierarchy based on the rooms inside the house and/or the types of entities.

Predictions are particularly useful for inhabitants who are not yet fully aware of how the smart home is behaving (i.e. in the early deployment of the network) [38]. They can, however, also be useful in later stages, for example when a set of rules was modified. Based on the quantitative and qualitative results, we also gained insights into how to improve the visual demonstrator. From E22 in our user study, it emerged that future designs should clearly distinguish between states that trigger the rule and states that satisfy the condition of the rule. Other properties, such as execution delays and repeating actions, can be visualized as well.

7.2 Limitations and Future Work

As FORTNIoT is the first attempt to pro-actively calculate intelligible predictions about the future, its proof-of-concept implementation, the visual demonstrator, and the evaluation still have limitations that present interesting opportunities for future work.

Prediction accuracy and scope. Predicting the future of a smart home is challenging, and the accuracy of the predictions is affected by several factors. First and foremost, we assume a closed world, where all information about the future is defined by the self-sustaining predictions and a set of trigger-condition-action rules. In an open world, however, unpredictable events happen (e.g. an inhabitant who interferes with the intelligent lighting by controlling some lamps manually) and the rule set might not cover all behavior (e.g. some behavior is embedded into the firmware of an entity). Furthermore, the accuracy of predictions deduced by FORTNIoT depends on the accuracy of the self-sustaining predictions. If the weather forecast turns out to be inaccurate, the predictions about rules and entities that depend on it were inaccurate as well. Our proof-of-concept implementation attempts to mediate this by considering only future paths with the highest probability, and updating all predictions for each detected change or event [50]. The visual demonstrator could show additional feedback that emphasizes these updates, to inform users of changes. In addition, richer models can take into account extra information, such as probabilities or logged information about the past, to more accurately model the holistic behavior of a smart home [11, 70]. The visual demonstrator can present these probabilities to show system confidence [42].

We envision two complementary approaches to handle probabilities: (1) defining probability density functions that denote for each reachable state the probability of actually switching to that state (e.g. probability of rain or sunshine in a weather forecast), and (2) calculating probability bounds that define the time window in which an event will probably occur (e.g. a bus arriving at a bus stop). Since predictions are interdependent, the prediction of one entity state will most likely impact other predictions as well. By branching a future prediction at each point of uncertainty, a tree of possible futures can be generated. Computing the entire tree quickly becomes infeasible, and visualizing it will overwhelm users. Instead, we recommend generating only the most probable future, by selecting the most probable option at each branch point. The visual demonstrator, however, could visualize these branch points and allow users to select an alternative state (e.g. because they think it is more likely to happen). The prediction algorithm would then compute an alternative future. Predicting and showing alternative futures on-demand is important for aligning the users' mental model with what the system is doing [11].

Evaluation. Due to the COVID-19 pandemic [58], we shifted to an online study in which we asked participants to answer questions about smart homes that are configured by us. As a result, their interpretation of a rule might be slightly different from ours. It would be interesting to evaluate FORTNIoT predictions in an in-the-wild study, where participants deal with a set of trigger-condition-action rules that they composed themselves. Another side effect of our remote study is that it was hard to measure time accurately and draw any conclusions about this. Furthermore, the experiment included state-of-the-art methods for measuring trust and the quality of explanations that have yet to prove their robustness.

Conflicts. Our proof-of-concept implementation of FORTNIoT is still prone to conflicts between rules, such as loops and race conditions. These conflicts can lead to unpredictable and dangerous behavior, such as a door that is unintentionally unlocked, a financial loss because of duplicate transactions, or health problems caused by duplicate injections of patients. Therefore, resolving conflicts is an important challenge [12, 24, 70]. Similar to previous work that focused on eliminating such conflicts during the composition of rules [20, 23, 46] or by automatically generating and correcting rules on behalf of the user [40], FORTNIoT can be adapted to detect conflicts during the prediction of the future. Since visualizations developed to enhance intelligibility can often also be used to debug (unintended) behavior [22, 48], the visual demonstrator is highly suitable to present these conflicts to the user, even before they happen.

8 CONCLUSION

In this paper, we introduce FORTNIoT, an approach to predict what will happen in the near future of a smart home to help users understand how it will behave. FORTNIoT adapts information that is present in existing IoT middleware to generate self-sustaining predictions. By simulating trigger-condition-action rules on these self-sustaining predictions, we can deduce when these rules will trigger in the future and what state changes they will cause to connected smart home entities. While previous work has focused on explaining *what* the smart home is doing in real-time [29, 39, 47, 65, 66], what it has done it the past [13, 51], or what it would do in a simulated context entered by the user [21, 30, 61], FORTNIoT is the first attempt to pro-actively calculate intelligible predictions about the future. To demonstrate how such predictions can be used, we contribute a proof-of-concept implementation and visual demonstrator.

Our between-subject study with 42 participants shows that non-experienced users have a more accurate understanding about what will happen in a smart home, and are more confident about that understanding. The responsibility of the user shifts from the cumbersome assembly of answers to merely verifying answers. While our data do not show a significant difference in undertrust, we see that predictions by FORTNIoT significantly reduce overtrust. Predictions about the future make smart homes more accountable and intelligible. As an added benefit, the visual demonstrator can be useful as a first step in troubleshooting malfunctions in the past as well (e.g. if a rule is unexpectedly not being triggered, the sensor that triggers the rule might be broken). The visualization can inspire other researchers and practitioners who are focusing on making the behavior of smart homes more intelligible, both in the past and future. We envision a wide variety of situations where predictions can be helpful, such as debugging unintended behavior and managing conflicts by exception [38]: inhabitants are warned about unwanted or unsafe behavior in the near future, so they can resolve the issue before it happens. Furthermore, predictions can provide rich intelligibility when composing or editing trigger-condition-action rules. With FORTNIoT, we contribute an important first step towards generating these predictions and hope to spark a new generation of intelligible tools for ubiquitous environments that include information about the future to the benefit of inhabitants of smart homes.

ACKNOWLEDGMENTS

This research was supported by the Special Research Fund (BOF) of Hasselt University (R-8150) and the Research Foundation of Flanders (FWO), project G0E7317N. We would like to thank all participants who took part in our study for their time. The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. Icons by Icons8.com and streamlineicons.com, and freepik.com.

SUPPLEMENTARY MATERIAL

Screenshots of our use cases in both conditions can be found in the ACM Digital Library.

REFERENCES

[1] Pierre A. Akiki, Arosha K. Bandara, and Yijun Yu. 2017. Visual Simple Transformations: Empowering End-Users to Wire Internet of Things Objects. *ACM Trans. Comput.-Hum. Interact.* 24, 2 (April 2017), 10:1–10:43. https://doi.org/10.1145/3057857

[2] Stavros Antifakos, Nicky Kern, Bernt Schiele, and Adrian Schwaninger. 2005. Towards improving trust in context-aware systems by displaying system confidence. In *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*. ACM Press, 9. https://doi.org/10.1145/1085777.1085780

[3] Carmelo Ardito, Maria F. Costabile, Giuseppe Desolda, Marco Manca, Maristella Matera, Fabio Paternò, and Carmen Santoro. 2019. Improving Tools that Allow End Users to Configure Smart Environments. In *End-User Development (Lecture Notes in Computer Science)*, Alessio Malizia, Stefano Valtolina, Anders Morch, Alan Serrano, and Andrew Stratton (Eds.). Springer International Publishing, 244–248.

[4] Mark Assad, David J. Carmichael, Judy Kay, and Bob Kummerfeld. 2007. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In *Pervasive Computing*, Anthony LaMarca, Marc Langheinrich, and Khai N. Truong (Eds.). Vol. 4480. Springer Berlin Heidelberg, Berlin, Heidelberg, 55–72. http://link.springer.com/10.1007/978-3-540-72037-9_4

[5] Louise Barkhuus and Anind Dey. 2003. Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. In *UbiComp 2003: Ubiquitous Computing: 5th International Conference, Seattle, WA, USA, October 12-15, 2003. Proceedings*, Anind K. Dey, Albrecht Schmidt, and Joseph F. McCarthy (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 149–156. https://doi.org/10.1007/978-3-540-39653-6_12

[6] Lyn Bartram, Johnny Rodgers, and Rob Woodbury. 2011. Smart Homes or Smart Occupants? Supporting Aware Living in the Home. In *Human-Computer Interaction – INTERACT 2011*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Springer Berlin Heidelberg, 52–64.

[7] Victoria Bellotti and Keith Edwards. 2001. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction* 16, 2 (Dec. 2001), 193–212. https://doi.org/10.1207/S15327051HCI16234_05

[8] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)* 57, 1 (1995), 289–300. www.jstor.org/stable/2346101 Publisher: [Royal Statistical Society, Wiley].

[9] John Brooke. 1996. *SUS: A quick and dirty usability scale.*

[10] A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. 2011. Home automation in the wild: challenges and opportunities. ACM Press, 2115. https://doi.org/10.1145/1978942.1979249

[11] Lei Bu, Wen Xiong, Chieh-Jan Mike Liang, Shi Han, Dongmei Zhang, Shan Lin, and Xuandong Li. 2018. Systematically Ensuring the Confidence of Real-Time Home Automation IoT Systems. *ACM Transactions on Cyber-Physical Systems* 2, 3 (June 2018), 22:1–22:23. https://doi.org/10.1145/3185501

[12] Danilo Caivano, Daniela Fogli, Rosa Lanzilotti, Antonio Piccinno, and Fabio Cassano. 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 (Oct. 2018), 295–313. https://doi.org/10.1016/j.jss.2018.06.035

[13] Nico Castelli, Corinna Ogonowski, Timo Jakobi, Martin Stein, Gunnar Stevens, and Volker Wulf. 2017. What Happened in my Home?: An End-User Development Approach for Smart Home Data Visualization. ACM Press, 853–866. https://doi.org/10.1145/3025453.3025485

[14] Hao-Fei Cheng, Ruotong Wang, Zheng Zhang, Fiona O'Connell, Terrance Gray, F. Maxwell Harper, and Haiyi Zhu. 2019. Explaining Decision-Making Algorithms Through UI: Strategies to Help Non-Expert Stakeholders. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 559:1–559:12. https://doi.org/10.1145/3290605.3300789

[15] Scott Clifford and Jennifer Jerit. 2014. Is There a Cost to Convenience? An Experimental Comparison of Data Quality in Laboratory and Online Studies. *Journal of Experimental Political Science* 1, 2 (2014), 120–131. https://doi.org/10.1017/xps.2014.5

[16] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. 2012. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics* 17 (Dec. 2012), 25–32. https://doi.org/10.1016/j.websem.2012.05.003

[17] Sven Coppers, Kris Luyten, Davy Vanacken, David Navarre, Philippe Palanque, and Christine Gris. 2019. Fortunettes: Feedforward About the Future State of GUI Widgets. *Proc. ACM Hum.-Comput. Interact.* 3, EICS (June 2019), 20:1–20:20. https://doi.org/10.1145/3331162

[18] Sven Coppers, Jan Van den Bergh, Kris Luyten, Karin Coninx, Iulianna van der Lek-Ciudin, Tom Vanallemeersch, and Vincent Vandeghinste. 2018. Intellingo: An Intelligible Translation Environment. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, Montreal QC, Canada, 1–13. https://doi.org/10.1145/3173574.3174098

[19] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. Empowering End Users in Debugging Trigger-Action Rules. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 388:1–388:13. https://doi.org/10.1145/3290605.3300618

[20] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor. In *End-User Development (Lecture Notes in Computer Science)*, Alessio Malizia, Stefano Valtolina, Anders Morch, Alan Serrano, and Andrew Stratton (Eds.). Springer International Publishing, 18–33.

[21] Joelle Coutaz and James L. Crowley. 2016. A First-Person Experience with End-User Development for Smart Homes. *IEEE Pervasive Computing* 15, 2 (April 2016), 26–39. https://doi.org/10.1109/MPRV.2016.24

[22] Simon P. Davies. 1993. Models and theories of programming strategy. *International Journal of Man-Machine Studies* 39, 2 (Aug. 1993), 237–267. https://doi.org/10.1006/imms.1993.1061

[23] Luigi De Russis and Alberto Monge Roffarello. 2018. A Debugging Approach for Trigger-Action Programming. ACM Press, 1–6. https://doi.org/10.1145/3170427.3188641

[24] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Empowering End Users to Customize their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools. *ACM Transactions on Computer-Human Interaction* 24, 2 (April 2017), 1–52. https://doi.org/10.1145/3057859

[25] Colin Dixon, Ratul Mahajan, Sharad Agarwal, A. J. Brush, Bongshin Lee, Stefan Saroiu, and Victor Bahl. 2010. The Home Needs an Operating System (and an App Store). In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. ACM, New York, NY, USA, 18:1–18:6. https://doi.org/10.1145/1868447.1868465

[26] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. 2002. But how, Donald, tell us how?: on the creation of meaning in interaction design through feedforward and inherent feedback. ACM Press, 285. https://doi.org/10.1145/778712.778752

[27] E. S. Reetz, D. Kuemper, K. Moessner, and R. Toenjes. 2013. How to Test IoT-based Services before Deploying them into Real World. In *European Wireless 2013; 19th European Wireless Conference*. 1–6.

[28] W. Keith Edwards and Rebecca E. Grinter. 2001. At Home with Ubiquitous Computing: Seven Challenges. In *Ubicomp 2001: Ubiquitous Computing: International Conference Atlanta Georgia, USA, September 30–October 2, 2001 Proceedings*, Gregory D. Abowd, Barry Brumitt, and Steven Shafer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 256–272. https://doi.org/10.1007/3-540-45427-6_22

[29] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, George Fitzmaurice, Elmar Eisemann, and Scott Bateman. 2017. *Ivy: Exploring Spatially Situated Visual Programming for Authoring and Understanding Intelligent Environments.* Technical Report. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine. https://doi.org/10.20380/GI2017.20

[30] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. *ACM Transactions on Computer-Human Interaction* 24, 2 (April 2017), 14:1–14:33. https://doi.org/10.1145/3057861

[31] Andreas Holzinger, André Carrington, and Heimo Müller. 2020. Measuring the Quality of Explanations: The System Causability Scale (SCS). *KI - Künstliche Intelligenz* (Jan. 2020). https://doi.org/10.1007/s13218-020-00636-z

[32] Home Assistant 2020. Home Assistant. https://www.home-assistant.io/ Accessed: 2020-05-06.

[33] Home Assistant 2020. Home Assistant Automations. https://www.home-assistant.io/docs/automation/ Accessed: 2020-08-12.

[34] Home Assistant 2020. Home Assistant History. https://www.home-assistant.io/integrations/history Accessed: 2020-07-28.

[35] Justin Huang and Maya Cakmak. 2015. Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. Association for Computing Machinery, Osaka, Japan, 215–225. https://doi.org/10.1145/2750858.2805830

[36] Ting-Hao Kenneth Huang, Amos Azaria, and Jeffrey P. Bigham. 2016. InstructableCrowd: Creating IF-THEN Rules via Conversations with the Crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 1555–1562. https://doi.org/10.1145/2851581.2892502

[37] Timo Jakobi, Corinna Ogonowski, Nico Castelli, Gunnar Stevens, and Volker Wulf. 2017. The Catch(Es) with Smart Home: Experiences of a Living Lab Field Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1620–1633. https://doi.org/10.1145/3025453.3025799

[38] Timo Jakobi, Gunnar Stevens, Nico Castelli, Corinna Ogonowski, Florian Schaub, Nils Vindice, Dave Randall, Peter Tolmie, and Volker Wulf. 2018. Evolving Needs in IoT Control and Accountability: A Longitudinal Study on Smart Home Intelligibility. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4 (Dec. 2018), 171:1–171:28. https://doi.org/10.1145/3287049

[39] Yuna Jeong, Hyuntae Joo, Gyeonghwan Hong, Dongkun Shin, and Sungkil Lee. 2015. AVIoT: web-based interactive authoring and visualization of indoor internet of things. *IEEE Transactions on Consumer Electronics* 61, 3 (Aug. 2015), 295–301. https://doi.org/10.1109/TCE.2015.7298088

[40] L. Zhang, W. He, J. Martinez, N. Brackenbury, S. Lu, and B. Ur. 2019. AutoTap: Synthesizing and Repairing Trigger-Action Programs Using LTL Properties. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 281–291. https://doi.org/10.1109/ICSE.2019.00043 Journal Abbreviation: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE).

[41] Jonathan Lazar, Adam Jones, and Ben Shneiderman. 2006. Workplace user frustration with computers: an exploratory investigation of the causes and severity. *Behaviour & Information Technology* 25, 3 (May 2006), 239–251. https://doi.org/10.1080/01449290500196963

[42] Brian Y. Lim and Anind K. Dey. 2010. Toolkit to support intelligibility in context-aware applications. ACM Press, 13. https://doi.org/10.1145/1864349.1864353

[43] Brian Y. Lim and Anind K. Dey. 2011. Design of an intelligible mobile context-aware application. ACM Press, 157. https://doi.org/10.1145/2037373.2037399

[44] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. 2009. Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 2119–2128. https://doi.org/10.1145/1518701.1519023

[45] Brian Y Lim, Qian Yang, Ashraf M Abdul, and Danding Wang. 2019. Why these Explanations? Selecting Intelligibility Types for Explanation Goals.

[46] Marco Manca, Fabio, Paternò, Carmen Santoro, and Luca Corcella. 2019. Supporting end-user debugging of trigger-action rules for IoT applications. *International Journal of Human-Computer Studies* 123 (March 2019), 56–69. https://doi.org/10.1016/j.ijhcs.2018.11.005

[47] Simon Mayer, Yassin N. Hassan, and Gábor Sörös. 2014. A magic lens for revealing device interactions in smart environments. ACM Press, 1–6. https://doi.org/10.1145/2669062.2669077

[48] Anneliese von Mayrhauser and A. Marie Vans. 1997. Program Understanding Behavior During Debugging of Large Scale Software. In *WORKSHOP ON EMPIRICAL STUDIES OF PROGRAMMERS*. ACM Press, 157–179.

[49] Sarah Mennicken and Elaine M. Huang. 2012. Hacking the Natural Habitat: An In-the-Wild Study of Smart Homes, Their Development, and the People Who Live in Them. In *Pervasive Computing (Lecture Notes in Computer Science)*, Judy Kay, Paul Lukowicz, Hideyuki Tokuda, Patrick Olivier, and Antonio Krüger (Eds.). Springer Berlin Heidelberg, 143–160.

[50] Sarah Mennicken, David Kim, and Elaine May Huang. 2016. Integrating the Smart Home into the Digital Calendar. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 5958–5969. https://doi.org/10.1145/2858036.2858168

[51] Sarah Mennicken, Jo Vermeulen, and Elaine M. Huang. 2014. From today's augmented houses to tomorrow's smart homes: new directions for home automation research. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct*. ACM Press, Seattle, Washington, 105–115. https://doi.org/10.1145/2632048.2636076

[52] Rebeca Campos Motta, Káthia Marçal de Oliveira, and Guilherme Horta Travassos. 2019. A Framework to Support the Engineering of Internet of Things Software Systems. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '19)*. ACM, New York, NY, USA, 12:1–12:6. https://doi.org/10.1145/3319499.3328239

[53] Bonnie M. Muir. 1994. Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics* 37, 11 (Nov. 1994), 1905–1922. https://doi.org/10.1080/00140139408964957

[54] N. M. Mosharaf Kabir Chowdhury and R. Boutaba. 2009. Network virtualization: state of the art and research challenges. *IEEE Communications Magazine* 47, 7 (July 2009), 20–26. https://doi.org/10.1109/MCOM.2009.5183468

[55] Michele Nitti, Virginia Pilloni, Giuseppe Colistra, and Luigi Atzori. 2016. The Virtual Object as a Major Element of the Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1228–1240. https://doi.org/10.1109/COMST.2015.2498304

[56] Donald A Norman, Broadbent Donald Eric, Baddeley Alan David, and Reason J. 1990. The 'problem ' with automation: inappropriate feedback and interaction, not 'over-automation'. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 327, 1241 (April 1990), 585–593. https://doi.org/10.1098/rstb.1990.0101

[57] Dan R. Olsen. 2007. Evaluating user interface systems research. ACM Press, 251. https://doi.org/10.1145/1294211.1294256

[58] World Health Organization. 2020. Coronavirus disease (COVID-19) Pandemic. https://www.who.int/emergencies/diseases/novel-coronavirus-2019

[59] Philipp Rosenkranz, Matthias Wählisch, Emmanuel Baccelli, and Ludwig Ortmann. 2015. A Distributed Test System Architecture for Open-source IoT Software. ACM Press, 43–48. https://doi.org/10.1145/2753476.2753481

[60] S. R. Ellis. 1994. What are virtual environments? *IEEE Computer Graphics and Applications* 14, 1 (Jan. 1994), 17–22. https://doi.org/10.1109/38.250914

[61] Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, and Wenjun Wu. 2019. SAPIENS: Towards Software Architecture to Support Peripheral Interaction in Smart Environments. *Proc. ACM Hum.-Comput. Interact.* 3, EICS (June 2019), 11:1–11:24. https://doi.org/10.1145/3331153

[62] Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte Carlo Methods for Managing Interactive State, Action and Feedback Under Uncertainty. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 235–244. https://doi.org/10.1145/2047196.2047227

[63] Ben Shneiderman. 2003. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *The Craft of Information Visualization*, BENJAMIN B. Bederson and BEN Shneiderman (Eds.). Morgan Kaufmann, San Francisco, 364–371. https://doi.org/10.1016/B978-155860915-0/50046-9

[64] Blase Ur, Jaeyeon Jung, and Stuart Schechter. 2013. The current state of access control for smart devices in homes. HUPS 2014.

[65] Jo Vermeulen, Kris Luyten, and Karin Coninx. 2012. Understanding Complex Environments with the Feedforward Torch. In *Ambient Intelligence (Lecture Notes in Computer Science)*, Fabio Paternò, Boris de Ruyter, Panos Markopoulos, Carmen Santoro, Evert van Loenen, and Kris Luyten (Eds.). Springer Berlin Heidelberg, 312–319.

[66] Jo Vermeulen, Jonathan Slenders, Kris Luyten, and Karin Coninx. 2009. I Bet You Look Good on the Wall: Making the Invisible Computer Visible. In *Ambient Intelligence*. Vol. 5859. Springer Berlin Heidelberg, Berlin, Heidelberg, 196–205. https://doi.org/10.1007/978-3-642-05408-2_24

[67] Jo Vermeulen, Geert Vanderhulst, Karin Coninx, and Kris Luyten. 2009. *Answering Why and Why Not Questions in Ubiquitous Computing*.

[68] J. Vermeulen, G. Vanderhulst, K. Luyten, and K. Coninx. 2010. PervasiveCrystal: Asking and Answering Why and Why Not Questions about Pervasive Computing Applications. In *2010 Sixth International Conference on Intelligent Environments*. 271–276. https://doi.org/10.1109/IE.2010.56

[69] van der B. J. J. Vlist, J. Hu, G. Niezen, and L. M. G. Feijs. 2012. Semantic connections : a new interaction paradigm for smart environments. In *7th International Workshop on Design and Semantics of Form and Movement (DeSForM 2012), April 18-20, 2012, Wellington, New Zealand*. https://research.tue.nl/nl/publications/semantic-connections--a-new-interaction-paradigm-for-smart-environments(3cb00d46-4594-4e12-b309-f01872bc0c6d).html

[70] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. 2019. Charting the Attack Surface of Trigger-Action IoT Platforms. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, London, United Kingdom, 1439–1453. https://doi.org/10.1145/3319535.3345662

[71] Rolf H Weber and Romana Weber. 2010. *Internet of things*. Vol. 12. Springer.

[72] Fumeng Yang, Zhuanyi Huang, Jean Scholtz, and Dustin L. Arendt. 2020. How do visual explanations foster end users' appropriate trust in machine learning?. In *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI '20)*. Association for Computing Machinery, Cagliari, Italy, 189–201. https://doi.org/10.1145/3377325.3377480

[73] Rayoung Yang and Mark W. Newman. 2013. Learning from a Learning Thermostat: Lessons for Intelligent Systems for the Home. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 93–102. https://doi.org/10.1145/2493432.2493489