Context-dependent Explainable Daily Automations

Simone Gallo[1], Sara Maenza[1], Andrea Mattioli[1], and Fabio Paternò[1*]

[1] CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 56124 Pisa, Italy

Abstract

The pervasiveness of objects equipped with sensors and actuators in daily environments has enabled the possibility of creating automations able to connect their behaviour according to trigger-action rules. However, in a smart space, there may be multiple active automations, even created by different people, and thus the resulting behaviour can be different from that expected for several reasons. Thus, there is a need for explanations able to indicate why such problems occur, and how to modify the relevant automations in order to better achieve the current user goals. To make such explanations effective, it is important that they be context-dependent because the actual results may depend on the current state of environmental variables. We introduce an approach able to support such adaptive explanations and two possible front-ends to present them (one visual and one conversational), with the associated feedback received in a preliminary user study that can be useful to derive general suggestions about how to present such explanations.

Keywords

End-user development, explainability, trigger-action programming, internet of things

1. Introduction

Current technological trends determine the wide availability of smart spaces in various parts of our lives characterised by the presence of ecosystems of connected objects, services, and devices. The IoT ecosystem is projected to encompass approximately 29 billion connected devices by 2030 (https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/). In order to better exploit them in an integrated manner several automations are often deployed in such settings. The automations can be generated automatically through machine learning or under user control. In some cases, for example in smart homes, there can be even multiple people that create automations for the same smart space. Such automations can be described in terms of trigger-action rules. In general, while creating automations is not particularly difficult, it can be problematic to obtain a set of automations that behave as expected [2] or that are effective in achieving the current user goals [4], which are fundamental aspects for humanations [8], automations that people can understand and modify.

The possible issues can be determined by several aspects: conflicting automations [1], automations that activate other ones even if that was not a planned effect and automations that in theory could be useful but in the current state of the considered environment are not effective. Unfortunately, if we consider current tools used for creating automations, such as Home Assistant, IFTTT, Alexa routines, none of them provide support for addressing such problems. Thus, there is a strong need for adaptive explanations able to consider the current contextual state, the created automations and users' intentions and indicate why there are problems, and how they can be addressed by modifying the current automations [8].

Joint Proceedings of the ACM IUI Workshops 2025, March 24-27, 2025, Cagliari, Italy

* Corresponding author.

simone.gallo@isti.cnr.it (S. Gallo); sara.maenza@isti.cnr.it (S. Maenza); andrea.mattioli@isti.cnr.it (A. Mattioli); fabio.paterno@isti.cnr.it (F. Paternò)

0000-0002-5162-0475 (S. Gallo); 0009-0000-2205-6599 (S. Maenza); 0000-0001-6766-7916 (A. Mattioli); 0000-0001-8355-6909 (F. Paternò)

To address such issues, we have investigated how to identify such potential issues and provide relevant information to explain them. In addition, we have also started to consider the problem of how to provide explanations to describe such problems and help users solve them. For this purpose, we have also started to consider the use of two interaction modalities (visual dashboard and conversational agent). In this paper, we also report on some initial user feedback received on these two front-ends for explainable automations, and discuss possible design implications.

2. The Approach for Adaptive Explainable Automations

The proposed approach can be connected to two possible environments (see Figure 1):

- The real smart home, through the support of an installation of Home Assistant, an open-source tool that supports integration and communication with a wide variety of sensors and connected objects.
- An interactive context simulator allowing users to specify the specific contextual situation to consider, and a set of automations to consider.

The module performing the back-end rule analysis is composed of two parts: one dedicated to identify undesired interdependencies between the automations, and one oriented to assess the actual effectiveness of the automations for the current user goal considering the state of the environmental parameters in the smart space. The main undesired interdependencies considered are the detection of action conflicts and of activation chains between pairs of automations. For the former, there is a first analysis of whether two automations are triggering conflicting actions on the same object**,** and in the positive case, a check whether such actions can be activated in overlapping time intervals. For activation chain identification, the system detects when the activation of an automation is caused by another one. These rule chains can be direct, when a rule action performs a change that is also a trigger for another automation. There is also the possibility that an action acts on an environment variable that is "observed" by a trigger of another automation; thus, an indirect relation [6] exists between the two rules. Although there is no certainty that the first rule will lead to the activation of the second, users should be aware of the possibility. This feature is provided using a semantic module described below. If more automations act on the same environment variable causing another automation to start, both are notified to the user. Loops can be considered a special case of rule chains when the last action reactivates the first trigger.

For goal-related explanations, the system starts by analyzing the impact of current automations on environmental parameters such as light, temperature and air pollution. This analysis is supported by a semantic module developed by considering solutions in the literature defining IoT objects' effects [3, 7]. It maps the relationships between actions and environmental parameters and identifies how these parameters can lead to potential negative effects in relation to the target chosen by the user.

The analysis proceeds by assigning environmental parameters to fuzzy variables. The system checks whether the situation could activate the fuzzy rules to determine whether the negative effect occurs within the selected context. If this is the case, it returns the activation strength confidence of each activated rule. This approach enables the system to provide users with a clear indication of the problem's severity. For example, a fuzzy rule might state that if the user is at home and the noise level is excessively high, then there will be a serious well-being and health issue.

Whatever issue is addressed, the back-end for the rule analysis connects to a recommendation system to obtain suggestions on possible solutions to the identified problem. This system exploits a dataset of automation rules and provides recommendations based on the similarity of rules. In the case of problems related to specific objectives, the recommendation system receives, in addition to the problematic automation, the goal with which it conflicts and the fuzzy rule that highlights the problem. In a goal-related analysis, for example, if the goal is **well-being** and the automations indicate that *if the user is at home and the temperature is higher than 23 degrees then the windows should be open* but the current temperature outside is higher than 23 degrees then the system is able to suggest to not open the window and activate the air conditioning for reaching the desired goal.

The results of the analysis are provided to two front ends, a conversational agent and a visual dashboard, since one of the goals of our study is to understand the most effective solution in providing explanations about automations and their issues. In the following we describe how the two front-ends provide such explanations.

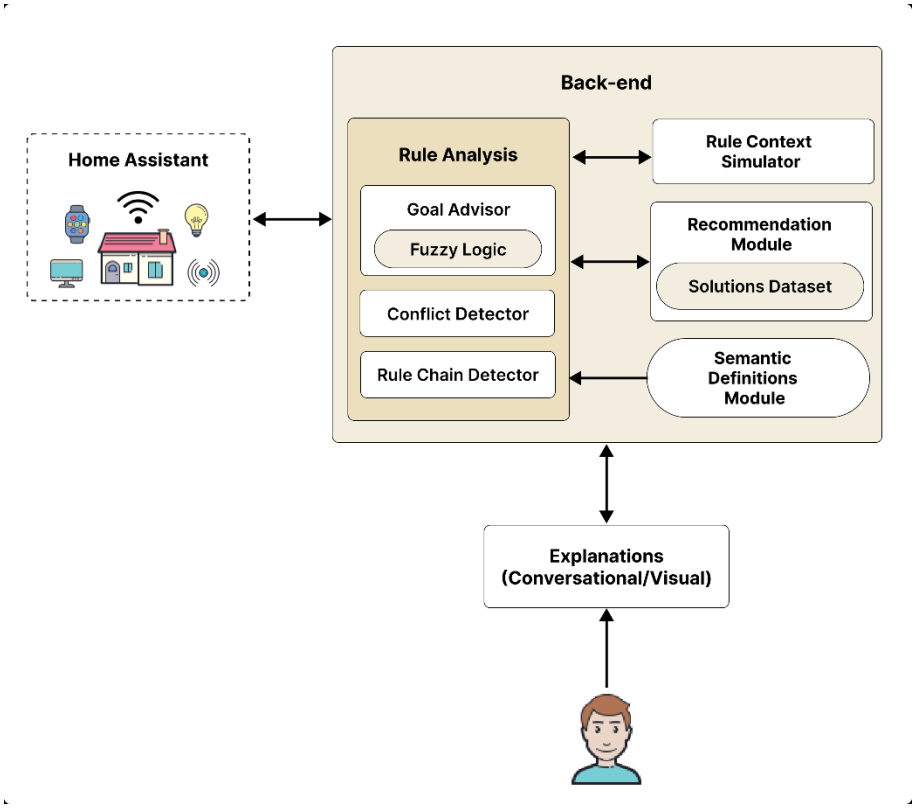

Figure 1: The Proposed Approach.

3. Visual Dashboard

The visual system is a Web dashboard designed for home users in smart homes, allowing them to retrieve information about the devices installed in the environment, the configured automations, detected issues, and related suggestions via APIs that interface with the backend. The system is intended for users with a basic understanding of smart home setups. Once logged in, the user is greeted by the homepage, which provides an overview of the analysis functionalities available to identify and resolve issues within their automation setup. The functionalities, named "Conflicts and Chains" and "Conflicts with Goals", are presented with a brief description of the issues they help identify, along with a button for direct access. On the left side of the page, there is a navigation menu for quick access to the platform's various sections.

On the "Automations" page, the user is presented with a table listing all the automations configured through Home Assistant. Each row represents an automation, displaying its name and a description in the When-If-Action format.

The analysis section is divided into two parts. The first, "Conflicts and Chains," features two areas: a general overview and a section that identifies and explains detected issues. The first area, accessed via the "Overview" tab, summarizes the number of conflicts and chains identified, along with a history of recently detected issues. The second area, accessible through the "Explanations" tab, displays three blocks, each corresponding to a category of problems: “Conflicting Automations”, “Direct Automation Chains”, and “Indirect Automation Chains”.

Each block includes an info button that uses practical examples to explain when and why certain problems occur. Selecting a block reveals a list of identified conflicting automation pairs, for example: "*when air quality is poor, open the windows*" may conflict with "*Turn on heating and relaxing light if the temperature is below 17°C*". Clicking on an item in the list generates two additional panels on the right. The first panel shows the involved automations in the When-If-Action format, while the second provides a textual explanation of the issue, such as: *the execution of the automation “When air quality is poor, open the windows” may trigger the automation “Turn on heating and relaxing light if the temperature is below 17°C” because it can alter the temperature variable*. In this section, clicking the "How can I resolve this?" button displays suggestions for resolving the issue, organized into three categories: modifying the event, adding conditions, or changing actions. All the displayed information is retrieved from the backend via API calls.

The second analysis tool, "Conflicts with Goals," helps identify issues between automations and a user-selected goal related to the environment, which can be either real or simulated, and to user preferences. For the latter, the platform offers a page dedicated to configuring desired values for the smart environment. In this section, users can set or modify specific parameters, such as the temperature in a particular area of the house (e.g., the living room).

Like the previous tool, the page includes an overview of identified issues and a history of recent findings. A scenario simulator is also available, allowing users to create, modify, or delete virtual environments representing device states, environmental conditions (such as weather or time), and personal parameters (like the user's location). Each simulated environment is identified by a user-defined name. The automation analysis section features a guided workflow. Initially, users can choose to analyze all automations or only select a few. Next, they select the environment, choosing between the real one, based on current device data, or one of the previously created simulated environments. Finally, they are asked to select a specific goal from the following options: well-being, safety, preservation, energy saving, or health. Once the goal is selected, all choices are sent to the backend for analysis.

After the analysis is complete and results are retrieved, a list of automations conflicting with the selected goal is displayed. Selecting an automation from the list opens an explanation panel on the right, detailing the causes of the conflict (e.g., *the light is on when no one is home, causing energy waste*), along with a confidence level expressed qualitatively (e.g., *high confidence*). Each issue includes a "+" button that allows users to explore the details, including a more in-depth description of the causes (e.g., *this issue occurs when the light is on, and no one is home*), the confidence level also expressed quantitatively (e.g., *high 83.33%*), the environmental variables responsible for the issue (e.g., *Light: on, People: no one is home*), and suggestions for resolution.

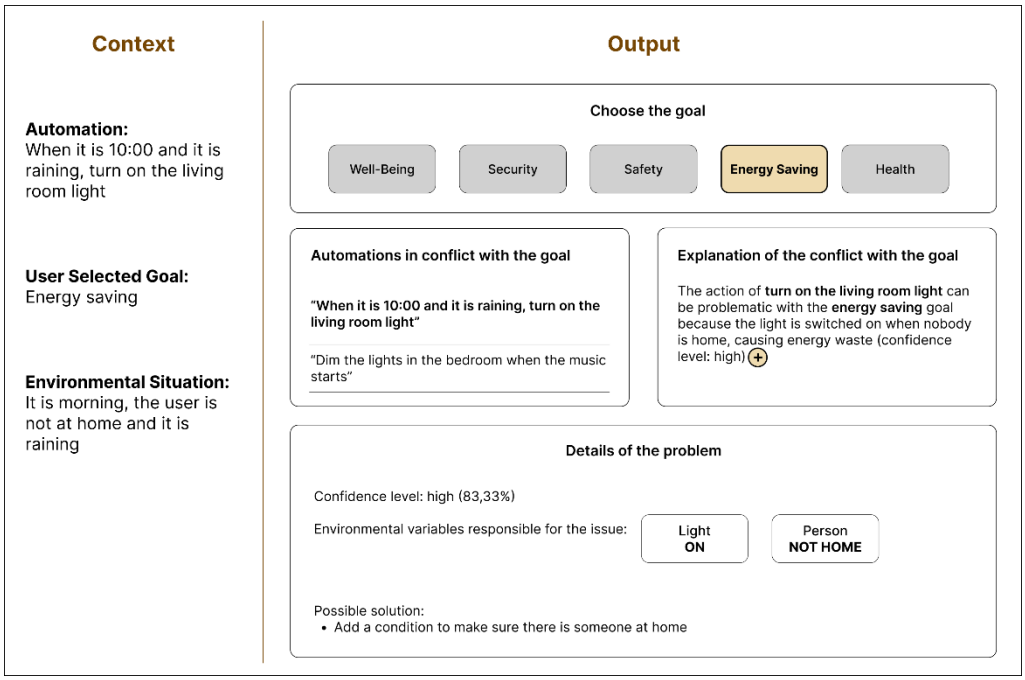

Figure 2: Schematic functioning of the visual dashboard.

4. Conversational Agent

The conversational agent is based on a previously developed and tested prototype to create trigger-action automation [5]. The new system employs a multi-agent architecture, with each agent specifically designed to address a distinct problem. This modular and scalable approach enables specialised LLM-based modules to collaborate in performing discrete tasks, thereby improving reliability and usability. By employing task-specific prompts, the system mitigates challenges such as model hallucination and simplifies the refinement and debugging of individual components. The chatbot architecture includes a conflict **manager agent** responsible for resolving issues between two automations (e.g., conflicts, direct chains), a **goal optimiser** agent tasked with identifying and optimising automations based on specified objectives, and an **explainer agent** dedicated to generating precise conflict explanations. These agents operate under the coordination of a **dialogue manager** agent, which directs user requests to the appropriate agent. Each agent interacts with the user until its task is completed, after which it returns the user to the dialogue manager along with details of the completed task.

After login, the user interacts with the chatbot through a web-based chat interface. The chatbot initiates the conversation by introducing itself and briefly outlining its functionalities. The left section of the interface displays a list of detected problems, with each issue identified by a generated conflict name and an associated ID. Selecting a conflict name reveals a brief description of the problem (e.g., "The automation 'open windows if poor air quality' indirectly triggers 'turn on heaters when 18 degrees'") and provides a "Why?" button. Clicking this button sends the problem to the chatbot, prompting it to generate an explanation.

Alternatively, the user can directly instruct the chatbot to search the problem list and identify a specific issue (e.g., "Why are the windows open even if the heating is on?"). The chatbot identifies the relevant problem and dynamically generates an explanation (via the explainer agent) to clarify its cause, then asks the user for confirmation to resolve the issue (e.g., "The windows open because the automation 'open windows if poor air quality' triggers when the air quality is 'poor'. This automation generates the issue 'Indirect Chain: windows and heating (ID:

12)': The windows open when the air quality is 'poor', and the heating turns on when the inside temperature drops below 18 degrees. Do you need further details, or would you like to resolve the issue?").

Upon receiving the user's agreement, the conversation is handed over to the conflict manager agent, who receives the problem details from the dialogue manager. The conflict manager begins by concisely summarizing the issue and asks which of the two automations the user would prefer to modify to resolve the conflict (e.g., "We have two automations involved in an indirect chain: 'open windows if poor air quality' and 'Turn on heaters when 18 degrees' which of the two automation would you like to change to solve the problem?"). After the user chooses, the chatbot prompts them to specify which part of the rule they wish to modify (e.g., "Do you want to edit the event, add a condition, or change the action of the 'open windows if poor air quality' automation?"). At this stage, the chatbot consults the recommender system to obtain potential modifications for the automation (e.g., "Here are some solutions you could consider to solve the problem by adding a condition: 1. Add the condition 'if the heating is off' to avoid opening the windows when the heating is on. 2. Add the condition to verify if the outside temperature is above a certain value. Would you like to proceed with one of these suggestions, or would you prefer to adopt a different solution?"). The users can either implement one of the suggested changes or propose their solution. Once the user is satisfied with the modifications, the chatbot summarises the updated automation in a "When, If, Then" format and requests confirmation (e.g., "**When** the air quality is 'poor', **if** the heating is off and the outside temperature is above 22 degrees, **Then** open the windows"). The new automation is generated and stored in the database upon receiving the user's approval. At any stage of the process, the user can request clarifications, revise their choices, or seek additional assistance from the chatbot.

The process for retrieving and resolving a conflict related to a user-defined goal follows a similar workflow but is managed by the goal optimiser agent. Users can access the history of previously detected conflicts associated with a goal from the left-hand section of the interface (under the "goal" section). The user can prompt the chatbot for an explanation using the "Why?" button or type the explanation request and, if desired, initiate the resolution process.

Additionally, users can request to adapt their automation in a simulated scenario to identify potential conflicts with a goal (e.g., "Could I have energy-saving problems this winter?" or "What security issues could arise when I leave home?"). The goal optimiser agent initiates a conversation to define a scenario based on the user's query and seeks confirmation (e.g., "To simulate a winter scenario, we can set a lower outdoor temperature and higher humidity. Does this align with what you had in mind?"). Once the user confirms or refines the details, the agent adjusts the parameters accordingly and generates a simulation that aligns with the specified conditions. Finally, the user goal and the defined scenario parameters are sent to the backend for analysis. If any problem is detected, the chatbot prompts the user to initiate the resolution process ("The conflict between the goal "Energy saving" and the automation "Turn on heating and relaxing light if below 18 degrees" occurs because the automation is triggered when the indoor temperature drops below 14°, activating the heating system and a relaxing light in the living room also happens when the user is not at home, which contradicts the energy-saving goal, as energy is consumed unnecessarily without any immediate benefit. Do you want to solve this problem?"). The problem is then resolved using the same procedure employed by the conflict manager agent, receiving recommendations and iteratively refining the automation until user confirmation.

5. User Feedback

We carried out a user study to assess the strong and weak points of the visual and conversational user interfaces of the proposed solution (ExplainTAP). The participants were recruited from a Digital Humanities degree course. Twenty-five participants (18 females and 7 males) participated in the study. Their ages range between 21 and 28 years (mean=23.96, std. dev.= 1.95). Students have taken some programming courses during their university career, but the degree program does not have an emphasis on programming.

During a meeting with students, the basic concepts of trigger-action programming for the Internet of Things were explained to them. They were also introduced to the main problems that can occur between automations, such as conflicts or chain activations, or between an automation and a long-term user goal. Next, they were presented with the tools they would use for the study (the visual and conversational interfaces), along with an explanation of the tools' main features. The students had to register for the platforms, login, and familiarize themselves with the tools. Next, they were presented with tasks to perform. There were three tasks per tool, which involved a direct chain, an indirect chain, and a goal-related optimisation. The tasks were written to be of balanced difficulty.

After receiving instructions, participants had to complete the six tasks independently using the tools, starting freely with one or the other. After completing the tasks, they were to fill out a questionnaire regarding demographic information (age and gender), the System Usability Scale (10 statements with which to express agreement using a 5-point Likert scale), two statements concerning the easiness of detecting and resolving problems with that interface (to be motivated with an open-ended answer), and two open-ended questions to obtain comments for positive and negative aspects of the systems used.

The efficiency and usability of the two tools were evaluated through an analysis of the questionnaire results and the open comments provided at the end of the test for each tool. The open comments were labelled and inductively grouped into themes, reported with some representative quotes.

Overall, the SUS scores indicate a preference for the conversational interface (mean score: 78.3, min=45, max=100, std. dev. = 14.98) over the graphical version (mean score: 47.7, min=5, max=85, std. dev. = 21.5). The Shapiro–Wilk test confirms that the data are normally distributed (Shapiro-Wilk normality test, $W=0.9$, $p=0.1$), and the difference between the distributions is significant (Student's t-test $t = -5.3$, $p < 0.001$). Cohen's d (1.1) suggests a large effect. Responses to the statements about the perceived ease of detecting problems were generally positive for both applications, with a preference for the conversational one (ExplainTAP: mean = 3.04, median = 3; RuleBot mean = 4.4, median = 4). The same trend can be observed in the questions specific to resolving problems (ExplainTAP: mean = 3.04, median = 3; RuleBot: mean = 4.24, median = 4).

Some themes emerge from the comments about the positive and negative aspects of the applications. The most cited (count > 3) are reported in the following.

Concerning the positive aspects of the visual interface, the most appreciated feature is the **interface design**, specifically its organization, layout, and graphic style, which was mentioned 17 times. Participants appreciated that the clear division into distinct sections, such as dashboards and analysis tools, enhances their understanding of the tool's structure and purpose. This is reflected in comments like "Organization into distinct sections (such as dashboards and

analysis) contributes to a general understanding of the structure and main purposes of the tool", "Pleasant graphical interface", and "Visual consistency". Another frequently mentioned strength is the **suggestions** feature and the variety of proposed alternatives cited by eight respondents. They valued the ability to present possible resolutions ("The fact that the system already provides you with solutions") and the usefulness of this feature ("the 'How Can I Resolve?' section"). The **explanations** of problems also stood out as a positive aspect, mentioned by five users. Detailed descriptions of conflicts were appreciated for their clarity and depth ("The explanation of the conflicts is exhaustively detailed and allows the user to have all the tools needed to understand how they can find a solution.") and for the value of making these problems more graspable ("The detailed explanation of each conflict, as it makes the work easy even for those without technical skills."). Finally, the **partitioning of problems** into categories was praised by four respondents ("in the Dashboard division between conflicts and chains and conflicts with goals").

Several negative aspects were also frequently highlighted, revealing areas for improvement. The most cited issue is the **lack of sufficient feedback** regarding the actual resolution or non-resolution of the problem mentioned by 16 respondents ("Clicking on "Resolve" makes it unclear if indeed the problem has been resolved, as I see no change in the interface.", "Lack of feedback as to whether we are acting well or acting poorly", "The system doesn't give you an output: it only allows you to save your solution but you can't tell where so it gives you the feeling that you are losing data and not really doing the work"). Another significant issue concerns the interface's **intuitiveness and clarity**, as mentioned by nine respondents. Participants sometimes found the application overly complex and difficult to navigate ("The application is very complex; you can't immediately interact with it", "The initial lack of clarity of ways to go about finding problems and subsequently solving them"). Issues with the **distribution and organization of content** were also reported, with seven respondents criticizing the interface for being unintuitive and overly fragmented. They described feeling disoriented by the arrangement of information ("The arrangement of information and the way to reach it is unintuitive,") while others noted that "The many sections are disorienting" and an "excessive branching/fragmentation of content". Finally, some users highlighted the lack of functionality for **viewing changes made or accessing a history** of actions on the platform, which seven respondents mentioned. The absence of this feature led to confusion and hindered their ability to track progress ("I no longer saw the change I made so much that I thought I had acted incorrectly"), and they suggested adding a feature to save and review resolutions ("Insert a section in which to review all the changes made", "One improvement that could be made is the ability to save the resolutions made, so that we have a history of the work done, perhaps by updating the 'overview" section.").

Concerning the Conversational interface, one of the most frequently mentioned aspects was the **clear structure and appearance of the interface** (13 respondents). They appreciated the absence of unnecessary elements ("The visual simplicity: the application contains no unnecessary or superfluous elements; this allows the user to focus on the elements that are important and can support problem-solving."). The UI was also commented on for being understandable and well-structured ("well-understandable graphical user interface.", "the structure of the application (especially the breakdown by conflict type on the left banner)"). Another standout feature was the interface's support for **natural language and conversational interaction**, cited by 11 respondents. Users valued how easily they could interact with the chatbot ("The fact that it is a chatbot that is easy to interact with conversationally"), highlighting its ability to understand and respond appropriately to inputs ("ability to formulate simple and direct answers to the bot that it understood right away," "Rulebot understands the messages sent rather well and provides appropriate answers."). The interface's ability to provide

troubleshooting guidance also received significant praise, with nine users highlighting its step-by-step approach to identifying and solving problems ("Step-by-step approach in identifying and solving problems," "Conversation was guided, I was able to understand the problem by asking for more information, and the resolution was guided through AI help,"), and the assistant-like support provided ("Chatbot acts as a troubleshooting assistant and guides the user"). **Simplicity** was a strength noted by five respondents. Users appreciated how the interface required **little to no prior knowledge** to be used effectively ("Ease of use, you don't need much prior knowledge", "The simplicity of the interface helps you settle in right away.") The interface's **speed** was also cited by five respondents, who described it as being fast in processing responses and resolving problems efficiently ("Speed in processing responses", "the speed of problem solving"). Additionally, four users mentioned the tool's **feedback**. Respondents highlighted the usefulness of feedback, in particular the one that confirmed conflict resolution ("Efficient feedback (e.g., that which indicates that the conflict has been resolved)"). Finally, four users appreciated the **variety of solutions** provided by the chatbot ("provide various options to problems that can be thought about", "Problem solutions proposed by the bot").

Users identified areas for improvement of the conversation interface, with some recurring themes emerging in their feedback. One was related to **errors or freezes** mentioned by nine respondents. Users reported instances where the chatbot would stop responding ("Sometimes I had to redo the same question because the chatbot was no longer responding. It was like stuck."), and noted that errors could cause the system to crash or hang indefinitely ("Sometimes it crashes and, especially if typos were made, it loads a message without ever sending it instead of reporting that it did not understand"). Another frequent complaint was about the **speed** of the chatbot (nine participants). While some appreciated the tool's speed in certain contexts, others felt it was inconsistent and occasionally slow ("Occasionally slow in searching for answers", "Improve speed of loading chatbot answers"). The lack of a **message history** functionality was also a concern mentioned by eight respondents. They expressed a desire to save conversations with the chatbot for future reference ("memory of the past conversations with RuleBot"). One user remarked "The ability to save resolutions made, saving the conversation had with the chatbot for later viewing... could be useful both for educational purposes and for a possible second check of what has been done". The **organization of the interface** was an area where users saw room for improvement, with seven respondents raising concerns and providing various UX suggestions ("Location of the Logout button: it would be better in the upper right corner within a drop-down menu", "Unable to reduce the sidebar containing the 'automations' and 'issues': opt for a full-screen view of the chat"). One user suggested integrating conflict explanations directly into the sidebar to eliminate the need for unnecessary conversations ("Maybe you could think about inserting the 'why?' of conflicts directly by pressing on them... so you don't have to start a conversation only to find out that it is not interesting at that moment"). Some users also pointed out challenges related to **facilitations for inexperienced users and accessibility**, as mentioned by five respondents. They noted that the application could be difficult to navigate ("hints for using the application itself. For example, in the "problems" section, it could have been useful to include text such as 'choose a problem and solve it through chat'. This could have helped less experienced and possibly visually impaired users using screen readers", "For those who are not experts in trigger-action automations, it is difficult to find their way around the application"). Finally, some users noted issues with **conversation paths and misunderstandings**, cited by four respondents. While the guided nature of the chatbot was appreciated during problem resolution, others found it limiting. One user remarked, "The fact that it is very guided is also a limitation because when I tried to get off the 'tracks', it encountered difficulties in understanding me." Others pointed out that the chatbot struggled when questions were not phrased in a specific

way: “You have to ask it the questions in a certain way; otherwise, the chatbot gets confused and can't answer you correctly”.

6. Conclusions and Future Work

In this paper, we introduce an approach (ExplainTAP) to identify and solve different problems within a smart environment. The preliminary versions of two visual and conversational interfaces were compared in a user test. The study identified some key aspects that are useful for improving interaction with such environments. One is implementing an interaction history, which could serve to visualise changes made in the rules (e.g., by providing a visualisation of the differences [11] between the old and the new automation). This idea also applies to the conversational interface by showing the list of past conversations to revise successful interactions that led to the resolution of the problem. Another aspect is feedback regarding the solution to the problems. It is important to make explicit whether the changes made to the rules have actually solved the issue (and do not cause further ones). The positive comments related to the suggested recommendations highlight the importance of supporting users not only by identifying and categorising problems but also by providing possible suggestions for resolution. Finally, it is crucial to balance the wealth of functionality offered with the need to make the interfaces streamlined and quick to learn. In this respect, the conversational solution was more convincing, as shown by the SUS scores. One aspect that led to the higher perceived usability is that with the conversational interface, users can directly express what they need instead of searching for specific functionality. Still, the usefulness of features for facilitating non-expert users has emerged even in this environment.

In future work, the presented interfaces will be improved using data from this study, e.g., by making feedback clearer and solving speed and blockage problems. Subsequently, we will organise a new, more structured user study, in which we will also evaluate how well the tools are able to help users identify and solve issues in automations from a quantitative point of view. We are also planning to improve the recommendations, for example, by considering implicit user intents [10] within the recommendation generation process. Additionally, we are exploring the development of a hybrid visual-conversational interface to capitalise on the strength of both approaches. A fruitful approach could be supporting users with a conversational interface for problem identification and with a visual interface for the resolution process. Another possibility is to enable multi-user management of the environment by providing diverse users with the most relevant information (e.g., the householder can receive detailed information on problems, while guests only receive essential information).

Acknowledgements

This work has been supported by the Italian MUR PRIN 2022 PNRR Project P2022YR9B7, End-User Development of Automations for Explainable Green Smart Homes, funded by European Union - Next Generation EU. This work is also funded by European Union - Next Generation EU, in the context of The National Recovery and Resilience Plan, Investment 1.5 Ecosystems of Innovation, Project Tuscany Health Ecosystem (THE), CUP: B83C22003920001.

References

[1] Chen, Xuyang, Xiaolu Zhang, Michael Elliot, Xiaoyin Wang, and Feng Wang. "Fix the leaking tap: A survey of Trigger-Action Programming (TAP) security issues, detection techniques and solutions." Computers & Security (2022): 102812.

[2] Coppers, Sven, Davy Vanacken, and Kris Luyten. "Fortniot: Intelligible predictions to improve user understanding of smart home behavior." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 4, no. 4 (2020): 1-24.

[3] Corno, Fulvio, Luigi De Russis, and Alberto Monge Roffarello. "A high-level semantic approach to end-user development in the Internet of Things." International Journal of Human-Computer Studies 125 (2019): 41-54.

[4] M. Funk, L. L. Chen, S. W. Yang, and Y.-K. Chen, "Addressing the need to capture scenarios, intentions and preferences: interactive intentional programming in the smart home," in International Journal of Design, vol.12, no. 1, pp. 53–66, 2018.

[5] Gallo, Simone, Fabio Paternò, and Alessio Malizia. "A conversational agent for creating automations exploiting large language models." Personal and Ubiquitous Computing (2024):1-16. doi: 10.1007/s00779-024-01825-5.

[6] Huang, Bing, Hai Dong, and Athman Bouguettaya. "Conflict detection in iot-based smart homes." In 2021 IEEE International Conference on Web Services (ICWS), pp. 303-313. IEEE, 2021.

[7] Kim, Sanghoon, and In-Young Ko. "A Conversational Approach for Modifying Service Mashups in IoT Environments." In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, pp. 1-16. 2022.

[8] Liao, Q. Vera, Daniel Gruen, and Sarah Miller. "Questioning the AI: informing design practices for explainable AI user experiences." In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1-15. 2020.

[9] Paternò, Fabio, "Humanations: A new understanding of human/automation interaction". In Proceedings of CHIGREECE '23. ACM, Article 1, 1–4.

[10] Gang Wu, Liang Hu, Yuxiao Hu, Yongheng Xing, Feng Wang, User intention prediction for trigger-action programming rule using multi-view representation learning, Expert Systems with Applications, Volume 267, 2025, 126198,ISSN 0957-4174, Elsevier

[11] Zhao, Valerie, Lefan Zhang, Bo Wang, Michael L. Littman, Shan Lu, and Blase Ur. "Understanding trigger-action programs through novel visualizations of program differences." In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1-17. 2021.