An Interactive IoT Automation Management Environment Supporting Transparency and Human Control

SIMONE GALLO, CNR - ISTI, Italy
ANDREA MATTIOLI, CNR - ISTI, Italy
FABIO PATERNÒ, CNR - ISTI, Italy

Smart home automation systems enable users to define personalised behaviours but often suffer from limited intelligibility, hidden dependencies, and conflicting rules, which hinder effective control and trust. We present **CASPER**, a tool designed to support the creation, inspection, and management of smart home automations while preserving user agency. The system integrates natural language interaction with a synchronised visual interface to expose system behaviour, detect automation issues such as conflicts, indirect activation chains, and missing revert conditions, and generate non-binding resolution proposals. The system is designed to enhance transparency by making internal reasoning and automation interactions explicit, and to support user control by ensuring that all changes remain user-driven. It frames assistant autonomy as augmentative rather than substitutive, allowing users to supervise and steer complex automation ecosystems. The conducted user studies indicate that this approach supports effective interaction with complex automations, achieving high usability and moderate cognitive workload, and shows how the system can improve intelligibility and control in smart home automation without reducing user agency.

CCS Concepts: • **Human-centered computing → Empirical studies in HCI**; **Ubiquitous and mobile computing systems and tools**.

Additional Key Words and Phrases: End-user development, Internet of Things, Automation, Conversational agents

ACM Reference Format:
Simone Gallo, Andrea Mattioli, and Fabio Paternò. 2026. An Interactive IoT Automation Management Environment Supporting Transparency and Human Control. In *Proceedings of AutomationXP26 Workshop of the 2026 CHI Conference on Human Factors in Computing Systems, April 14, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages.

1 Introduction

Smart home environments increasingly rely on automations to orchestrate the behaviour of connected devices. Trigger–Action Programming (TAP) has emerged as a widely adopted paradigm for this purpose [5, 21], enabling the specification of rules such as *"when motion is detected, turn on the light"*. Despite its effectiveness in simple scenarios, TAP-based systems become progressively difficult to understand and manage as the number of automations increases. In realistic deployments, multiple rules can interact in non-obvious ways, generating conflicts, cascading activations, and behaviours that diverge from users' expectations [6]. These challenges are further amplified by temporal dependencies, indirect environmental effects, and the absence of explicit support for long-term user goals such as energy saving, wellbeing, or safety.

Authors' Contact Information: Simone Gallo, CNR - ISTI, Pisa, Italy, simone.gallo@isti.cnr.it; Andrea Mattioli, CNR - ISTI, Pisa, Italy, andrea.mattioli@isti.cnr.it; Fabio Paternò, CNR - ISTI, Pisa, Italy, fabio.paterno@isti.cnr.it.

Manuscript submitted to ACM

From an End-User Development (EUD) perspective [18], limited intelligibility undermines users' ability to appropriate, debug, and evolve smart home environments over time. Previous work has proposed several approaches to improve automation understanding. Systems such as AppsGate leveraged dependency graphs to visualise relationships among automations and device states [11], while interrogative debugging and rule simulation techniques introduced mechanisms to support *why* and *why not* reasoning [9, 19]. Formal analyses based on graph-based or logic-based models have also been used to detect conflicts and unintended rule interactions [1, 4, 22]. Although effective in identifying specific classes of problems, these approaches typically focus on simplified rule structures and often treat debugging as a separate activity from automation creation.

More recently, conversational interfaces have been explored to lower the barrier to end-user programming in smart environments. Leveraging large language models, these systems allow users to define automations through natural language dialogue and iteratively refine them [14, 15]. Other solutions have targeted specific optimisation objectives such as energy efficiency or safety [15, 16], or combined natural language interaction with formal verification techniques [7]. However, most conversational systems primarily focus on rule creation, offering limited support for post-deployment understanding and troubleshooting. Explanations, when available, are often purely textual and detached from structured system representations, making it difficult for users to reason about interactions among multiple automations.

Explainability and goal-awareness have been recognised as fundamental requirements for user trust and control in intelligent systems [17]. In smart environments, intelligibility concerns understanding how interacting automations produce system behaviour rather than explaining algorithmic decisions [8]. Prior studies have shown that unintended behaviours and conflicts significantly reduce the perceived usefulness of smart home systems [3]. Furthermore, users often reason in terms of high-level intentions and long-term goals rather than individual device actions. While some research has investigated goal-aware automation and optimisation strategies [2, 10, 12, 13], the dynamic relationship between multiple automations and prioritised user goals remains insufficiently explored.

To address these challenges, we discuss the design choices and the lessons learnt in the development and evaluation of CASPER, an interactive environment that integrates visual and conversational interaction to support users throughout the entire automation lifecycle. The system goes beyond rule creation by automatically detecting conflicts, activation chains, and misalignments with long-term goals. These issues are not only identified but also explained and resolved through a tightly integrated multimodal interface.

2 System Overview

CASPER (see Figure 1) is designed as a unified environment that supports the creation, understanding, and evolution of smart home automations. Differently from traditional solutions that separate automation programming from troubleshooting, the system embeds detection, explanation, and repair of automation issues directly into the user workflow. This design aligns with EUD principles by supporting incremental refinement and reflection-in-action [20]. At a high level, CASPER integrates two complementary interaction modalities: a conversational agent and a visual dashboard. The conversational component enables users to express automation requirements in natural language, request explanations, and explore potential solutions through dialogue. This interaction paradigm builds upon recent research demonstrating the effectiveness of conversational interfaces in supporting end-user programming in smart environments [14, 15]. However, unlike prior conversational solutions that primarily support automation creation, CASPER extends dialogue interaction to explanation, debugging, and resolution of unexpected behaviours. To support these functionalities, CASPER adopts a multi-agent architecture coordinated by a central Main Agent. The Main Agent orchestrates the interaction flow, managing the dialogue with the user and dynamically invoking specialised sub-agents

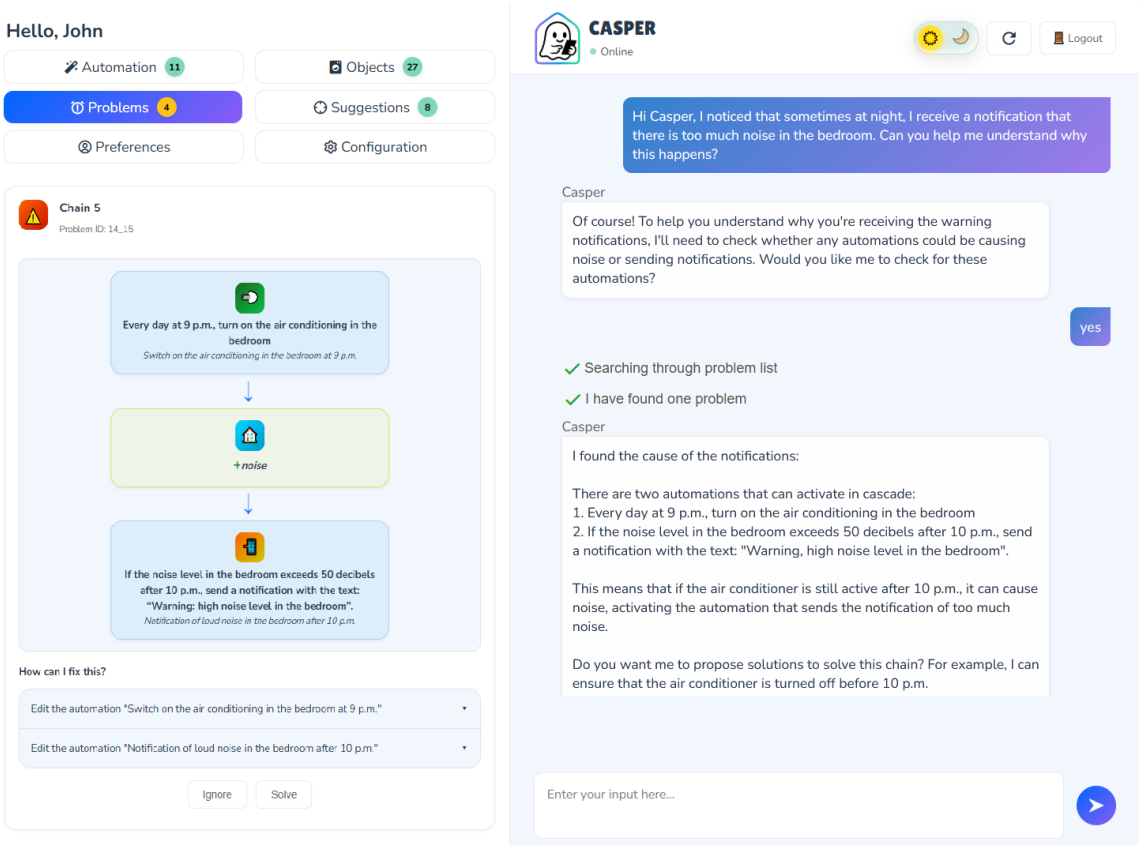

Fig. 1. CASPER Web Interface.

responsible for distinct tasks, namely Automation Generator, Issue Explainer, and Solutions Generator. Specifically, once the user defines the desired automation in a multi-turn conversation, the Main Agent activates the Automation Generation Agent. This component is tasked with producing the formal automation specification and may optionally return a clarification message to the user, for instance, if a requested device is not available within the current device set.

The remaining two sub-agents address the explanation and resolution of the detected issues. The Issue Explainer Agent receives as input a couple of automations identified as problematic by a detection algorithm and produces a contextualised explanation, which is presented both within the conversational interface and in the visual dashboard. Building upon this explanation, the Solution Generation Agent considers the available devices, the specification of the problematic automations, and the generated explanation to propose a set of modifications aimed at resolving the detected issue. Building upon the output of these agents, the visual dashboard provides structured representations of automations, detected issues, and suggested resolutions. Graphical representations make relationships among rules, devices, and contextual conditions explicit, supporting users in inspecting system behaviour and comparing alternative solutions. This approach extends earlier graph-based and inspection-based debugging techniques [9, 11, 19] by embedding them into a continuous interaction environment rather than providing them as standalone diagnostic tools.

When a user creates or modifies an automation through the conversational interface, CASPER translates the natural language request into a structured TAP rule and immediately performs automated analysis. The system detects three main classes of issues. First, it identifies conflicts between automations acting on the same devices, extending prior static and formal conflict-detection approaches [1, 4, 22] with contextual explanation mechanisms. Second, it detects direct and indirect activation chains that may produce unintended cascading behaviours. Third, the system performs goal-aware analysis by verifying the consistency between active automations and prioritised long-term user goals, building upon research investigating goal-oriented smart home automation [2, 10, 12, 13].

Detected issues are summarised conversationally and visualised as interactive problem cards. Each card provides an explanation of the detected problem and presents alternative resolution strategies. The conversational agent contextualises explanations, answering users' questions and clarifying system behaviour, while the visual interface supports exploration and comparison of solutions. This bidirectional coupling ensures that conversational explanations reference visual elements and that actions performed through the visual interface trigger follow-up conversational interactions. Users may rely on visual representations to obtain an overview of system behaviour or on conversational interaction to incrementally explore and resolve problems. This integration aims to improve intelligibility and control while reducing the cognitive burden associated with managing complex automation ecosystems.

3 User Feedback

Two complementary user tests were conducted to evaluate CASPER. The first study (11 males, 4 females; age range 22–42 years, $M = 25.67$, recruited from a Digital Humanities Master's course) investigated system usability and perceived effectiveness in a naturalistic setting, allowing participants to interact with the platform autonomously over an extended period. The second study (7 males, 9 females; age range 23–40 years, $M = 30$, recruited via mailing lists) focused on observing interaction strategies and usage patterns under controlled conditions, enabling detailed behavioural analysis. The studies aimed to explore the themes of **transparency** and **user control**. Both studies followed a similar task structure to ensure comparability while enabling different evaluation perspectives. Participants were provided with a preconfigured smart home environment containing sensors, devices, and an existing set of automations that included intentionally introduced conflicts, activation chains, and goal-related inconsistencies. Participants were required to complete a set of tasks involving automation creation, identification of undesired behaviours, resolution of rule conflicts, management of activation chains, and optimisation of goal-related behaviours.

3.1 Transparency

Implemented Mechanisms. CASPER supports transparency by explicitly exposing system behaviour and internal reasoning through both conversational and visual mechanisms. During multi-step operations, the conversational agent provides intermediate status messages that inform users about ongoing actions, such as retrieving automations, analysing specific rules, or generating new automations. These messages make system activity observable, reducing uncertainty and clarifying the progression of complex interactions. Transparency is further supported by structured explanations of detected issues, including conflicts, indirect activation chains, missing revert conditions, and goal inconsistencies. Each issue is accompanied by explanations that reference the involved automations, devices, and contextual conditions. These explanations are delivered conversationally and mirrored in the visual interface through dedicated *Problems* and *Suggestions* sections. The visual interface complements conversational explanations by externalising relationships among automations and devices, supporting inspection of system behaviour and potential consequences of modifications.

Conversational and visual components are synchronised, ensuring that explanations remain grounded in the current system state.

User Feedback. Across both studies, participants reported a high level of perceived transparency. In the laboratory study, users consistently described the chatbot's feedback as clear and helpful for understanding system behaviour, particularly during complex, multi-step tasks. Intermediate status messages were perceived as reassuring indicators of system activity. Quantitative results (SUS, NASA-TLX, and specific attribute ratings) confirm these perceptions. Participants rated both conversational and visual explanations as effective for understanding conflicts and activation chains, with visual explanations receiving slightly higher scores. This feedback was consistent across both the laboratory study and the autonomous study. Qualitative feedback indicates that transparency can be challenged when many issues are presented simultaneously. Several participants reported difficulties in scanning long lists of conflicts or suggestions and noted that dense textual descriptions or ambiguous goal labels occasionally reduced immediate clarity. This suggests that transparency depends not only on explanation availability but also on effective information organisation.

3.2 User Control

Implemented Mechanisms. User control in CASPER is preserved by design choices that keep users responsible for all system changes. While the system proactively detects issues and proposes solutions, no modifications are applied autonomously. Users can accept, modify, or ignore suggested resolutions. Control is further supported by the availability of conversational and visual interaction modalities. Automation creation is performed through natural language dialogue, while analysis and resolution can be carried out using either modality. This flexibility allows users to choose interaction strategies aligned with their preferences, expertise, and task complexity. The visual interface supports inspection and verification of system state, while the conversational interface facilitates iterative refinement and personalised guidance. Assistant autonomy is therefore framed as augmentative rather than substitutive.

User Feedback. Participants reported a strong sense of control when interacting with CASPER. In the laboratory study, the system achieved a very high SUS score ($M = 92.19$, $SD = 6.05$), indicating high perceived usability and confidence. No participant failed to complete the assigned tasks, and the overall task completion time (for eight tasks) averaged approximately 15 minutes, with more complex tasks requiring longer durations due to their intrinsic complexity. NASA-TLX results indicate a moderate workload, with Effort ($M = 25.63$) and Mental Demand ($M = 16.56$) emerging as the most prominent components, while Frustration, Temporal Demand, and Physical Demand remained low. This suggests that tasks required cognitive engagement without inducing stress or loss of control. Analysis of interaction modality usage shows that participants often began with the visual interface for initial exploration, while tasks involving automation creation and complex resolution were frequently completed through the conversational interface. Mode switches typically occurred when users encountered uncertainty and served as compensatory strategies to regain clarity. Notably, no participant switched from conversational interaction to visual interaction after establishing dialogue-based engagement. Qualitative feedback reinforces these findings, highlighting appreciation for the freedom to choose interaction modalities and for the chatbot's supportive tone. Some participants expressed a desire for stronger guidance when multiple solutions were available, pointing to a design tension between preserving user control and reducing decision effort.

4 Conclusions

In this paper, we discussed how CASPER integrates conversational interaction, visual inspection, and automated analysis within a single environment that supports the full lifecycle of smart home automations. By coupling issue detection

with contextual explanations and non-binding repair suggestions, the system makes rule interactions explicit while leaving decisions to the user. The evaluation highlights how this integration can sustain transparency and control when managing complex automation ecosystems. To validate our approach, we started an in-the-wild longitudinal study in real smart homes.

References

[1] Nayeon Bak, Byeong-Mo Chang, and Kwanghoon Choi. 2018. Smart Block: A Visual Programming Environment for SmartThings. In *2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23-27 July 2018, Volume 2*, Sorel Reisman, Sheikh Iqbal Ahamed, Claudio Demartini, Thomas M. Conte, Ling Liu, William R. Claycomb, Motonori Nakamura, Edmundo Tovar, Stelvio Cimato, Chung-Horng Lung, Hiroki Takakura, Ji-Jiang Yang, Toyokazu Akiyama, Zhiyong Zhang, and Md. Kamrul Hasan (Eds.). IEEE Computer Society, 32–37. doi:10.1109/COMPSAC.2018.10199

[2] Bernardo Breve, Gaetano Cimino, and Vincenzo Deufemia. 2023. Identifying Security and Privacy Violation Rules in Trigger-Action IoT Platforms With NLP Models. *IEEE Internet Things J.* 10, 6, March 15 (2023), 5607–5622. doi:10.1109/JIOT.2022.3222615

[3] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring End User Programming Needs in Home Automation. *ACM Trans. Comput. Hum. Interact.* 24, 2 (2017), 11:1–11:35. doi:10.1145/3057858

[4] 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 Trans. Cyber Phys. Syst.* 2, 3 (2018), 22:1–22:23. doi:10.1145/3185501

[5] Federico Cabitza, Daniela Fogli, Rosa Lanzilotti, and Antonio Piccinno. 2017. Rule-based tools for the configuration of ambient intelligence systems: a comparative user study. *Multim. Tools Appl.* 76, 4 (2017), 5221–5241. doi:10.1007/S11042-016-3511-2

[6] Xuyang Chen, Xiaolu Zhang, Michael Elliot, Xiaoyin Wang, and Feng Wang. 2022. Fix the leaking tap: A survey of Trigger-Action Programming (TAP) security issues, detection techniques and solutions. *Comput. Secur.* 120 (2022), 102812. doi:10.1016/J.COSE.2022.102812

[7] Ye Cheng, Minghui Xu, Yue Zhang, Kun Li, Ruoxi Wang, and Lian Yang. 2024. AutoIoT: Automated IoT Platform Using Large Language Models. *CoRR* abs/2411.10665 (2024). arXiv:2411.10665 doi:10.48550/ARXIV.2411.10665

[8] 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 (2020), 124:1–124:24. doi:10.1145/3432225

[9] 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 2019, Glasgow, Scotland, UK, May 04-09, 2019*, Stephen A. Brewster, Geraldine Fitzpatrick, Anna L. Cox, and Vassilis Kostakos (Eds.). ACM, 388. doi:10.1145/3290605.3300618

[10] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2021. From Users' Intentions to IF-THEN Rules in the Internet of Things. *ACM Trans. Inf. Syst.* 39, 4 (2021), 53:1–53:33. doi:10.1145/3447264

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

[12] Mathias Funk, Lin Lin Chen, Shao Wen Yang, and Yen Kuang Chen. 2018. Addressing the need to capture scenarios, intentions and preferences: Interactive intentional programming in the smart home. *International Journal of Design* 12, 1 (2018), 53–66. doi:ws/portalfiles/portal/96723920/A_2995_10715_3_PB.pdf

[13] Simone Gallo, Sara Maenza, Andrea Mattioli, and Fabio Paternò. 2025. Explaining Automations in IoT Settings. *IEEE Access* 13 (2025), 161370–161397.

[14] Simone Gallo, Fabio Paternò, and Alessio Malizia. 2024. A conversational agent for creating automations exploiting large language models. *Pers. Ubiquitous Comput.* 28, 6 (2024), 931–946. doi:10.1007/S00779-024-01825-5

[15] Mathyas Giudici, Luca Padalino, Giovanni Paolino, Ilaria Paratici, Alexandru Ionut Pascu, and Franca Garzotto. 2024. Designing home automation routines using an LLM-based chatbot. *Designs* 8, 3 (2024), 43.

[16] Md Shihabul Islam and Murat Kantarcioglu. 2025. A Tool for Safe and Accurate IoT Automation Rule Generation Using Large Language Models. In *2025 IEEE Security and Privacy, SP 2025 - Workshops, San Francisco, CA, USA, May 15, 2025*, Marina Blanton, William Enck, and Cristina Nita-Rotaru (Eds.). IEEE, 191–198. doi:10.1109/SPW67851.2025.00025

[17] Q. Vera Liao, Daniel M. Gruen, and Sarah Miller. 2020. Questioning the AI: Informing Design Practices for Explainable AI User Experiences. In *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, Regina Bernhaupt, Florian 'Floyd' Mueller, David Verweij, Josh Andres, Joanna McGrenere, Andy Cockburn, Ignacio Avellino, Alix Goguey, Pernille Bjøn, Shengdong Zhao, Briane Paul Samson, and Rafal Kocielnik (Eds.). ACM, 1–15. doi:10.1145/3313831.3376590

[18] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. *End-User Development: An Emerging Paradigm.* Springer. doi:10.1007/1-4020-5386-X_1

[19] Marco Manca, Fabio Paternò, Carmen Santoro, and Luca Corcella. 2019. Supporting end-user debugging of trigger-action rules for IoT applications. *Int. J. Hum. Comput. Stud.* 123 (2019), 56–69. doi:10.1016/J.IJHCS.2018.11.005

[20] Donald A Schön. 2017. *The reflective practitioner: How professionals think in action.* Routledge. doi:10.4324/9781315237473

[21] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical trigger-action programming in the smart home. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, Matt Jones, Philippe A. Palanque, Albrecht

Schmidt, and Tovi Grossman (Eds.). ACM, 803–812. doi:10.1145/2556288.2557420

[22] Lefan Zhang, Weijia He, Jesse J. Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. 2019. AutoTap: synthesizing and repairing trigger-action programs using LTL properties. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE / ACM, 281–291. doi:10.1109/ICSE.2019.00043