Understanding In-Situ Programming for Smart Home Automation

XIAOYI LIU* and YINGTIAN SHI*, Department of Computer Science and Technology, Tsinghua University
CHUN YU†, Department of Computer Science and Technology, Tsinghua University
CHENG GAO, Department of Computer Science and Technology, Tsinghua University
TIANAO YANG, Department of Computer Science and Technology, Tsinghua University
CHEN LIANG, Department of Computer Science and Technology, Tsinghua University
YUANCHUN SHI, Key Laboratory of Pervasive Computing, Ministry of Education, Department of Computer Science and Technology, Tsinghua University, and Qinghai University

Programming a smart home is an iterative process in which users configure and test the automation during the in-situ experience with IoT space. However, current end-user programming mechanisms are primarily preset configurations on GUI and fail to leverage in-situ behaviors and context. This paper proposed in-situ programming (ISP) as a novel programming paradigm for AIoT automation that extensively leverages users' natural in-situ interaction with the smart environment. We built a Wizard-of-Oz system and conducted a user-enactment study to explore users' behavior models in this paradigm. We identified a dynamic programming flow in which participants iteratively configure and confirm through query, control, edit, and test. We especially identified a novel method "snapshot" for automation configuration and a novel method "simulation" for automation testing, in which participants leverage ambient responses and in-situ interaction. Based on our findings, we proposed design spaces on dynamic programming flow, coherency and clarity of interface, and state and scene management to build an ideal in-situ programming experience.

CCS Concepts: • **Human-centered computing** → *Ambient intelligence*; *Interface design prototyping*.

Additional Key Words and Phrases: Smart Home, End-User-Programming, automation, Internet of things

ACM Reference Format:
Xiaoyi Liu, Yingtian Shi, Chun Yu, Cheng Gao, Tianao Yang, Chen Liang, and Yuanchun Shi. 2023. Understanding In-Situ Programming for Smart Home Automation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 66 (June 2023), 31 pages. https://doi.org/10.1145/3596254

1 INTRODUCTION

Recent years have witnessed the development of AIoT (Artificial Intelligence of Things) enabled smart homes, bringing new opportunities for smart home automation. A plethora of smart devices, AI-based intelligent agents,

*Both authors contributed equally to this research.
†Corresponding author.

Authors' addresses: Xiaoyi Liu, liu-xy20@mails.tsinghua.edu.cn; Yingtian Shi, shiyt0313@gmail.com, Department of Computer Science and Technology, Tsinghua University; Chun Yu, Department of Computer Science and Technology, Tsinghua University; Cheng Gao, gaocheng20@mails.tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University; Tianao Yang, yta20@mails.tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University; Chen Liang, lliangchenc@163.com, Department of Computer Science and Technology, Tsinghua University; Yuanchun Shi, Key Laboratory of Pervasive Computing, Ministry of Education and Department of Computer Science and Technology, Tsinghua University and Qinghai University.

2474-9567/2023/6-ART66
https://doi.org/10.1145/3596254

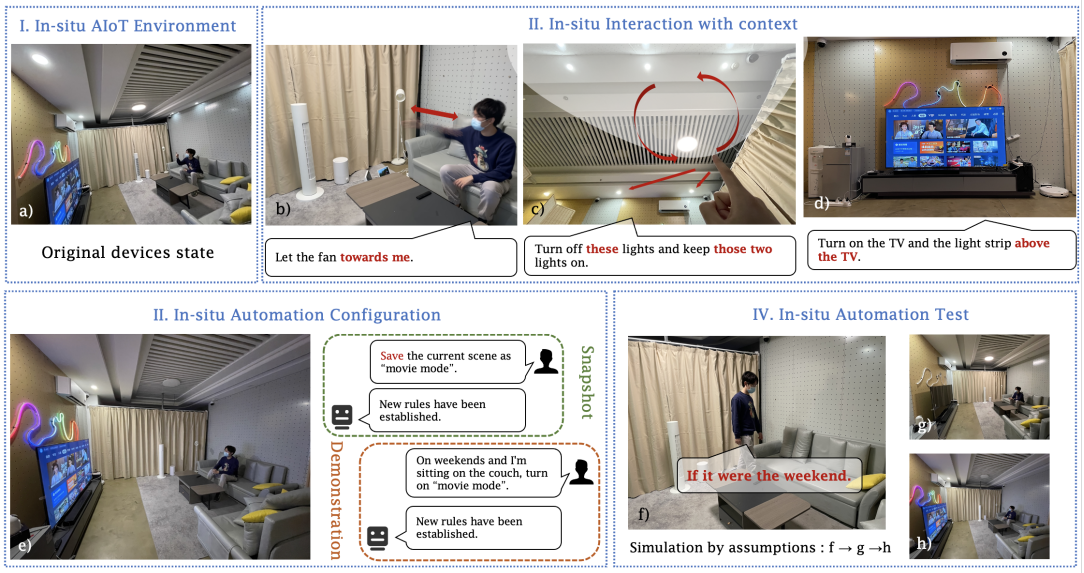

Fig. 1. Example of a user using the in-situ programming (ISP) system to configure a movie mode. Fig. a) shows the experimental environment for ISP and the device state before programming. Fig. b) c) d) show that the user uses multimodal (gesture, voice) and context (position, posture) to control the device to complete the in-situ interaction. Fig. e) shows the scene after configuration and respectively shows the process of automation programming by using "snapshot" and demonstration approaches. Fig. f) g) h) shows the process of the user using the "simulation" approach for automation testing. After the user sits on the coach, the rule is triggered, and the scene changes from g) to h).

and end-user programming (EUP) techniques empower end-users with a ubiquitous programmable IoT ecosystem in which they can tailor automation services based on household routines [96]. Smart home assistants such as Amazon Alexa [1], Apple HomeKit [4], Google Assistant [2], and task automation tools (TA) [41] like IFTTT [5], and Zapier [6] allow users to connect various smart devices and web services via smartphones or PCs. Generally, users configure IoT automation based on the rule-based trigger-action programming (TAP) paradigm [122], in the form of "IF *triggers*, THEN *actions*".

Despite the TAP paradigm being easy to learn and use [122], it has limitations in user experience [43, 66, 122]. Particularly, we identify two issues:

- Unnatural Programming Interaction: In the actual smart home experience, users express their needs through multi-model interfaces and natural behaviors (e.g., language, gestures, etc.). However, contemporary paradigms commonly adopt a fixed structure and workflow through a graphic user interface on smartphones or PCs, which is inconsistent with users' natural and intuitive interaction patterns in smart homes.
- High Programming Cost: The increasingly growing IoT ecosystem makes programming IoT automation more demanding. It set challenges for users to deal with relationships of various entities (i.e., people, devices, environment) and complex real-life scenarios. While contemporary paradigms commonly rely on users to anticipate and pre-define the automation, which brings programming cost to design, configure and debug [123].

Thus, the contemporary programming mechanisms separate the programming process from the actual interaction in smart homes, failing to utmost utilize the contextual information and human behaviors in IoT space to facilitate natural and low-cost programming.

To overcome these issues, we claim that a new IoT-related programming paradigm should be designed and coordinated with the smart home experience. Under this vision, we proposed in-situ programming (ISP), as a novel programming paradigm that allows users to program smart home automation with in-situ contextual information and interaction patterns. The goal of ISP is to allow end–users to program smart environments in an immersive and seamless way with ubiquitous programming interfaces. We aim to frame the design spaces of this paradigm by identifying users' intuitive behavior patterns and interaction models as the optimal design references.

In this paper, we first introduced the concepts, the roles, and the general design goals of ISP. Then we built a Wizard-of-Oz system prototyping based on our design goals and state-of-art end-user programming technology as our research environment. Furthermore, we conducted a user study with the following research questions: 1) How do users interact with the system during in-situ programming? 2) How do users intuitively understand in-situ context and leverage in-situ behaviors in smart home programming? 3) How do in-situ features affect the programming model and facilitate the programming process?

We found that the participants go through a dynamic programming flow, and iteratively configure and confirm by four interaction nodes: query, control, edit, and test. We especially identified a novel method "snapshot" for configuration and a novel method "simulation" for testing, in which participants leverage ambient responses and in-situ interaction. We further proposed 3 design spaces for ISP: 1) dynamic programming flow, 2) coherency and clarity of interface, 3) state and scene management.

The contributions of our work include:

(1) We propose the in-situ programming (ISP) paradigm in IoT scenarios which greatly leverages in-situ contextual information and user interaction. We systematically investigate users' behavior models and mental models under this paradigm.
(2) We frame the programming flow and approaches in an ISP scenario. We proposed two novel approaches: 1) a configuration approach "snapshot" in which users configure automation by in-situ controlling devices and preserve the instant states. 2) a test approach "simulation", in which users test the rules with assumptions.
(3) We design and prototype a Wizard-of-Oz system to support the ideal ISP scenario. We validate the system's effectiveness through a user-enactment study and proposed design spaces on programming mechanism, interface, and scene management.
(4) We collect multi-modal data from the user study and form a text-based dataset of user-system interaction annotated with interaction features, acts, and intentions for IoT end-user programming research.

2 RELATED WORKS

2.1 In-Situ Interaction in Smart Homes

Considerable research highlighted in-situ interaction with the ubiquitous ambient intelligence [11, 62, 64, 101]. Basically, there are two key features of in-situ smart home interaction.

One is a synthesis of pervasive multi-modality interfaces beyond traditional graphic user interfaces (GUI) on smartphones [38], including voice interface [124], tangible interface [14], gesture interface (movement, motion) [60, 71, 88], etc. These modalities are further applied in situated interaction approaches, e.g., pointing to, facing, or gazing at devices [25, 78, 107] accompanied by spatial descriptions verbally [77]. In recent years, smart speakers with intelligent virtual agent [18] has become the most popular entrance for smart home control [23, 45, 102], which allows users to ask for contextual information (e.g., the weather or temperature), and control IoT devices [99] and automation [117], and compose simple rules [13], e.g., Google Assistant [2], Amazon Alexa [1]. Literature studied how users communicate with intelligent systems in smart homes [35, 69] by articulation, simplification, supplementation, and correction [80, 100]. Chiang et al. [35] reported that users dynamically changed communication strategies depending on the current confidence [53] of the system.

The second feature is the utilization of contextual information. The context in a smart home is defined as any information that can be used to characterize the situation of an entity, including the person, environment, and devices [7, 44, 52, 101, 120]. Past literature proposed spatial sensing techniques and context-awareness [65, 97] computing paradigms into smart home architecture to sense and reason in real-time and situated contexts. Further, this contextual information is explored to enhance in-situ interaction, especially in spatial and human-related contexts. [120] studied spatial expressions including topological terms (on, in), distance-related terms (close, far), and path-related terms (across, through). Others [17, 95] proposed spatial interaction approaches based on proximity, orientation, location, and movement. These in-situ approaches are believed to enhance natural and seamless smart home interaction.

2.2 End-User Programming Paradigms in Smart Homes

End-user programming (EUP) is a set of methods, techniques, and tools that aim to empower end users, especially non-technical users' program software artifacts [19, 89]. In the smart home context, EUP enables end users to program a smart environment by automating smart devices and services [20]. For decades of development, the most widely used EUP paradigm is the rule-based Trigger-Action Paradigm (TAP) [59, 66, 122, 123]. It is embedded in popular automation tools such as IFTTT [5], in which users set triggers and actions of the rule by defining how a set of device behaviors respond to certain events [59]. Although literature [20, 39, 122] has proved that TAP is intuitive, understandable, and easy to use, smart home automation is mostly adopted by pilot users with expertise knowledge [31, 98, 106, 110, 112]. Non-technical users (i.e., passive users) still found the programming process demanding and tedious with high programming costs and unnatural programming mechanisms [43, 66, 122].

Hence, a large body of literature worked on supporting users to better understand and express TAP-based rules. One direction is to improve the expressiveness of TAP in complex tasks. Huang and Cakmak [66] investigated the mental model accuracy of different triggers (states and events) and actions (instantaneous, extended, and sustained actions) to improve its expressiveness in rules with multiple triggers and actions. Event-Conditional-Action (ECA) paradigm, i.e., IF (trigger), WHILE (condition), THEN (action), is found more intuitive and extends TAP by incorporating multiple conditions, exceptions, and logic operators to rule specifications [20, 50, 105].

Another direction is to leverage natural user expressions like language and behavior in TAP to facilitate natural programming mechanisms. [37, 61] outlined the importance of natural language programming. [15, 39, 56] incorporated context-awareness capability into TAP to support semantic programming parameters, like "darker" and contextual intentions. InstructableCrowd [67], Heytap [40] and Convo[125] developed conversational agents (CA) which facilitated users to composite IF-THEN rules by specifying their needs or the contents of rules.

Recent works also explored voice-based EUP to replace traditional smartphone-based GUIs [13, 18, 27, 33, 37]. Van Brummelen et al. [125] stressed the criticalities of a voice-based approach and the issues of flexibility, transparency, and cognitive load. De Russis et al. [48] found that users composed rules vocally in a similar trigger-action format and rephrased the rules until it is correctly understood by the system. [35, 45, 47, 76, 79, 82, 94, 118] explored approaches to composite or modify rules via smart speakers with natural language (e.g., If it is 7 am turns on the light). Others [15, 15, 56, 68, 91, 128] incorporated user traces of behaviors (activities, operations) as input of TAP rules. Programming by demonstration (PBD) [24] approach is applied in IoT programming [36, 51, 82–84], in which users directly manipulated physical devices and demonstrated natural instructions with either or both mobiles and tangible interfaces.

2.3 State of Art of In-Situ Smart Home Programming

In-situ programming paradigms are widely explored in end-user robot programming [10], which refers to programming robot behaviors situated in the task context [113]. Basically, these approaches are related to

tangible interfaces, contextual information, spatial interaction via situated instruction, and deictic gestures [55, 72, 113, 114]. In smart homes, in-situ programming paradigms allow users to configure the IoT automation grounded on the actual using scenarios and IoT environment [74, 109]. Literature highlighted in-situ programming style benefits users with consistent and seamless user experience [74, 82, 90] and a direct understanding of IoT settings [16]. Ariano et al. [16] propose an augmented reality solution to provide situated visualization, monitoring, and programming experience among various IoT devices. However, the switch between the screen and the scene may interfere with the programming process [82]. Some voice-based EUP methods applied in-situ contextual information to improve expressiveness and usability. For example, Jarvis [79] supports users to set non-trivial TAP rules with time-based, event-based, contextual awareness, and causality queries. CoMMA [76] incorporated disambiguation strategy into modification dialogues such as checking parameters and modifying positions, etc. Despite hand-free flexibility [13], the sole voice interface lacks information from users' situated interactions.

Summarily, the above explorations suggest the feasibility and significance of a natural programming paradigm grounded on in-situ change and interaction. While the state-of-art [16, 76, 79] approaches propose easy-to-learn and workable programming solutions, it is still an open question if these paradigms aligned with users' natural behavior models and mental models. To the best of our knowledge, we are the first to use the Wizard-of-Oz approach to investigate the paradigm and natural user behavior characteristics of in-situ programming in IoT scenarios. We focused on the explore the unique mechanisms and patterns when users can leverage more in-situ context and behaviors when programming the smart environment.

.

3 OVERVIEW OF IN-SITU PROGRAMMING PARADIGM

This section has an overview of in-situ programming (ISP). We first introduced the concepts, roles, and general design goals, then we design and prototype the ISP system with the Wizard-of-Oz techniques [43].

3.1 Concepts, Roles and Design Goals

In-situ programming (ISP) is defined as a programming paradigm featuring in-situ interaction with the smart environment that can be embedded into the Trigger-Action paradigm (TAP) [122]. The umbrella term in-situ (the abbreviation of in-situation) refers to "on-site" or "in-place", meaning the prerequisite that the user is programming in the IoT environment to be programmed. ISP is built upon ruled-based TAP, so the semantic space that ISP can represent would not exceed that of TAP. The role of ISP focuses on the mechanisms to construct and modify the rules more naturally, seamlessly, and easily for users in an in-situ programming scenario (Fig. 1).

The design goals of ISP are proposed based on the analysis of the relevant literature and extend previous in-situ EUP solutions in three aspects:

- on-the-fly: It allows live programming [9, 119] with real-time feedback and dynamic control flows. It supports rapid rule editing, iterating, testing, debugging, and learning.
- in-place: It allows situated programming, leveraging in-situ smart home interaction and situated context information. It supports programming in the consistent task context where the program is actually executed.
- multi-modality: It allows multi-modality programming consistent with users' interaction patterns in a smart home. It supports multi-modal input and feedback via voice, gesture, and tangible interface.

ISP comprehensively coordinates smart devices (D, including device states and events), human behavior (B, including voice, gesture, and tangible operation), and time-spatial context (C, including time, location, orientation, spatial relation. etc.) in programming smart home automation. The formulation of the traditional TAP rule (r) achieved by one-time voice-based editing can be represented as $r = SI(V) = [Tr(V) \rightarrow Ac(V)]$, where SI, Tr,

Ac, respectively represent the semantic interference module, the trigger, and the action extracted from the voice command as defined in the TAP. In ISP, we extend this formula into:

$$r_{new} = SI(V, r_{old}, G, T, C) = [Tr(V, r_{old}, G, T, C) \rightarrow A(V, r_{old}, G, T, C)] \quad (1)$$

where r_{old} represents either the last version of the rule or a null rule. Under such a paradigm, three unique specifications of ISP can be derived: 1) the introduction r_{old} allows a comparative (r_{new} v.s. r_{old}) and iterative editing strategy (e.g., optimizing r_{new} with multiple rounds of interaction); 2) the introduction of in-situ context C, can be leveraged to represent spatiotemporal information and relations between entities (human, devices, environment); 3) multi-modality input like gestures G and tangible operations T allows the user to expand their input channel with better expressiveness.

3.2 Designing and Prototyping Wizard-of-Oz In-Situ Programming System

In order to rapidly verify the ISP and explore users' natural behavior, we constructed a high-fidelity living room smart home scenario (Fig. 4) and designed a Wizard-of-Oz system [43] to support users completing in-situ programming under our design goals. The participation of wizard ensures that the response of the experimental system has higher flexibility to support users' rich interaction so that the users can focus on how they prefer to interact with ISP, rather than how to adapt the ISP. The scenario contains 16 popular smart home devices, including a TV, an air conditioner, a sweeping robot, a fan, a humidifier, a TV speaker, two smart speakers, five spotlights, one ceiling light, one light strip, and a lamp. Detailed device list is in Appendix.A.3.Table 2.

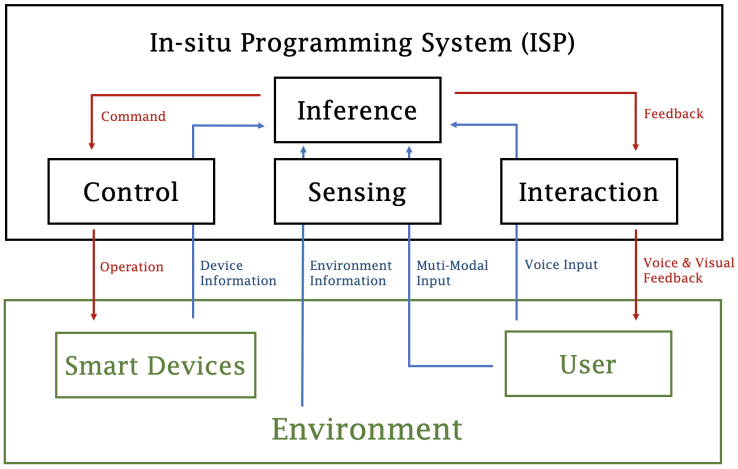

Fig. 2. In-situ programming (ISP) system architecture diagram. Black boxes represent ISP system components, and green boxes represent programming environments and users. The blue lines represent the user and environment inputs to the system, and the red lines represent environmental control and user feedback provided by the system.

The in-situ programming system is built in the smart home environment, which completes the management and control of smart devices by interacting with users and combining environmental information. Based on our design goals and related literature, we divided the Wizard-of-Oz system into four parts: interaction, sensing, control, and inference (Fig.2). The control part is responsible for device management and control, the sensing part is used for environmental context information perception and some user behavior sensations, and the interaction

part is responsible for communicating with users to assist them in completing programming. The inference layer is responsible for information integration, recognizing interaction intent, and establishing rules.

3.2.1 Control. The control part is responsible for device control, as well as automation control and generation. We connected and controlled all the smart devices with the Home Assistant Platform [3] via Bluetooth or Wi-Fi. We wrote software that standardizes the control interface of all devices and can obtain and store the states of the devices. When users performed automation settings, we saved the trigger and action contents in the system and recorded them as a shortcut command. The experimenter will input the command to the script on the PC to execute the action when the triggers are met, to provide users real-time automation response. The saved contents are organized in the form of Trigger-Action rules in the automation database. Fig.3.a shows the software architecture of the Wizard-of-Oz system.

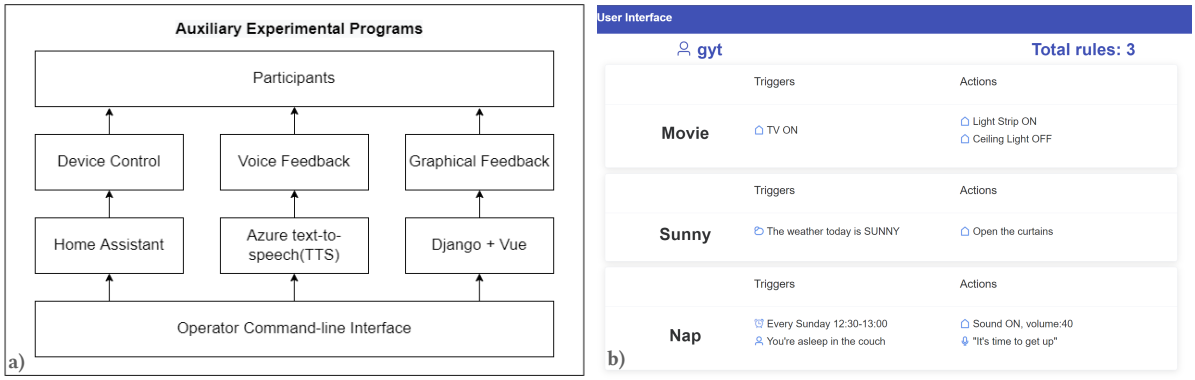

Fig. 3. Software settings. a) Software architecture of the auxiliary experimental system. b) The visual feedback interface.

3.2.2 Interaction. The interaction part is responsible for multi-modality interaction with the users, collecting active input information and providing real-time feedback.

For the voice interaction, we set up an intelligent agent named XiaoCheng in the system. We settled on basic design principles and commonly used feedback scripts, including greetings, control, queries, error responses, and replies related to automation settings (configuration process, rule broadcasting). For improvised dialogues on the spot, we reserved a text input interface so that the experimenter can directly convert the input text into speech feedback to the user. During the experiment, the PC running the script was connected to the smart speaker in the experimental environment via Bluetooth.

For tangible interactions, we linked BlueTooth physical switches to the lights in the room. We put all the tangible buttons on the tea table so that users can easily control the lights. In addition, the user can directly control the device through the original remote control and the physical buttons on the device. The input from the tangible interface is synchronized in the control console.

For visual-based interaction, we build a graphic user interface (GUI) to display the status and content of the rules (Fig.3.b). The rules are synchronized from the automation database and structured into trigger-action rules with concise language.

Based on our design goals and the previous literature on in-situ smart home interaction, the programming interfaces should be flexible, real-time, and multi-modal. Through the user study, we supported: 1) voice input with XiaoCheng, 2) tangible input through tangible buttons and remotes, 3) posture and gesture input through

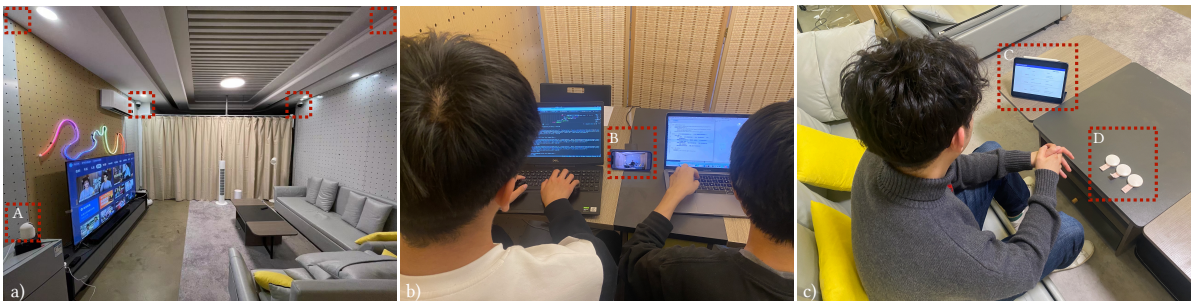

Fig. 4. Physical configuration of the user study: a) The physical environment. The red dotted box marks the location of the recording cameras. Box A marks the real-time video stream. b) The wizard perspective. The left wizard conducts system control. The right wizard conducts voice feedback. Box B displays the video stream from Box A. c) Interface setting. Box C is the GUI display. Box D is the tangible buttons of lights.

wizard real-time observation through a live camera. For real-time feedback, we provided 1) voice feedback through XiaoCheng which can broadcast the rules, 2) ambient feedback which can display the rules with the ambient response, 3) visual feedback through a tablet on the table which displays the rule.

3.2.3 Sensing and Inference. The sensing part is responsible for perceiving users' explicit multi-modal expression and implicit in-situ context information.

The Inference part comprehensively analyzes the information the remaining three parts provided, extracts high-level semantics and understands user intent, obtains parameters for automation generation, and provides information to the interaction and control parts. Specifically, the inference part analyzed multi-modality interaction, situated context, and previous automation, to support current automation generation and system responses.

Since the capacities of sensing and inference significantly impact the experiment, we use the wizards' observation and inference to replace system recognition and inference. During the user study, the wizards observed the participants through a live camera stream and gave corresponding voice and ambient responses. When the experimenter cannot accurately judge the users' intention, the users will be prompted by voice, asking them to repeat the request to ensure that users' needs can be accurately understood.

4 USER STUDY

In this section, we conducted a user study to explore behavior models and mental models under in-situ programming (ISP) scenarios based on the Wizard-of-Oz system described in Section 3.2. We aimed to explore the following research questions: **RQ1: How do users interact with the system during in-situ programming? RQ2: How do users intuitively understand in-situ context and leverage in-situ behaviors in smart home programming? RQ3: How do in-situ features affect the programming model and facilitate the programming process?**

4.1 Participants

16 participants (7 males, 9 females) with an average age of 24.1 (SD=1.93) were recruited through an online HCI recruiting notice. Each was compensated 150 - 200 Chinese Yuan based on their experiment time. Through a background survey, we collected participants' demographic information, residential situation, profession level of programming experience, smart home experience, and IFTTT experience. The participants' information is illustrated in Table 1. We presented the basic information such as age, gender, educational background, and

Table 1. Information of Participants

ID	Genders and ages	Educational background	Programming ex	Smart Home ex	IFTTT ex
P1	Female,24	Design	0	1	1
P2	Male,26	Electrical engineering	1	4	4
P3	Female,25	Economics	0	2	1
P4	Female,26	Design	0	1	1
P5	Male,24	Computer science	1	2	1
P6	Male,26	Computer science	1	3	2
P7	Female,23	Design	0	2	1
P8	Male,24	Economics	0	2	2
P9	Female,25	Design	0	3	3
P10	Female,23	Network sciences	1	2	1
P11	Female,24	Design	0	1	1
P12	Female,26	Computer science	1	2	1
P13	Male,18	Electrical engineering	1	3	1
P14	Male,24	Computer science	1	4	1
P15	Female,24	Design	1	4	1
P16	Male,24	Computer science	1	2	1

whether they have programming experience (labeled as 0 or 1). Their smart home experience (scored 1-5) was self-evaluated based on their smart home usage years, the number of devices, frequency of use, and interaction methods. IFTTT experience score (scored 1-5) is self-evaluated based on their familiarity with IFTTT, the number of configuration rules and experience in writing automation, etc.

Summarily, 9 of the participants have co-occupants such as parents (N=2) and peers (N=7), and 7 of them live alone. Half of them have technical backgrounds skilled in programming and half are non-technical backgrounds, with 1 of them having programming knowledge. Participants scored their smart home and IFTTT experience on a scale of 1-5 (1 means no relevant experience, 5 means very familiar). The majority are familiar with smart home experience (N=13) but with no IFTTT experience (N=12).

4.2 Scenario and Task Design

We designed ISP scenarios and tasks focusing on in-situ settings and context. We conducted a user-enactment process [35, 103], in which we asked the participants to intuitively act in the given scenarios which are structured in the dimensions we are interested in. We distilled user behaviors by observation and a following semi-structured interview.

We totally design 12 scenario scripts (Fig.5), equally divided into 2 groups. Both groups were used equally. The scripts are designed under the analysis of the relevant EUP literature and popular IoT-related rules from IFTTT website [5], grounding to our experimental ISP system. To investigate the performance of ISP in non-trivial rules, we choose more in-situ tasks with multiple actions. Besides, to utmost support participants to act more naturally and openly, we do not give concrete rule contents but only the scenario description. Participants are allowed to create the rules based on their understanding of scenarios and the rule varies from person to person. Hence the user-created rules may contain more triggers and actions than the requirements set. To ensure the task covers the features we focused to explore, we set task requirements and assigned the features within each group. The chosen features include:

- Action complexity: Rule complexity is a key factor to investigate the paradigm expressiveness and affects users' behavior in conceiving and expressing. We choose action complexity to better relate the task to the situated scenario. We choose the action features based on the research on rule complexity [29, 66, 79], including the numbers of actions, the numbers and varieties of involved devices in actions, the temporal-constrained actions, and the conditional branches of actions. The temporal-constrained action [66] involves temporal information in action expression (such as delayed, period, repetitive [79]). Conditional branches refer to the transmission of rule action flows under multiple triggers [50]. We scale the action complexity into 3 levels: Level I: single action or single-type devices configuration. Level II: multiple types of devices configuration or temporal-constrained actions. Level III: with conditional action branches.
- In-situ entities: The types of in-situ entities (person, devices, environment) may lead to different manners to leverage contextual information and interaction [16]. This dimension is to discover how different entities are interpreted and configured and we ensured that both groups cover all the entity types in triggers or actions. Specifically, both the script groups cover all the in-situ devices and at least involved 3 rules related to in-situ human states or activities, and 3 rules related to in-situ environmental context (e.g., time, temperature).
- Sequential/Parallel actions. This dimension describes the temporary relationships of multiple actions, respectively referring to the actions that are expected to execute in sequential and non-sequential orders [42]. We aim to discover whether and how users manage the distributed devices differently under a process-oriented and state-oriented scenarios. The two groups both contain these two types of rules and the total rule numbers of each type are equally 3.

4.3 Procedure

The study was composed of three sessions: an introductory session, an enactment session, and an interview session. The whole process lasted around 60-90 minutes and was recorded by the cameras. Each participant was invited to the Wizard-of-Oz environment with two wizards and one moderator' participating (see Fig.6 and Fig.4). One of the wizards is responsible for equipment control and interface feedback, and the other is responsible for automation management and voice feedback (Fig.4.b).

In the introductory session,the moderator introduced the basic concepts and instances of smart homes, automation rules, and our user study goals. Then the moderator detailed introduced the system, including smart device functions, the intelligent agent XiaoCheng, and multi-modality interfaces (introduction scripts see Appendix.A.1). The participant was then given 10 minutes to get familiar with the system during which the moderator gave necessary demonstrations and instructions.

Then the participant was assigned a script group and started the enactment session. In order to keep the participant immersive in the situated environment rather than the script document, the moderator broadcast the script one at a time off-site. The participant was asked to enact the scenario and create a rule through intuitive and natural behaviors. After the given 6 tasks in each group, the participant was allowed to freely configure rules as he/she wished. The moderator observed participants through the real-time video stream and recorded the issues we cared about.

After the enactment session, we immediately conducted a semi-structured interview. We first asked open questions on the specific issues we observed during the enactment. To investigate the underlying mental model, we asked the participant to retrospect the detailed process, the chain of thoughts, and opinions of specific behavior features and methods they applied. We avoided concluding for participants and provided the video recording for users to recall. Then we concluded with structured questionnaires (Appendix.A.2), investigating factors they care about, preferences for the programming paradigm (interaction flow, interface, approach, and feedback), benefits and drawbacks of in-situ programming, and expectations of system improvement. Finally, each participant rated

User-Enactment Scripts								
ID	Scenario	Task	Comple-xity	In-situ Entity Features			Action Features	
				Device	Human	Environ-ment	Parallel	Sequential
Group A								
T1	You usually go straight to bed and sometimes forget to turn off some of the appliances in the living room.	Create a rule to help you turn off appliances when you go to bed.	1	√	√			
T2	Every time you open the curtain, you have to manually turn off the lights in the living room.	Create a rule to help you turn off the lights when you open the curtain.	1	√		√		
T3	Your requirements for the living room environment are similar every day.	Create a rule for your living room environment. Include at least **light** and **temperature** settings.	2	√			√	
T4	As a movie lover, you watch a movie every weekend. You wish to make the living room more immersive for movies.	Create a rule to adjust the atmosphere when watching movies. At least include the **light** and **multimedia** settings**.**	2	√	√		√	
T5	When you come home from work every Friday, you wish to make your home welcome you.	Create a rule to make the devices respond to the **human activities sequentially.** Include at least **2 kinds of human states.**	3	√	√	√		√
T6	You wish to have a good rest every weekend afternoon.	Create a rule to set the living room environment according to the weather or the people in the house. Include at least **2 conditional branches.**	3	√	√	√		
Group B								
T7	Every time the curtain is closed, it will block the sweeper robot.	Create a rule to open the curtain when the robot is working.	1	√				
T8	You wish the living room temperature can be adjusted automatically.	Create a rule to adjust the air conditioning according to the living room temperature**.**	1	√		√		
T9	You are used to taking a 10min nap on the couch at noon every day.	Create a rule for your lunch break. Set the **start and end of the lunch break,** and make the devices wake you up **after a period of time.**	2	√	√	√		√
T10	You often read on the sofa and you wish the living room environment to be more suitable for reading.	Create a rule to adjust the atmosphere for reading. Include at least the **light and music** settings.	2	√	√		√	
T11	You often stay up late watching TV and feel tired at work the next day.	Create a rule to remind you go to bed early. Make the reminder changes **acording to your different responses over time.** Include at least 2 situations.	3	√	√	√		√
T12	You like to play smartphones or watch TV or take a nap on the sofa, but it is troublesome to adjust the environment manually.	Create a rule to automatically adjust the environment according to your activities. Include at least **2 kinds of activities**.	3	√	√			

Fig. 5. Task List of User-Enactment: Each script contains a scenario and a task. The scenario describes the background to enact. The task provides task goals and requirements.

the immersion of the study based on whether the study environment affected their natural expression (1-5, immersion rate from lowest to highest).

4.4 Data Analysis

Through the study, we collected a total of 42 hours of video footage and 213 rules. After cutting off the irrelevant parts, we finally got 33 hours of user behavior footage and 9 hours of interview footage. We conducted three rounds of analysis based on inductive approaches [121] including open coding [75] and affinity paradigm [92].

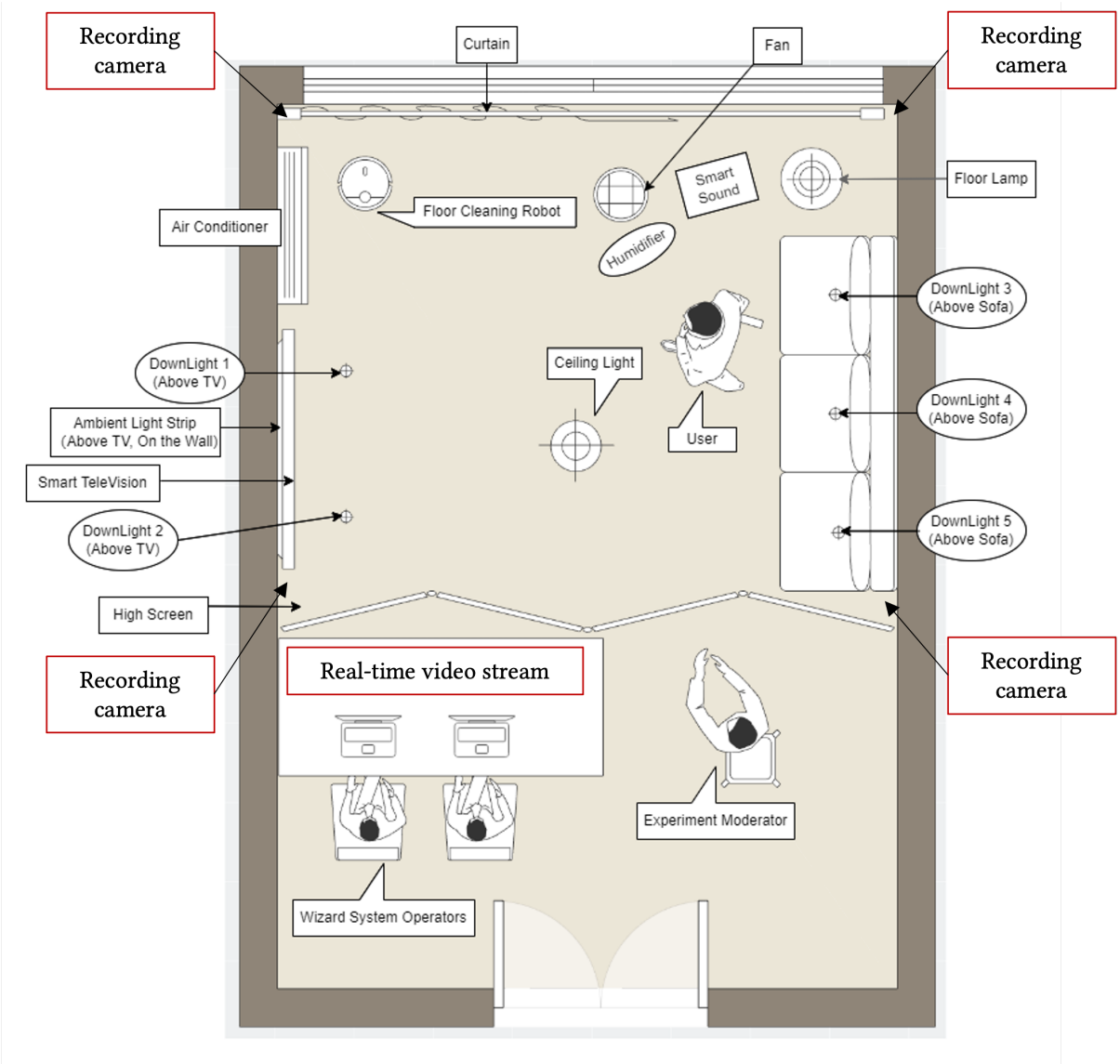

Fig. 6. User Study Configuration

Round 1: For the rule analysis, we labeled the rules features including the trigger counts, action counts, involved device counts, in-situ features, and human-related features.

For the behavior footage, four researchers independently watched the video, recording, and labeling every step of the user's activities. We defined one step as an acting unit (AU) which refers to a coherent and self-contained user act recognizable in context as a complete interaction with an independent agent. For each AU, we labeled the objective interaction characteristics, including interaction objects (device, agent. etc.), interaction modalities (voice, gesture, posture, tangible), concrete interaction contents (utterances, gestures, movement,

towards, etc.), feedback (voice, GUI, ambience.etc) and direct intentions (control devices, configure triggers..etc). For the interview material, the audio was transcribed, aligned to the corresponding AU, and labeled by open-code keywords.

We constructed a dataset based on experimental data and annotation results. An example of the annotation is shown in Fig.7. Each piece of data contains eight fields.

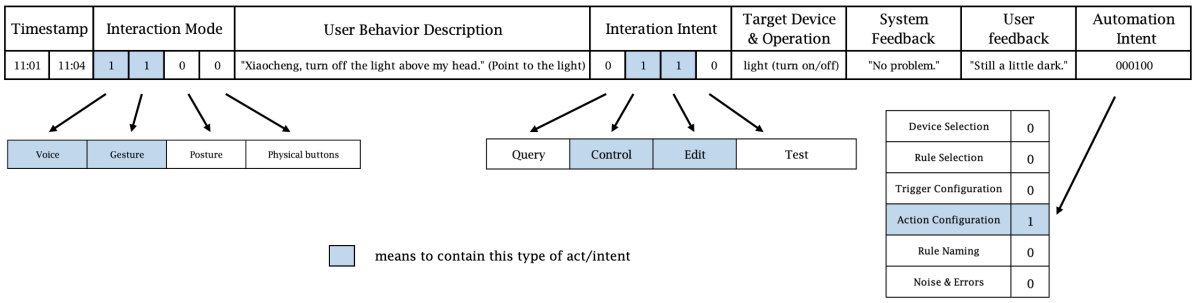

Fig. 7. Data Sample(Field from left to right: Timestamp, Interaction Mode, User Behavior Description, Interaction Intent, Target Device & Operation, System Feedback, User Feedback, Automation Intent)

Round 2: In the second round, the researchers top-down discussed the behavioral characteristics comprehensively based on the first round results. We followed a general user interaction model [70] in which we went through a hierarchical analysis to understand users' intentions, behavior, and feedback in three dimensions (task-level, rule-level, and content-level). We settled on a standard analysis norm of AU and rules. Then researchers exchanged the data to revise and further analyzed.

Round 3: In the third round, the researchers conducted an affinity diagramming approach. We grouped and labeled the topics according to our research questions and discussed the controversial notes. We literately updated the affinity wall until finally reached a consensus. This process fostered five high-level themes which we detailed and presented in the next session.

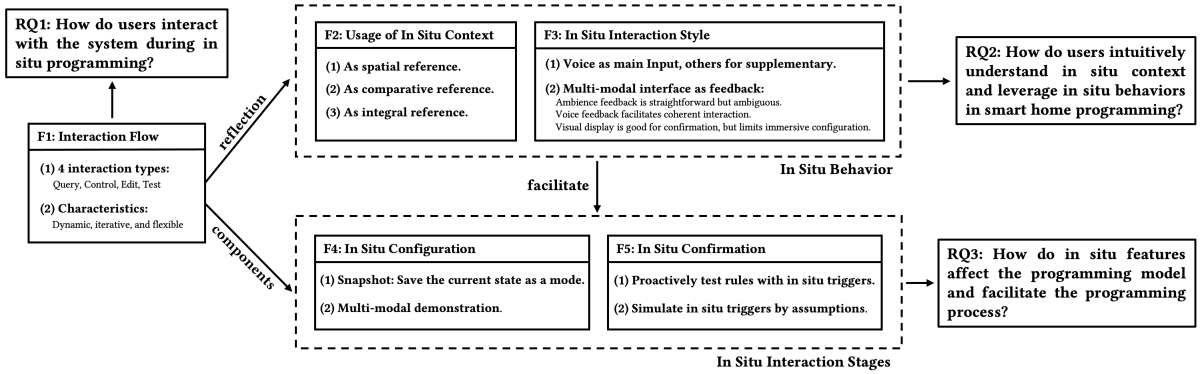

Fig. 8. A structural overview of our findings.

5 FINDINGS

From the results of the user study, we found that participants dynamically leverage the in-situ interaction and feedback to configure and confirm the rules. We unfold our findings into five main categories: 1) interaction flow, 2) usage of in-situ context, 3) in-situ interaction style, 4) in-situ configuration approaches, and 5) in-situ confirmation approaches. Fig.8 illustrates a structural overview of our findings and how these findings echo our three research questions.

5.1 Interaction Flow

We frame the interaction flow of in-situ programming (Fig.9), defined as a session of user-system interaction in one programming task. We identify two phases, that is, **configuration and confirmation**, which are dynamically realized by four main interaction nodes: query, control, edit, and test. Specifically, these nodes refer to a higher level of interaction intention, independent of the specific ways to perform it. These nodes are labeled in the dataset based on the following definitions.

- **Query:** Participants acquire information on the context, system capacity, rules, etc.
- **Control:** Participants conduct operations on devices.
- **Edit:** Participants edit the rules' contents, including names, triggers, and actions.
- **Test:** Participants activate the rules to check the correctness.

Generally, ISP starts with the configuration phase and ends with the confirmation phase. Query (n=101, 5.8%) includes directly asking the agent and checking the feedback of the interfaces. Edit (n=528, 30.3%) is the core part of rule configuration, including the rule-level editing (create, modify, delete) and content-level editing. Especially, control and test are two unique interaction nodes in ISP. Control (n=768, 44.1%) leads to directly changing the state of the environment by which participants can obtain in-situ ambiance responses. Test (n=345, 19.8%) is also intuitively conducted to confirm the rules. After testing that the rules are correct, participants completed the final confirmation and finished the whole programming, otherwise, they iteratively configure and confirm the rules.

The programming flow of ISP presents a significant **dynamic nature**, in which participants **iteratively** and **flexibly** undertake configuration and confirmation.

5.1.1 Iterative. s Generally, participants configure a rule through iterative refinement and confirmation based on the in-situ effects and feedback, with 83% of the rules (n=177) completed through iteration. Besides, instead of a batch process in traditional paradigms, participants adopted **incremental acts** in one task. For example, participants progressively supplement details of the rules, e.g., adding triggers, specific condition branches, and names.

5.1.2 Flexible. Flexibility is greatly reflected in the expression style and the applied approaches. Participants flexibly switch the intention nodes and comprehensively utilize in-situ interaction to construct different in-situ methods. Especially, the combination of control and edit is highlighted as a new in-situ configuration approach **"snapshot"**(Section 5.4). The combination of query and test is addressed as a new in-situ confirm approach **"simulation"** (Section 5.5).

Besides, the mental model of the trigger-action paradigm is still recognizable albeit fuzzy. Participants followed a more **flexible expression and rule structure based on their understanding of the scenarios and their needs.** Through interviews, all of them reported that they conceived the trigger first then the action (N=16). While in practice, 66% of the rules (n=140) are without names; 48% (n=102) are without triggers. 31% of rules (n=66) are configured by adding triggers or names after setting actions.

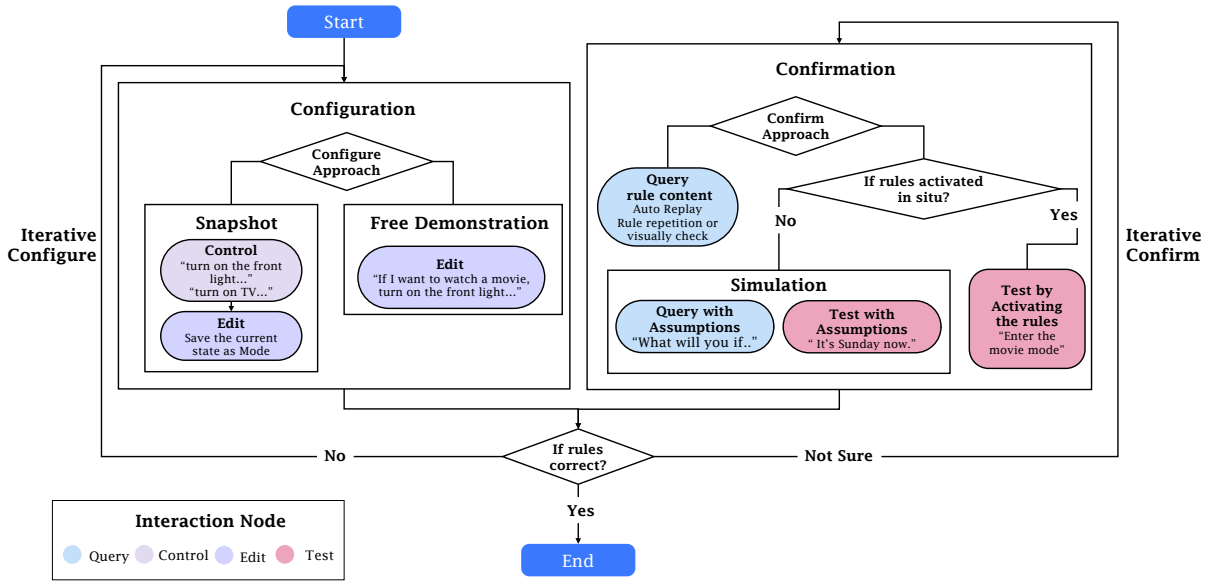

Fig. 9. Interaction flow of in-situ programming

5.2 Usage of In-Situ Context

We found that participants interpret in-situ context based on **spatial, comparative and integral references**, which **reduces the cognitive load and interaction load** in configuring spatial-related dynamic rules and batch operations.

5.2.1 Spatial Reference. The spatial references [120] are counted 85 times, 55% are expressed by explicit articulation (n=47), 45% are by pointing gestures and indicative pronouns (n=47), e.g., saying "open that light" and pointing to the specific device. The majority (n=74, 87%) are used to target the lights distributed in the environment, including absolute reference(n=8, 9.4%) like "the light front", relative reference (n=77, 90.6%) like "the light above the TV" or "the light above me". 23% of the spatial reference is used to describe the function with spatial-related parameters (n=21), e.g., "air conditioning blowing upward".

In addition, participants tend to **locate the devices based on their real-time location and orientation**, leading to rules with **dynamic in-situ parameters** (n=17, 8%). For example, "let the fan towards me"(P1, P14). "Wherever I go, turn on the lights above me" (P3, P6, P7, P16). "When I snap my fingers at something, please help me switch it on or off" (P6, P16). These dynamic rules are all set in their free configuration process.

From the interview, participants agreed that it is intuitive to establish a self-centered spatial relationship with the surroundings. P2, who is experienced in the IFTTT setting, reported that "it is easier to select devices and describe those human-related parameters that may be dynamic."

5.2.2 Comparative reference. We found that participants interpret the rules based on a comparative reference with the current situation. They explicitly query the in-situ context and have comparative expressions.

Query about the in-situ context (n=81) takes 60.3% of all the queries, e.g., the current temperature, time, and device states. Indicative pronouns to the current situation are counted 104 times. For example, "save current mode as daily mode" (P2, P8, P16), and "Please remind me to sleep at this time every day" (P12). Comparative references based on in-situ states of the context are counted 59 times, in which participants **express incrementally based on their current situation**. Comparative terms are used to set parameters related to devices or environment, e.g., "brighter", "warmer", etc., in which participants fine-tune by **fuzzy configuration instead of specifying the exact value**. Especially, once they achieve the expected effects, they do not seek to be informed of the exact value. From the interview, they reported the in-situ state of the environment is enough for them to confirm the outcomes. P5 stated that "The fuzzy expression of parameters could clearly convey my needs as I could perceive the straightforward effects in the field." Most inexperienced participants (N=8) reported it is useful to set parameters as it is hard to match their needs to an accurate value. However, P12 and P14, who are experienced in programming while inexperienced in IFTTT reported that they prefer an exact value to build confidence in the configuration.

5.2.3 Integral Reference. We found participants build an **integral understanding based on the in-situ context** rather than focusing on individual devices setting. Integral references are counted 87 times and mostly are used for batch control and configuration of devices. Generally, participants described the relationship between the part and the whole, e.g., "close all the lights except for the main light".

Group selection is also combined with spatial and quantity information, e.g., "turn off the two lights in front". As is explained in the interview, participants reported that the **spatially adjacent devices are naturally identified as a group**. Specifically, in-situ light setting is more integral. P8 said, "Basically, I tend to set a group of lights to the same parameters, then tune the parameters in groups to see the holistic effects." P5 also said "I care more about the whole light effects and it is more straightforward to make integrated adjustments rather than configure individual lights then make coordination." From the interview, a majority (N=13) agreed that it is more convenient for operations in a group as they can express themselves easily with more in-situ references.

Beside, participants pay more attention to the holistic scene (the situated room), and some made commands to control the scene. For example, P10 and P15 made similar commands like "close the current scene". From the interview, they explained the specific meaning of the commands that they expected to reverse or withdraw all state changes of the devices in the current scene.

5.3 In-Situ Interaction Style

Participants intuitively **follow a multi-modality and communicative interaction style**, which is consistent with the interaction patterns they have when interacting with the smart homes. Generally, they prefer **voice interfaces as dominant input and multi-model interfaces as feedback.**

5.3.1 Voice for main input, others for supplementary. Voice interface is used as dominant input while other interfaces like gestures and tangible interfaces are used as supplementary to support configuration accuracy. Voice interface is the most used modality (n=1369, 87.4%), respectively for query (n=90, 89.2%), control (n=569, 74%), edit (n=468, 88.8%) and test (n=242, 70.2%). Other interfaces are primarily used as supplementary. Gesture and posture are the second used modality (n=160, 10.2%), control (n=70, 43.8%), edit (n=9, 5.6%), and test (n=81, 50.6%). Pointing is greatly used to assist in targeting devices. Especially, **a gesture is used as a demonstration when there is a gesture-related trigger** in the rules. For example, P3 set a rule as "XiaoCheng, when I do this, turn on the air conditioner." She made an action demonstration of finger snapping at the same time. Besides, participants applied gestures to activate specific rules in the test. Noteworthy, participants showed great interest in the gesture-triggered rules both during the study and in the later interview. They explored more gesture-triggered rules in a free configuration process. **The tangible interface is only adopted as an alternative way when**

they do not how to express the specific function. It is rarely used (n=38, 2.4%) through the user study, for control (n=32, 84.2%), edit (n=1, 2.6%), and test (n=5, 13.2%).

From the interview, we found that **accuracy** and **cost** are two key factors affecting the choice of the input interface. Participants agreed on a strategy that they would **search for a shortcut to ensure the highest expression accuracy at the lowest expression cost**. A majority (N=15) reported they preferred the voice interface because it is more accurate and easy to express their needs. P5 explained that "I prefer to make fewer gestures if the system can understand me correctly. I prefer to stay lazy and leisurely at home." Actually, as we observed, most of our participants leaned on the sofa through the entire configuration after they get familiar with the system. Except for testing the rules triggered by human positions such as "someone come in", participants rarely moved around the room and use the distributed tangible interfaces.

5.3.2 Multi-modal Interfaces as feedback. With the three kinds of feedback forms we provide, participants reported that they benefit from all kinds of forms, especially a multi-model combination based on different scenarios in both configuration and confirmation processes.

Ambient feedback is straightforward but limited in visibility and ambiguity. Ambient feedback includes static reference and dynamic response when users control devices or rules. In the confirmation process, ambient responses provide straightforward feedback, which reduces the strong dependency on other external channels. 86% of rules (n=183) are confirmed only by testing the execution of the rules without seeking the specific content through voice or visual feedback. In the configuration process, ambient feedback is a key reference to build and fine-tune the rules.

However, ambient feedback has limitations in visibility. For one thing, the response is only visible when there is a **perceivable** change in the device states. For example, P7 transited from movie mode to daily mode. She was confused because these two modes were similar, and she could not see obvious changes. Another issue is ambiguity, as ambient feedback provides a vague reference to the accurate content of the rules. A few participants reported that they could not recall what they had set and felt insecure about the configuration (P1, P12). However, a majority (N=10) reported that it was not necessary for them to be acknowledged with accurate content.

Voice feedback facilitates coherent interaction. Voice feedback is a primary channel that intelligent agents communicate with users. In the configuration phase, voice feedback is preferred. Participants reported that it built coherent communication with the system, as they primarily used the voice interface as an input channel. All agreed that the voice feedback closely followed their commands and explicitly assisted them to ensure the validity of the input and clarifying the process of configuration. While in the confirmation phase, rule broadcasting is less used (n=40). P4 and P10 reported concerns that "it will be annoying and demanding when the rule is complex."

Visual display is a nice-to-have in confirmation but limits immersive configuration. Visual display was mostly used in the confirmation process, especially when participants needed to be informed of the specific contents of the rules. The display is explicitly checked (n=24), mostly when participants were confused about the system failure. From the interview, a majority (N=10) reported they did not need display-based feedback when configuring rules but welcome it in later confirmation or management. Some (N=6) reported the visual display helped them confirm complex rules. However, **visual display distracts participants' immersion in the environment in the configuration process**. P11 approved the point, saying "I do not look at the display when configuring, as it distracts my attention from the environment."

5.4 In-Situ Configuration Approach

We identified two user-enacted configuration approaches in which participants leverage in-situ interaction style and feedback. We outlined the approaches as "snapshot" and "free demonstration". Generally, participants individually or comprehensively applied these two approaches based on their understanding of the task.

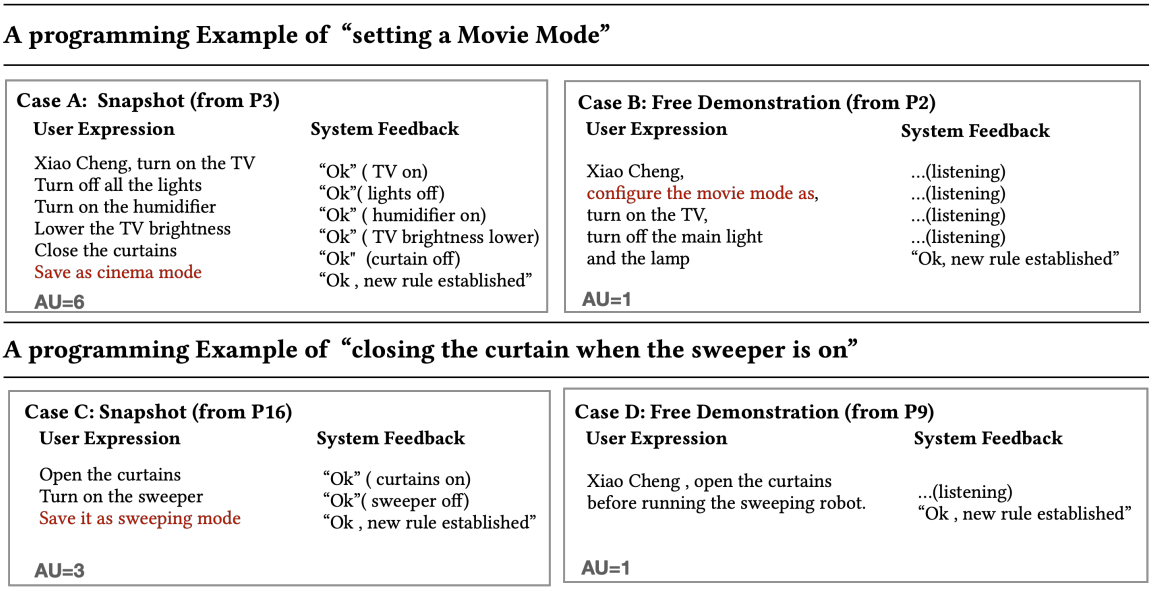

Fig. 10. Programming Examples of Snapshot and Free demonstration: AU means the number of action unit (AU) in a complete configuration process.

5.4.1 Snapshot: save the current state as a scene. "Snapshot" is a novel state-oriented programming-by-demonstration (PBD) paradigm [24], in which participants adjust the environment to their desired state by controlling devices. When the holistic environment state meets their requirements, they "save" the current scene or command that "next time [**trigger**], keep the scene this way". For modification, participants replaced the old states with the new states and resaved the scene. Compared with process-oriented PBD approaches, participants do not explicitly record the sequential behaviors/events [66] of devices but record a set of parallel activated states.

The snapshot patterns are found (n=71, 33%), mostly used in non-sequential tasks, e.g., setting a movie scene. Fig. 10 shows two example cases of snapshot expression. Case A saves a set of device states as the rule's action and case C saves the relationship of two devices in a rule. The system reacts to every user command and provides in-situ responses. When asked about the objects they preferred to record, participants reported they most cared about the devices which were activated in the scene or manipulated during configuration.

Snapshot is highly valued by the most (N=15) from the interview. They reported that the approach provides a straightforward preview of the rules. P4 reported, "I wasn't quite sure about the exact effects until I actually experienced it because I couldn't set the precise parameters by imagination." Besides, snapshot simplifies configuring rules with multiple devices by naturally splitting a long description of multiple devices into step-by-step control commands. However, some are also concerned that snapshot is limited in expressing complex rules with multiple conditions branches or sequential requirements.

5.4.2 Multi-model Free Demonstration. Free Demonstration is an approach that participants configure by multi-modality demonstration, that is, **language specification with acting as supplementary**. Compared with traditional voice-based EUP, the rule specification is more flexible, improvisational, and incremental based on the

traditional trigger-action form. Basically, participants demonstrated in sequential order, successively editing the trigger and then the action. The most used scripts include "when *Trigger* occurs, *Action*", "if *Trigger/Condition*, then *Action*", or sequential description like "first step, then...". Also, it is used to modify rules, in which participants directly described the contents of the revision. For example "turn on the main light in TV Mode", "set the air conditioner to 25 in Get up Mode". As is shown in Fig.10, the system does not react until the end of the users' complete specification.

Acting with demonstration is primarily found in complex rules with dynamic human-related contents like position, and gesture. For example, P4 set a rule in which she wanted the environment responded to the changes in human position (Script T5). The rule is sequential and has three condition branches. To set the rule, P4 acted on the position change while specifying the rule contents. In the later interview, she said this acting is helpful for her to conceive such complex rules.

67% of the rules (n=142) are configured by free demonstration and participants also applied the approach as an important complement to "snapshot" to modify or add the settings. Through the interview, participants reported that the free demonstration increased expressiveness in rules with dynamic and human-related parameters. However, the expression cost increases when the complexity of rules rises. For example, although P8 approved of good experience in using free demonstration, he reported concerns.

> I was under a lot of pressure when specifying the rules and it took me a long time to think. I was concerned about how to make the system understand me. I would be in a hurry to express myself because I was afraid the system will interrupt me early. Sometimes, I would say something wrong or pause to think and be afraid that the system gets it wrong and leads to trouble.

5.5 In-Situ Confirmation Approach

We found participants intuitively **test the rules in the confirmation phase, greatly in place of seeking feedback on rule contents**. Participants enacted two kinds of test methods based on if the rule can be in-situ activated.

5.5.1 Proactively activate rules with in-situ triggers. For the rules which can be in-situ activated, participants tended to proactively activate the rules to test. In this phase, they expected to preview the whole execution of the rules and observe the obvious state transition of the environment.

Generally, they conducted different test practices after "snapshot" and "free demonstration". Tests more naturally followed the "free demonstration" as participants could not obtain the in-situ responses through configuration. It is expected as a type of rule **auto replay** (P6, P16), that is, participants asked for a test, the system automatically executed the rule actions and demonstrated the skipped parts (usually triggers and time delays, etc.). With regard to "snapshot", participants preferred a **reset-and-replay** method. Because when completing "snapshot", they are in the exact scene they just set. To test the rule, they tended to change the current state first, then activate the rule. For example, they would reset the environment by commanding "close all the devices" or "restore to the state before configuration".

5.5.2 Simulate In-Situ Triggers by Assumptions. For triggers that can't be rapidly in-situ activated (e.g., specific time or temperature, long time delay, multiple family members), participants used assumptions to simulate the triggers. For example, to test a rule with the trigger as "9 pm on Friday", participants said "XiaoCheng, it's nine o'clock on Friday." To rapidly simulate time delays such as "10 minutes later, turn on the lights", participants said "XiaoCheng, 10 minutes are up." However, some (P10, P12) were confused about the priority of the simulated trigger and the actual states. P10 said although she tested successfully with the simulated trigger, she was still worried when the simulated trigger conflicted with the current situation.

Another simulation method is an **interrogative simulation**, in which participants query how the system would react if a certain trigger occurs. For example, "XiaoCheng, what will you do ten minutes later?"

Level of configuration confidence and test costs are two main factors that affect participants' test preferences. All said that they would prefer to test all the rules to make sure that the rule is set correctly and will execute properly, except for the very simple rules. P2 and P12 mentioned the test cost. P2 said if the test process requires too many operations or interrupts the current activities, he would not test. A majority (N=13) reported they preferred a rapid test and considered time as a critical cost.

6 DESIGN SPACES OF IN-SITU PROGRAMMING PARADIGM

By examining users' intuitive behavior models and mental models, we highlighted three design spaces for in-situ programming (ISP) systems: dynamic programming, in-situ interface, and state and scene management. In this section, we discussed how these design spaces echo, extend, and distinguish from the design implications of the state-of-art in situ solutions. Further, we propose design recommendations under a trade-off between technical feasibility and our design goals with regard to the programming mechanism, interface, and automation system.

6.1 In-Situ Dynamic Programming

The most dramatic finding of the ISP paradigm is the dynamic programming style. Compared with traditional TAP paradigms, ISP is more flexible and malleable strongly depending on the in-situ context and tasks, reflected in more fine-grained aspects, eg., flexible intention switch, dynamic rules with contextual parameters, ambiguous and flexible expression style [32, 116, 125] and mental models (e.g., fuzzy parameters [76], flexible expression structure [32, 57], free demonstration), multi-model interaction patterns, and mixed approaches to configure and confirm the rules. It is indicated that the dynamic characteristic primarily derives from users' natural behavior patterns in intelligent ambiance and conversational agent [76] with positive expectations [18]. The ambiguity also increases, since people tend to omit information that can be easily inferred in the situated context [63]. The significance, challenges, and mechanisms to support dynamic programming are widely explored in domains like living programming, and voiced-based EUP. Notably, in ISP scenarios, it is more challenging because the highlighted flexibility and ambiguity are closed related to the dynamic time-spatial context (e.g., "whenever I go, turn on the lights above me."). In our ISP prototype, dynamic programming is well supported via the wizard techniques and encouragingly effectively enhances intuitive interaction and adaptability in different tasks, especially complex ones. For one thing, the **in-situ context and multi-model input channel provide reference and confidence for fuzzy expression so that users can express themselves more easily**. For example, users are allowed fuzzy parameters without knowing exact values, handle the distributed devices more easily with spatial reference, and express human-related parameters with multi-model information (position, gesture acting) as a programming language. For another, the voice-based and ambiance-based paradigms set fewer time-spatial limits when configuring, and testing.

1) Plastic Programming Mechanism. The programming mechanism should be plastic and scalable, easy to be attached, modified, and dynamically adapt to users' incremental operations [119]. The structure of automation should be a more flexible extension based on TAP and allow for void and dynamic parameters (e.g., void rule names or triggers, current time, real-time user position) and flexible configuration order. The default names can be extracted from the rules triggers [49] and the default triggers can be filled as "triggering via smart speakers" [94]. The system should support users' dynamic commands such as, such as rolling back programming flow (eg., "withdraw the previous commands", "recover the scene"), and trial-and-error commands (eg., rephrase, correct, supplement), agile iteration based on programming flow [76] (eg., appending, replacing, modification after test).

2) Reduce the ambiguity of programming input. Given the ambiguous inputs resulting from the omitted [63], iterative, and spatial expression, the system should improve dynamic time-spatial context awareness to

infer the in-situ parameters and interaction intents. Literature [12, 39, 76] proposed ambiguity strategies to infer omitted or high-semantic parameters and commands (eg., room, time, value) based on the multi-turn dialogue, contextual information, and user-predefined profiles. Furthermore, our findings indicated that users' situated location, orientation, and spatial keywords [77, 120] are important. The ambiguity strategies should also avoid additional conversation turns [76]. The strategy conducted in our ISP prototype can be drawn on. Specifically, the system only proposes a follow-up question [12] if the ambiguity can not be addressed successively after system inference and user correction [35].

6.2 In-Situ Interface with Coherency and Clarity

While we provided multi-model interfaces during the study, the findings revealed that **the synthesis of the ambient interface and voice interface would be sufficient and workable for the design goals of in-the-fly and in-the-space.** The limited use of other interfaces (gesture, tangible button, visual display) suggests that users intuitively minimize interaction load to convey their intentions. It is surprising that the tangible-based programming style seems less intuitive as the previous works claimed [36, 51] due to high physical load. There seems to be behavioral inertia since users preferred not to conflict with their activity coherency. Moreover, an intriguing gap is that, despite the flexibility of behavior patterns and ambiguity of mental models (discussed in Section 5.1), users are highly focused on whether they are accurately expressing their intent and whether the rules are aligned with that intent. One concern is that users generally capture a vague understanding of the context, so it is hard to build an accurate mental model of the rules, like accurate parameters, distinguishing action states or events [66], etc. The vague understanding will produce confusion and insecurity about what they had created. In the ISP paradigm, the gap is greatly ameliorated by incorporating straightforward and real-time [46] in-situ ambient feedback to reduce users' demands for accuracy. For one thing, ambient feedback simplifies the configuration by helping users build up the rules grounding the actual context, reducing configure costs by straightforward reference. For another, it facilitates rapid conformation, in which users can preview how the rules execute and the final effects through real-time responses. This is evident in the use case of snapshot and simulation, in which users gain a clear understanding of the programming process and outcomes. Taken together, we highlighted the design space as **clarity instead of accuracy** in ISP design. Below in this section, we propose multi-modality interface design recommendations for building coherency and clarity.

1) Facilitate coherent user thinking and expression. The system should react consistently to users' preferences under different interaction intentions. For example, the voice input "turn on the TV" can be a control command or a part of rule-setting. In a free demonstration, users prefer coherently expressing their configure needs without device responses, then previewing the holistic effects by ambiance response. Usually, there may be a long speech input. This is different from the proposals of previous works, in which they suggested a fragmented [79] or step-by-step [46, 54] multi-turn interaction. Thus, the dialogue system should avoid interrupting the user at this time and can ask users whether to end the input during long pauses. While in "snapshot", users prefer to get in-situ responses as they tend to step by step set and tune individual devices. Hence the dialogue system should react to the control command and inform the rule creation after "save". The difference requires different waiting times for the two approaches.

2) Clarify key information, transition, and process. Through the findings, we identified the limitations of ambient feedback. One important aspect is, **ambient feedback greatly relies on users' perceptions**. Sometimes it can be hard to perceive when the state transition is not obvious. Another issue is its limitation to exhibit complex logic in the rules like time delay, conditions branches, etc. **Ambient response with voice demonstration** is an applicable solution. In the configuration process, the voice agent should give timely feedback on the key configuration points and improve the transparency [76] of the configuration process, including the creation, modification, testing, and the reason for failure configuration. In the confirmation phase, the voice agent should

concisely clarify the key logic or simulated triggers of the rules, rather than simply exhibiting static and lengthy rule contents and increasing cognitive load [115]. Besides, voice is necessary to inform the imperceivable ambient responses. A vivid example is, some users require a rule test after "snapshot". Whereas, as they are already in the specific mode, the system appears not to be responding even though the rules are executed. Hence, this time, the agent should prompt the user that the rule is already triggered and inform the user to change the scene for testing.

6.3 In-Situ Approaches with State and Scene Management

Confirming the previous works, the mental model of the state/event of the individual trigger and action is ambiguous for users and affects the accuracy of configuration and debugging [26, 66, 104, 129]. Our findings also reinforce the point that users are facing the scene [82] during situated programming and a set of state actions [26] simplifies programming among distributed devices, indicating the importance of STATES in IoT-related automation. For one thing, users intuitively interpret the action of TAP rules with the states of holistic context, distinguished from the traditional paradigms which individually manage the events or behaviors of the context [16, 50] or process-oriented paradigms [24, 29]. For another, in-situ approaches take the holistic scene both as the interactive programming environment [126] and execution environment [81], which requires balancing the isolation and resource distribution among parallel tasks and programs. Hence, the context states should be recorded, maintained, and blocked/activated dynamically based on the programming process to support in-situ approaches.

1) Scene management for "snapshot". The key to "snapshot" is to exhibit the scene to the users consistent with their mental model as well as reduce the cognitive load from unrelated states. **First, the system should record all the device states in the scene (the situated room), rather than the operations to achieve the scene.** Given that some of the recorded states are context-dependent (e.g, the light from the next room may affect the current light adjustment), the system should also record important implicit environment states (temperature, brightness) as a reference for further scene iteration. Second, the snapshot rule displayed to users through voice or visual feedback should only contain related states, including the activated devices and the manipulated devices, the on/off status, and specific setting values.

2) State priority management for "simulation". In ISP, the situated environment undertakes automation editing, testing, and execution, so programming tasks should be prioritized and free of interference. The system should ensure the programming process will not trigger other rules by mistake or be interrupted the rules under execution. Especially, in "simulation", users usually test the automation rules by specifying the simulated trigger. State priority management should create a virtual state in the system, with higher priority than the real state. The start of testing mode and the priority strategies should be explicitly informed to users in case of confusion.

7 DISCUSSION, LIMITATIONS AND FUTURE WORKS

7.1 Issues for Real-World Deployment

In this work, we investigated the potentials of in-situ programming (ISP) under an ideal Wizard-Of-Oz setting mainly to eliminate the effect of system implementation on users' behavior. Here we further discuss how such a system to sense human behavior and give intelligent responses can be implemented, as well as the technical limitations and further research directions. For the sensing system, sensing information frequently used for interaction includes 1) positions and orientations of humans, devices, and objects [111]; 2) voice commands [108]; 3) body and hand gestures [85–87]. Although the latest signal processing (e.g., indoor positioning [28, 111], 6DoF tracking [34], and speech recognition [108]) and computer vision (e.g., pose tracking [22] and facial landmark tracking [73]) techniques provided potential sensing solutions to in-situ programming, their performances are to be evaluated and optimized. Further research on sensor form and tracking algorithms for in-situ programming

should be conducted. For example, heterogeneous home layout and device deployment bring the challenge to the tracking and unified representation of human behaviors, where the system should determine the optimal gestural input channels for reliable performance. The system should also allow certain degrees of customization for different needs among different user groups (e.g., privacy preservation).

The inference system is expected to resolve user input into system commands and give intelligent responses to the user. The recent development of large language models (LLMs, e.g., GPT-3 [30]) and corresponding AI-empowered techniques (e.g., text-to-SQL [127], text generation [30] and dialog [8]) could be potential solutions to implement a smart and high-fidelity voice agent. For example, when the user says "when I do this gesture (hand waving upward) and the TV is on, turn up the voice", the inference system first resolves it into IFTTT commands, checking the validity and giving feedback to the user. Another critical question is how the system deals with errors in tracking and ambiguity in semantics. As validated in our study, confirmation is an essential mechanism for ensuring the validity of user-defined rules. Moreover, voice agents with multi-round and context-preserving dialog capabilities could drive a more intelligent interaction process. For example, the user could interrupt the agent and append new descriptions to improve semantic clarity while the agent could actively ask for clarification regarding certain dialog sessions.

7.2 Threats to Validity

This section discussed potential threats to the research validity, and how we alleviated the threats.

7.2.1 Internal Threats. The internal validity in this research refers to the extent to which natural behaviors can be confidently observed and analyzed.

1) Experimental Effects: Since we explored users' behavior through user-enactment techniques in a mocked-up environment, the behavior of participants may be affected by the experiment, slightly different from that in their homes. Besides, to cover the task features we focused on, we provided users mostly with preset scenarios. The process of users' understanding of the scene may affect their configuration behavior. User behavior in open tasks will be more natural, so realistic smart home programming behaviors need to be explored in the future.

We minimize the experimental threat by providing sufficient time for users to get familiar with the environment and the enactment scenario. Especially, the scenario is broadcast by the moderator off-site in order to make participants imagine the scenarios instead of reading the scenario from the documents. After the study, users rated if they behaved naturally during the study from 1-5. The result is an average of 4.8, with the lowest being 4 and the highest being 5.

2) Observation Limitation: The sensing part of the Wizard-of-Oz system is conducted by researcher observation and the behavior is analyzed by researchers through videos. Due to technical limitations, we only focused on direct interaction, such as gestures and voice. Other subtle but meaningful interactive inputs, such as gazing and facial expressions are not the focus of the analysis. Future works can consider these inputs for a better understanding of users' intents and produce a richer set of interaction data.

7.2.2 External Threats. The external threats refer to which extent our findings can the generalized to a broader context.

1) Participant Bias: Participant bias discussed how the selection of participants affects the generalizability of the behavior model and the usability of the ISP paradigm. From a demographic perspective, most of the participants are colleague students around 24 years old. While the population is fairly heterogeneous in age, the young population (around 18–35 years) and highly-educated students are representatives of potential users of smart home technology [21, 117]. From a residential perspective, our participants are diverse in living arrangements, including living with co-occupants (N=9) and living alone (N=7). We do not capture evident differences in behavior from residential differences, but we collected rules involving co-occupants and non-occupants, indicating future

research's attention on the social needs automation [112]. From the users' knowledge perspective, our participants vary in different technical backgrounds (8 with technical background and 8 with non-technical background), and the majority (N=12) are inexperienced in IFTTT settings. Although we unified the system capacity introduction and the system behaviors through the studies, participants reported that they had different expectations for the system depending on different technical backgrounds and their previous experiences [93]. For example, a few naive users (N=2) could not think of advanced system functions like the test. Research on richer user types will improve the generalizability of the results such as IFTTT advanced users.

2) Scenario Limitations: As we focus on the in-situ behaviors, the study is arranged in a living room and with a single participant. The limited scenarios may not cover situations of multi-user activities [58] and cross-scenario tasks. We addressed the threats by allowing free configuration based on participants' real-life situations. Within the user-customized scenarios involving multiple users (n=10) and ex-situ scenarios (n=16), we identified similar behavior patterns to that in the preset scenarios.

7.3 Future Works

We are optimistic that ISP opens up a new direction to explore IoT end-user programming paradigm, which is ubiquitous, immersive, and seamless. Our research process, dataset, and design implications are helpful for future research to refine the paradigm. Future works may explore capacities in other programming phases (e.g., debugging, testing, and management) and real-life feasibility from technical and social perspectives. We encourage other in-situ solutions to validate and apply our findings in different scenarios. Extensions of user behavior dataset can serve as valuable materials for future smart home interaction and EUP systems in user intent recognition, interaction approaches, and performance improvement.

8 CONCLUSION

As AI-based IoT technologies foster a new prospect of future smart homes, this study stepped forward to understanding how to construct ubiquitous and seamless programming paradigms in AIoT. We proposed in-situ programming (ISP) as a novel programming paradigm in the IoT environment. By exploring users' intuitive interaction patterns, we outlined the programming flow, configuration and confirmation approaches, and users' behavior features of leveraging in-situ interaction and context. We discussed how in-situ features facilitate a natural and accessible programming experience and concluded with design recommendations for ISP design. We believe that in-situ programming (ISP) has excellent potential to improve the smart home automation programming experience.

ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China under Grant No. 62132010, and 2025 Key Technological Innovation Program of Ningbo City under Grant No.2022Z080, and also by Beijing Key Lab of Networked Multimedia, and Institute for Artificial Intelligence, Tsinghua University (THUAI).

REFERENCES

[1] 2022. Amazon Alexa. https://developer.amazon.com/alexa
[2] 2022. Google. https://assistant.google.com/
[3] 2022. HomeAssistant. https://www.home-assistant.io/
[4] 2022. Homekit. https://apple.com/ios/home/
[5] 2022. IFTTT. https://ifttt.com
[6] 2022. Zapier. https://zapier.com/
[7] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. 1999. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*. Springer, 304–307.

[8] Eleni Adamopoulou and Lefteris Moussiades. 2020. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 373–383.

[9] Ademar Aguiar, André Restivo, Filipe Figueiredo Correia, Hugo Sereno Ferreira, and João Pedro Dias. 2019. Live software development: Tightening the feedback loops. In *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*. 1–6.

[10] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. 2021. A survey on end-user robot programming. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36.

[11] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. 2012. A review of smart homes—Past, present, and future. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)* 42, 6 (2012), 1190–1203.

[12] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–13.

[13] Tawfiq Ammari, Jofish Kaye, Janice Y Tsai, and Frank Bentley. 2019. Music, Search, and IoT: How People (Really) Use Voice Assistants. *ACM Trans. Comput. Hum. Interact.* 26, 3 (2019), 17–1.

[14] Leonardo Angelini, Elena Mugellini, Omar Abou Khaled, and Nadine Couture. 2018. Internet of Tangible Things (IoTT): Challenges and opportunities for tangible interaction with IoT. In *Informatics*, Vol. 5. MDPI, 7.

[15] Carmelo Ardito, Giuseppe Desolda, Rosa Lanzilotti, Alessio Malizia, Maristella Matera, Paolo Buono, and Antonio Piccinno. 2020. User-defined semantics for the design of IoT systems enabling smart interactive experiences. *Personal and Ubiquitous Computing* 24, 6 (2020), 781–796.

[16] Raffaele Ariano, Marco Manca, Fabio Paternò, and Carmen Santoro. 2022. Smartphone-based augmented reality for end-user creation of home automations. *Behaviour & Information Technology* (2022), 1–17.

[17] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*. 121–130.

[18] Barbara Rita Barricelli, Elena Casiraghi, and Stefano Valtolina. 2019. Virtual assistants for end-user development in the Internet of Things. In *International Symposium on End User Development*. Springer, 209–216.

[19] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. 2019. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software* 149 (2019), 101–137.

[20] Barbara Rita Barricelli and Stefano Valtolina. 2015. Designing for end-user development in the internet of things. In *International symposium on end user development*. Springer, 9–24.

[21] Patricia Baudier, Chantal Ammi, and Matthieu Deboeuf-Rouchon. 2020. Smart home: Highly-educated students' acceptance. *Technological Forecasting and Social Change* 153 (2020), 119355.

[22] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. 2020. BlazePose: On-device Real-time Body Pose tracking. https://doi.org/10.48550/ARXIV.2006.10204

[23] Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle Lottridge. 2018. Understanding the long-term use of smart speaker assistants. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–24.

[24] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. Robot programming by demonstration. In *Springer handbook of robotics*. Springer, 1371–1394.

[25] Alexandre Bissoli, Daniel Lavino-Junior, Mariana Sime, Lucas Encarnação, and Teodiano Bastos-Filho. 2019. A human–machine interface based on eye tracking for controlling and monitoring a smart home using the internet of things. *Sensors* 19, 4 (2019), 859.

[26] Will Brackenbury, Abhimanyu Deora, Jillian Ritchey, Jason Vallee, Weijia He, Guan Wang, Michael L Littman, and Blase Ur. 2019. How users interpret bugs in trigger-action programming. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–12.

[27] Dave Braines, Nick O'leary, Anna Thomas, Daniel Harborne, Alun David Preece, and William M Webberley. 2017. Conversational homes: a uniform natural language approach for collaboration among humans and devices. *International Journal on Advances in Intelligent Systems* 10, 3/4 (2017), 223–237.

[28] Ramon F Brena, Juan Pablo García-Vázquez, Carlos E Galván-Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, and James Fangmeyer. 2017. Evolution of indoor positioning technologies: A survey. *Journal of Sensors* 2017 (2017).

[29] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring end user programming needs in home automation. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2 (2017), 1–35.

[30] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[31] AJ Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. 2011. Home automation in the wild: challenges and opportunities. In *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2115–2124.

[32] 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 (2018), 295–313.
[33] Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*. 341–350.
[34] Ke-Yu Chen, Shwetak N Patel, and Sean Keller. 2016. Finexus: Tracking precise motions of multiple fingertips using magnetic sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1504–1514.
[35] Yi-Shyuan Chiang, Ruei-Che Chang, Yi-Lin Chuang, Shih-Ya Chou, Hao-Ping Lee, I-Ju Lin, Jian-Hua Jiang Chen, and Yung-Ju Chang. 2020. Exploring the design space of user-system communication for smart-home routine assistants. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.
[36] Jeannette Shiaw-Yuan Chin, Victor Callaghan, and Graham Clarke. 2006. An End-User Programming Paradigm for Pervasive Computing Applications.. In *ICPS*, Vol. 6. 325–328.
[37] Meghan Clark, Prabal Dutta, and Mark W Newman. 2016. Towards a natural language programming interface for smart homes. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. 49–52.
[38] Miguel Angel Contreras-Castañeda, Juan Antonio Holgado-Terriza, Gonzalo Pomboza-Junez, Patricia Paderewski-Rodríguez, and Francisco Luis Gutiérrez-Vela. 2019. Smart home: Multimodal interaction for control of home devices. In *Proceedings of the XX International Conference on Human Computer Interaction*. 1–8.
[39] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. A high-level semantic approach to end-user development in the Internet of Things. *International Journal of Human-Computer Studies* 125 (2019), 41–54.
[40] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2020. HeyTAP: Bridging the Gaps Between Users' Needs and Technology in IF-THEN Rules via Conversation. In *Proceedings of the International Conference on Advanced Visual Interfaces*. 1–9.
[41] Miguel Coronado and Carlos A Iglesias. 2015. Task automation services: automation for the masses. *IEEE Internet Computing* 20, 1 (2015), 52–58.
[42] Joelle Coutaz, Alexandre Demeure, Sybille Caffiau, and James L Crowley. 2014. Early lessons from the development of SPOK, an end-user development environment for smart homes. In *Proceedings of the 2014 acm international joint conference on pervasive and ubiquitous computing: Adjunct publication*. 895–902.
[43] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. 1993. Wizard of Oz studies—why and how. *Knowledge-based systems* 6, 4 (1993), 258–266.
[44] Scott Davidoff, Min Kyung Lee, Charles Yiu, John Zimmerman, and Anind K Dey. 2006. Principles of smart home control. In *International conference on ubiquitous computing*. Springer, 19–34.
[45] Advisors Luigi De Russis, Alberto Monge Roffarello, and Cookie Policy-Login. [n. d.]. Conversational Agents for Creating Personalization Rules in the IoT. ([n. d.]).
[46] Luigi De Russis and Fulvio Corno. 2015. Homerules: A tangible end-user programming interface for smart homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 2109–2114.
[47] Luigi De Russis and Alberto Monge Roffarello. 2020. Personalizing IoT Ecosystems via Voice.. In *EMPATHY@ AVI*. 37–40.
[48] Luigi De Russis, Alberto Monge Roffarello, and Carlo Borsarelli. 2021. Towards Vocally-Composed Personalization Rules in the IoT.. In *EMPATHY@ INTERACT*. 1–5.
[49] Alexandre Demeure, Sybille Caffiau, Elena Elias, and Camille Roux. 2015. Building and using home automation systems: a field study. In *International Symposium on End User Development*. Springer, 125–140.
[50] 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 (TOCHI)* 24, 2 (2017), 1–52.
[51] Anind K Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 33–40.
[52] W Keith Edwards and Rebecca E Grinter. 2001. At home with ubiquitous computing: Seven challenges. In *International conference on ubiquitous computing*. Springer, 256–272.
[53] Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O Riedl. 2019. Automated rationale generation: a technique for explainable AI and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 263–274.
[54] Kasra Ferdowsifard, Allen Ordookhanians, Hila Peleg, Sorin Lerner, and Nadia Polikarpova. 2020. Small-step live programming by example. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 614–626.
[55] Maxwell Forbes, Rajesh PN Rao, Luke Zettlemoyer, and Maya Cakmak. 2015. Robot programming by demonstration with situated spatial language understanding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014–2020.
[56] 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.

[57] Manuel García-Herranz del Olmo, Pablo A Haya, and Xavier Alamán. 2010. Towards a ubiquitous end-user programming system for smart spaces. *Journal of Universal Computer Science* (2010).
[58] Christine Geeng and Franziska Roesner. 2019. Who's in control? Interactions in multi-user smart homes. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.
[59] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2 (2017), 1–33.
[60] Pomboza-Junez Gonzalo and A Holgado-Terriza Juan. 2015. Control of home devices based on hand gestures. In *2015 IEEE 5th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 510–514.
[61] Michal Gordon and David Harel. 2014. Steps towards scenario-based programming with a natural language interface. In *From Programs to Systems. The Systems perspective in Computing*. Springer, 129–144.
[62] Kirsten Gram-Hanssen and Sarah J Darby. 2018. "Home is where the smart is"? Evaluating smart home research and approaches against the concept of home. *Energy Research & Social Science* 37 (2018), 94–101.
[63] Herbert P Grice. 1975. Logic and conversation. In *Speech acts*. Brill, 41–58.
[64] Simone Hämmerle, Matthias Wimmer, Bernd Radig, and Michael Beetz. 2005. Sensor-based situated, individualized, and personalized interaction in smart environments. *Informatk 2005. Informatik Live! Band 1* (2005).
[65] Peizhao Hu, Jadwiga Indulska, and Ricky Robinson. 2008. An autonomic context management system for pervasive computing. In *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 213–223.
[66] 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*. 215–225.
[67] 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*. 1555–1562.
[68] 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. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–28.
[69] Rikke Hagensby Jensen, Yolande Strengers, Jesper Kjeldskov, Larissa Nicholls, and Mikael B Skov. 2018. Designing the desirable smart home: A study of household experiences and energy consumption impacts. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–14.
[70] Brigitte Jordan and Austin Henderson. 1995. Interaction analysis: Foundations and practice. *The journal of the learning sciences* 4, 1 (1995), 39–103.
[71] Runchang Kang, Anhong Guo, Gierad Laput, Yang Li, and Xiang'Anthony' Chen. 2019. Minuet: multimodal interaction with an internet of things. In *Symposium on spatial user interaction*. 1–10.
[72] Michal Kapinus, Vítězslav Beran, Zdeněk Materna, and Daniel Bambušek. 2019. Spatially situated end-user robot programming in augmented reality. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1–8.
[73] Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. 2019. Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs. https://doi.org/10.48550/ARXIV.1907.06724
[74] Fahim Kawsar, Gerd Kortuem, and Bashar Altakrouri. 2010. Supporting interaction with the internet of things across objects, time and space. In *2010 internet of things (IOT)*. IEEE, 1–8.
[75] Shahedul Huq Khandkar. 2009. Open coding. *University of Calgary* 23 (2009), 2009.
[76] Sanghoon Kim and In-Young Ko. 2022. A Conversational Approach for Modifying Service Mashups in IoT Environments. In *CHI Conference on Human Factors in Computing Systems*. 1–16.
[77] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 259–266.
[78] Christine Kühnel, Tilo Westermann, Fabian Hemmert, Sven Kratz, Alexander Müller, and Sebastian Möller. 2011. I'm home: Defining and evaluating a gesture set for smart-home control. *International Journal of Human-Computer Studies* 69, 11 (2011), 693–704.
[79] André Sousa Lago, João Pedro Dias, and Hugo Sereno Ferreira. 2021. Managing non-trivial internet-of-things systems with conversational assistants: A prototype and a feasibility experiment. *Journal of Computational Science* 51 (2021), 101324.
[80] Dounia Lahoual and Myriam Frejus. 2019. When users assist the voice assistants: From supervision to failure resolution. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–8.
[81] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–16.
[82] Jisoo Lee, Luis Garduño, Erin Walker, and Winslow Burleson. 2013. A tangible programming tool for creation of context-aware applications. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 391–400.
[83] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M Mitchell, and Brad A Myers. 2018. Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 105–114.

[84] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. 2017. Programming IoT devices by demonstration using mobile apps. In *International Symposium on End User Development*. Springer, 3–17.
[85] Chen Liang, Chi Hsia, Chun Yu, Yukang Yan, Yuntao Wang, and Yuanchun Shi. 2023. DRG-Keyboard: Enabling Subtle Gesture Typing on the Fingertip with Dual IMU Rings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 170 (jan 2023), 30 pages. https://doi.org/10.1145/3569463
[86] Chen Liang, Chun Yu, Yue Qin, Yuntao Wang, and Yuanchun Shi. 2021. DualRing: Enabling Subtle and Expressive Hand Interaction with Dual IMU Rings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 115 (sep 2021), 27 pages. https://doi.org/10.1145/3478114
[87] Chen Liang, Chun Yu, Xiaoying Wei, Xuhai Xu, Yongquan Hu, Yuntao Wang, and Yuanchun Shi. 2021. Auth+Track: Enabling Authentication Free Interaction on Smartphone by Continuous User Tracking. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 2, 16 pages. https://doi.org/10.1145/3411764.3445624
[88] Chung-Ju Liao, Shun-Feng Su, and Ming-Chang Chen. 2015. Vision-based hand gesture recognition system for a dynamic and complicated environment. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2891–2895.
[89] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-user development: An emerging paradigm. In *End user development*. Springer, 1–8.
[90] Evgenia Litvinova and Petri Vuorimaa. 2012. Engaging end users in real smart space programming. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 1090–1095.
[91] Liwei Liu, Wei Chen, Lu Liu, Kangkang Zhang, Jun Wei, and Yan Yang. 2021. TAGen: Generating Trigger-Action Rules for Smart Homes by Mining Event Traces. In *International Conference on Service-Oriented Computing*. Springer, 652–662.
[92] Andrés Lucero. 2015. Using affinity diagrams to evaluate interactive prototypes. In *IFIP conference on human-computer interaction*. Springer, 231–248.
[93] Ewa Luger and Abigail Sellen. 2016. " Like Having a Really Bad PA" The Gulf between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 5286–5297.
[94] Marco Manca, Parvaneh Parvin, Fabio Paternò, and Carmen Santoro. 2020. Integrating Alexa in a Rule-based Personalization Platform. In *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*. 108–113.
[95] Ilaria Mariani, Livio Tommaso, and Umberto Tolino. 2019. Drawing Interfaces: When Interaction Becomes Situated and Variable. In *DeSForM 2019. Beyond Intelligence*. Northumbria University, 114–121.
[96] Davit Marikyan, Savvas Papagiannidis, and Eleftherios Alamanos. 2019. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change* 138 (2019), 139–154.
[97] Zhaozong Meng and Joan Lu. 2015. A rule-based service customization strategy for smart home context-aware automation. *IEEE Transactions on Mobile Computing* 15, 3 (2015), 558–571.
[98] 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 *International conference on pervasive computing*. Springer, 143–160.
[99] Progress Mtshali and Freedom Khubisa. 2019. A smart home appliance control system for physically disabled people. In *2019 Conference on Information Communications Technology and Society (ICTAS)*. IEEE, 1–5.
[100] Chelsea Myers, Anushay Furqan, Jessica Nebolsky, Karina Caro, and Jichen Zhu. 2018. Patterns for how users overcome obstacles in voice user interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–7.
[101] Ali Asghar Nazari Shirehjini and Azin Semsar. 2017. Human interaction with IoT-based smart environments. *Multimedia Tools and Applications* 76, 11 (2017), 13343–13365.
[102] Kenichiro Noda. 2018. Google Home: smart speaker as environmental control unit. *Disability and rehabilitation: assistive technology* 13, 7 (2018), 674–675.
[103] William Odom, John Zimmerman, Scott Davidoff, Jodi Forlizzi, Anind K Dey, and Min Kyung Lee. 2012. A fieldwork of the future with user enactments. In *Proceedings of the Designing Interactive Systems Conference*. 338–347.
[104] Mitali Palekar, Earlence Fernandes, and Franziska Roesner. 2019. Analysis of the susceptibility of smart home programming interfaces to end user error. In *2019 IEEE security and privacy workshops (SPW)*. IEEE, 138–143.
[105] Adrian Paschke. 2006. ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. *arXiv preprint cs/0610167* (2006).
[106] Erika Shehan Poole, Marshini Chetty, Rebecca E Grinter, and W Keith Edwards. 2008. More than meets the eye: transforming the user experience of home network management. In *Proceedings of the 7th ACM conference on Designing interactive systems*. 455–464.
[107] Irina Popovici, Ovidiu-Andrei Schipor, and Radu-Daniel Vatavu. 2019. Hover: Exploring cognitive maps and mid-air pointing for television control. *International Journal of Human-Computer Studies* 129 (2019), 95–107.
[108] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society.

[109] Antonio Rizzo, Giovanni Burresi, Francesco Montefoschi, Maurizio Caporali, and Roberto Giorgi. 2016. Making IoT with UDOO. *IxD&A* 30 (2016), 95–112.
[110] Jennifer A Rode, Eleanor F Toye, and Alan F Blackwell. 2005. The domestic economy: A broader unit of analysis for end user programming. In *CHI'05 extended abstracts on Human factors in computing systems*. 1757–1760.
[111] Zafer Sahinoglu, Sinan Gezici, and Ismail Guvenc. 2008. Ultra-wideband positioning systems. *Cambridge, New York* (2008).
[112] Antti Salovaara, Andrea Bellucci, Andrea Vianello, and Giulio Jacucci. 2021. Programmable Smart Home Toolkits Should Better Address Households' Social Needs. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.
[113] Yasaman S Sefidgar, Prerna Agarwal, and Maya Cakmak. 2017. Situated tangible robot programming. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 473–482.
[114] Emmanuel Senft, Michael Hagenow, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021. Situated live programming for human-robot collaboration. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 613–625.
[115] Ben Shneiderman. 2000. The limits of speech recognition. *Commun. ACM* 43, 9 (2000), 63–65.
[116] Danny Soares, João Pedro Dias, André Restivo, and Hugo Sereno Ferreira. 2021. Programming iot-spaces: A user-survey on home automation rules. In *International Conference on Computational Science*. Springer, 512–525.
[117] Benjamin K Sovacool and Dylan D Furszyfer Del Rio. 2020. Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies. *Renewable and sustainable energy reviews* 120 (2020), 109663.
[118] Evropi Stefanidi, Maria Korozi, Asterios Leonidis, and Margherita Antona. 2018. Programming intelligent environments in natural language: an extensible interactive approach. In *Proceedings of the 11th pervasive technologies related to assistive environments conference*. 50–57.
[119] Steven L Tanimoto. 2013. A perspective on the evolution of live programming. In *2013 1st International Workshop on Live Programming (LIVE)*. IEEE, 31–34.
[120] Thora Tenbrink. 2017. Situated interaction with a smart environment: Challenges and opportunities. *KI-Künstliche Intelligenz* 31, 3 (2017), 257–264.
[121] David R Thomas. 2006. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation* 27, 2 (2006), 237–246.
[122] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 803–812.
[123] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. 2016. Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 3227–3231.
[124] Michel Vacher, Benjamin Lecouteux, and François Portet. 2015. On distant speech recognition for home automation. In *Smart Health*. Springer, 161–188.
[125] Jessica Van Brummelen, Kevin Weng, Phoebe Lin, and Catherine Yeo. 2020. Convo: What does conversational programming need?. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.
[126] Terry Winograd. 1995. From programming environments to environments for designing. *Commun. ACM* 38, 6 (1995), 65–74.
[127] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769* (2018).
[128] Lefan Zhang, Weijia He, Olivia Morkved, Valerie Zhao, Michael L Littman, Shan Lu, and Blase Ur. 2020. Trace2TAP: Synthesizing trigger-action programs from traces of behavior. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–26.
[129] Valerie Zhao, Lefan Zhang, Bo Wang, Michael L Littman, Shan Lu, and Blase Ur. 2021. Understanding Trigger-Action Programs Through Novel Visualizations of Program Differences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–17.

A ADDITIONAL USER STUDY DETAILS

This section includes additional user study details, including device configuration list, introduction scripts, and the interview questionnaires.

A.1 Introduction Scripts

- System and Environment Introduction

(1) Background

You have just moved into a new apartment with many smart devices in the living room and an intelligent assistant, XiaoCheng.

(2) Smart assistant and Interaction
XiaoCheng will always hear you. You can say hello to it, "hi, XiaoCheng" (The system responds). You do not always have to say wake-up words or wait for a response. By communicating with it, you can get information about the environment, like temperature and humidity, or control all the devices in the room, like this: "XiaoCheng, turn on the living room lights" (The system responds and turns on the ceiling light). In addition, our system can detect your position, posture, movements, gestures, etc., so you can control it through natural body expressions, like this: "XiaoCheng, turn on this light" (Point to the floor lamp at the same time, then the system turns on the floor lamp). XiaoCheng is brilliant, so you can communicate with it as naturally as possible. In addition, you can also directly control the device through the light switch and the physical button on the device. We provide remote controls for some devices (TV, air conditioner, curtains, ceiling lights). The control buttons for some devices (fans, floor lamps) are on the device.
(3) Smart devices
You can control all 16 smart devices in this room. On the coffee table are all the light switch buttons. This is a floor-to-ceiling window equipped with a smart curtain. There are also eight smart lights, including a ceiling light, two downlights above the TV, three downlights above the sofa, an ambient light strip, and a floor lamp. With the button or voice command, you can control the switch, brightness, and color temperature of these lights. You can also switch between different color modes of the floor lamp and the ambient light strip. Audio and video equipment includes a TV and a speaker. As for environment-related devices, we have an air conditioner, a fan, a humidifier, and a floor sweeper.
(4) Automation Configuration
XiaoCheng also supports the configuration of various smart home automation. In the later process, we will give you the 8 scenarios (showing users the script's document), you can explore your specific ways to configure your automation rules by communicating with XiaoCheng and the whole smart environment. You can query, create, modify, and delete the rules. You can express the rules naturally and freely to define some automation or configure the environment. When the configuration is complete, you can use your way to test the rules to check if the effect meets your expectations.
(5) Feedback Interface
During the experiment, you can get real-time feedback from XiaoCheng and the smart environment. If there is a problem, you can communicate with XiaoCheng for information. To confirm the rules, we provide you with a visual display to exhibit the configuration status and the content of the rules. This interface is only used to confirm the information of the rules, and you cannot directly operate the interface to edit the rules. In addition, you can also ask XiaoCheng to broadcast the rules.

A.2 Interview Questionnaire

- **Open Questions:** Ask about the overall feeling of in-situ programming and the main problems encountered during the enactment process.
- **Questions on Configuration Process:**

(1) What is the thinking process when configuring rules?
(2) What interaction interface do you prefer to use?
(3) Different feelings about the configuration of different rule types?
(4) Different feelings about the configuration of different parameter types (human, environment, device)?
(5) Preferences for feedback during configuration?
(6) What is the most important concern in the process of configuration?
(7) What configuration methods will be used and why?

– **Questions on Test Process:**
(1) What situation will you choose to test, what is the thinking process?
(2) What methods will be used to test the rule, and why?
(3) Preferences for feedback during testing?
(4) What is the most important concern in the process of the test?
– **System Expectations:**
(1) What kind of assistance do you prefer from intelligent agents?
(2) What do you like and don't like about the system?
– **User Study Optimization:** Score immersion rate: 1-5 and talk about feelings and suggestions about the overall user study process and tasks.

A.3 Device Configuration List

Table 2 illustrates the list of all the smart devices configuration and the supported interaction types.

Table 2. Smart Devices List

Smart Devices List						
Category		**Devicename**	**Functions supported in the Wizard environment**	**Supported interaction modes**		
Function	**Product**			Remote control	**Button**	**Voice**
Light	Lamp (8)	Ceiling light	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Downlight1 (above the TV)	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Downlight2 (above the TV)	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Downlight3 (above the sofa)	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Downlight4 (above the sofa)	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Downlight5 (above the sofa	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease)		1	1
		Floor lamp	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease), Color		1	1
		Ambient light strip	Switch, Brightness (set, increase, decrease), Color temperature (set, increase, decrease), Color, Rhythmic pattern, Eight special modes		1	1
Audio & Video	Television (1)	Smart television	Switch, Volume (set, increase, decrease), Main functions on the remote control (up, down, right, left, home, choose) Open an app directly: 1: Bilibili 2: Desktop 3: Photo album 4: Weather 5: Tencent Video 6: Settings 7: Timed Reminder 8: Video headlines 9: Wireless projection 10: Calendar 11: Mango TV 12: Keep 13: QQMusic	1	1	1
	Sound (1)	Smart sound	Play, Pause, Volume (set, increase, decrease), Next / Previous song			1
Environmental conditioning	Curtain (1)	Curtain	Open, Close, Set position, Stop at current position	1		1
	Temperature & humidity (3)	Humidifier	Switch, Mode (Level1, 2, 3, Constant temperature mode), Humidity (set, increase, decrease)		1	1
		Fan	Switch, Fan speed, Mode (Natural wind, Normal wind), Oscillation, Rotation (clockwise and counterclockwise)		1	1
		Air conditioner	Switch, Mode (heat, cool, dry, fan only), Temperature (set, increase, decrease)	1		1
	Cleaning equipment (1)	Floor cleaning robot	Turn on, Mode (Sweep, Mop, Sweep and Mop), Go back to base, Stop		1	1