Fast-Forward Reality: Authoring Error-Free Context-Aware Policies with Real-Time Unit Tests in Extended Reality

Xun Qian*
qxziuan@gmail.com
Meta Reality Labs Research &
Purdue University

Tianyi Wang
tianyiwang@meta.com
Meta Reality Labs Research

Xuhai Xu
xoxu@mit.edu
Meta Reality Labs Research &
MIT

Tanya R. Jonker
tanya.jonker@meta.com
Meta Reality Labs Research

Kashyap Todi
kashyap.todi@gmail.com
Meta Reality Labs Research

Figure 1: Overview of the Fast-Forward Reality workflow: (a) An end-user initiates an authoring session of a context-aware policy (CAP) while being immersed in an Extended Reality (XR) authoring environment. The user includes *context instance*s into the CAP by selecting the corresponding XR icons. (b) The system adaptively generates multiple *unit test case*s based on the *context history* of the user's everyday routines and the CAP, and highlights them with XR visualizations. (c) The user acts out the suggested *test case*s by in-situ actions and selections of the involved *context instance*s to intuitively validate whether the CAP reacts as expected under each *test case*. (d) The user refines the CAP after noticing potential improvements and repeats the validation until the CAP is error-free under all *test case*s.

ABSTRACT

Advances in ubiquitous computing have enabled end-user authoring of *context-aware policies* (CAPs) that control smart devices based on specific contexts of the user and environment. However, authoring CAPs accurately and avoiding run-time errors is challenging for end-users as it is difficult to foresee CAP behaviors under complex real-world conditions. We propose Fast-Forward Reality, an Extended Reality (XR) based authoring workflow that enables end-users to iteratively author and refine CAPs by validating their behaviors via simulated *unit test cases*. We develop a computational approach to automatically generate test cases based on the authored CAP and the user's context history. Our system delivers each test case with immersive visualizations in XR, facilitating users to verify the CAP behavior and identify necessary refinements. We evaluated Fast-Forward Reality in a user study (*N*=12). Our authoring and validation process improved the accuracy of CAPs and the users provided positive feedback on the system usability.

CCS CONCEPTS

• **Human-centered computing → Mixed / augmented reality**; **Interactive systems and tools**.

KEYWORDS

Context-Aware Policy, Extended Reality, Validation, Unit Test

1 INTRODUCTION

Developments in ubiquitous computing [96] and smart environments [19] have enabled automating functionality of Internet-of-Things (IoT) and smart devices. *Context-aware policies* (CAPs) can define behaviors of these devices under personalized contexts of

end-users and environments [24] (e.g., turn on smart lights after sunset). With trigger-action programming [84], end-users can author customized CAPs by specifying context factors as triggers (e.g., 'after sunset' and 'leaving home'), and smart functions as actions (e.g., 'turn on the lights' and 'turn off the A/C'). To address diversified user needs, researchers and commercial products have greatly expanded the range of context factors to include a variety of environmental identities (e.g., time, temperature, weather, smart objects [3, 26, 42, 62, 102]), as well as user's location [49, 89, 101], status [104], and activities [92, 98]. While most existing tools focus on enabling end-users to author CAPs with great flexibility, a *runtime* issue emerges: it is challenging for users without programming skills to verify that the customized CAPs will behave as intended under varying contexts.

The following two scenarios illustrate how the authored CAPs may deviate from the user's original expectations. The first CAP, *"Dining"* → *"Turn on the TV"*, was authored on a relaxed evening, intended to automate an entertaining event while dining. However, this CAP is unexpectedly triggered another day when the user eats sandwiches while busily working in the study room. The second CAP, *"Doing yoga in front of the living room TV at night"* → *"Play the yoga playlist"*, was created by adopting the exact contexts during the authoring process. On another morning, the user expects music while doing yoga in the bedroom, yet, the authored CAP will not activate. These examples highlight two types of inaccuracies that frequently occur in CAP authoring. The first user forgot to consider the location and activity factors. So, the CAP is **under-specified** and may trigger actions in unwanted scenarios (false positives). Conversely, the second CAP is **over-specified** since the user did not realize that redundant context factors were added so that the CAP could not be triggered in other desired scenarios (false negatives). Such failures, if happen frequently, can lead to annoyance, frustration, loss of trust, and ultimately abandonment of such systems [64]. To avoid these errors, systems should provide users with intelligibility [8] such that users can accurately predict the functionality of an authored CAP under various scenarios. However, prior authoring tools and workflows have not investigated how they can address this crucial challenge. Hence, our work is motivated by the need for CAP authoring systems that assist end-users in *proactively* eliminating potential inaccuracies in CAPs before deploying them.

In the software engineering industry, **unit testing** [107] demonstrates a systematic approach to effectively avoid run-time errors [34, 76, 87]. It is worth exploring whether this principled method could be used to ensure correctness in the context of end-user authoring of CAPs. A high-quality *unit testing* relies on a set of carefully constructed **test cases** that is not only comprehensive enough to cover all corner cases but also precise enough with no redundancy. Nevertheless, unlike expert programmers and developers, end-users lack the expertise. If asking users to manually design *test cases*, they may be biased to the present contexts or consider contexts that are irrelevant to the current CAP [74, 80]. To this end, we aim to assist common users in generating *test cases* that are tailored to their local contexts, life routines, and the authored CAP. This way, users can efficiently validate the CAP in scenarios that are highly possible to happen in the runtime.

In addition, expert software engineers have the expertise to determine the expected outputs of each *test case* and identify errors

in the original function. In CAP authoring, considering that CAPs are deployed in end-users everyday environments (e.g., home, office), *test cases* include diversified combinations of not only general digital contexts (e.g., current time, having a meeting) but also space-sensitive contexts that vary across each deploying environment (e.g., smart object states or human actions at different locations). For instance, in an environment with multiple smart lamps around several couches in the living room, it introduces ambiguity to describe *test cases* that involve a specific smart lamp using typical text-based unit test tools [71]. Further, it requires additional mental efforts for end-users to distinguish the imaginary contexts of a *test case* from the present contexts. Hence, we aim to design an intuitive way to deliver each *test case* so that non-expert users can effortlessly interpret it as a real scenario that may happen in the local environment, facilitating users to specify the desired performance and identify errors in the authored CAP.

With these major considerations, we propose **Fast-Forward Reality**, a novel system that enables users to validate and make corrections to CAPs during authoring by following the "*test-fault-correct*" routine of unit testing. Fast-Forward Reality first introduces a computational framework that supports generating high-quality *test cases* tailored to each user and the deploying environment. The framework organizes a user's context history perceived by intelligent technologies (e.g., computer vision, IoTs, head-mounted devices) as a series of contextual combination instances. By analyzing the frequency and correlation of different contexts, it understands the user's habits and preferences. Given an authored CAP, the framework can compose *test cases* with specific combinations of related contexts where the CAP has a high possibility to happen in their lives but may perform inaccurately. In this way, the user can efficiently validate the CAP with highly customized *test cases*. Moreover, inspired by the prior works that immerse users in Extended Reality (XR) to facilitate the interpretation of space-sensitive contexts [16, 35, 92], Fast-Forward Reality adopts an XR interface that delivers each *test case* by overlaying (e.g., smart object states), augmenting (e.g., time), and placing (e.g., human action) corresponding context visualizations in-situ in either the physical environment or the virtual replica. The user can intuitively evaluate whether the CAP performs correctly as if experiencing it in real life. Meanwhile, the XR interface serves as the main authoring interface to initiate an authoring session (1a) and iterate the CAP (1d). By repeating the *author-test-refine* process, the user gradually removes the ambiguity of the CAP and becomes confident that the CAP will perform accurately once deployed. In summary, the main contributions of this work are:

- An *author-test-refine workflow* that enables end-users to validate and iterate CAPs at author-time by evaluating their performances via diverse simulated unit tests.
- A *computational approach for generating unit test cases* that are personalized to each user and environment to effectively reveal potential run-time inaccuracies of the CAP.
- An *XR-based authoring interface* that uses immersive visualizations to offer intuitive understandings of contexts presented in test cases, and direct operations to define and iterate the CAP.

2 RELATED WORK

The main contribution of our work is a novel approach for end-users to author and validate CAPs in XR. To this end, we review the current literature on context-aware computing, end-user authoring, validation techniques, and immersiveness in XR.

2.1 Context-Aware Policies for End-Users

With the development of Internet-of-Things (IoT) devices [48] and context-aware systems [1, 77], researchers have explored applications that automatically execute smart functions in everyday life when pre-specified contexts are detected [23]. We describe such automated applications as *context-aware policies (CAPs)*.

Following Dey's taxonomy of contexts [24], four major types of human-centered contexts have been applied for such applications: *identity*, *location*, *status(activity)*, and *time*. While *time* and *identity* (e.g., weather and calendar events) are straightforward contexts that can be easily sensed by computing systems [3, 26, 42], sensing location, activity, and user state is a more challenging problem. Prior works have used external motion sensors (e.g., RFID [95] and UWB [68]), information-theory-based frameworks [75], or predefined privacy-preserving queries [15] to determine user location. Beyond typical IoT applications, user location has been used to provide Ambient Assisted Living (AAL) services for the elderly [39, 57]. Activity detection has been explored using both external [17] and wearable [90] sensors to enable smart policies such as reminders and notifications based on user interactions (e.g., remind a resident that a specific food is about to expire during meal preparation). Further, description logic rules [97], and machine learning and deep learning approaches [10, 28, 98] have been applied to infer human activities from raw sensor data towards deploying activity-aware policies such as starting workouts or launching applications. Light and noise sensors have been used to infer other human states such as tiredness and whether a person is sleeping to automate smart home accessories such as lights and window blinds [104]. IMU sensors have been used for fall detection [33, 43] to provide automatic AAL services. Researchers have also proposed several approaches to perceive the status of IoT objects and leverage them into CAPs. Smart medicine boxes and bottles [9, 62], water bottles [11], and refrigerators [31] are capable of counting and identifying their contents towards providing relevant CAPs such as smart reminders.

While many context-aware systems can provide generalized automatic services, the habits and needs of end-users vary in daily life. Consequently, context factors and desirable policies are also highly dependent on personal preferences [24]. Thus, enabling end-users to *author* CAPs that include user-defined contexts and personalized smart functions has been an area of great interest.

2.2 End-User Authoring of Context-Aware Policies

To help end-users author context-sensitive applications, 'trigger-action' programming has been broadly adopted following the findings from an elicitation study [26]. Here, an end-user specifies a policy with a series of contexts as the trigger and a smart function as the action. At run-time, when all the specified contexts are present, the system executes the corresponding action.

Block-based programming in the style of 'if this then that' [42, 83] has been widely used in commercial products such as Alexa routine [3] and Apple Shortcuts [78]. Here, a user selects contexts represented using 2D icons into the 'this' block and adds smart functions into the 'that' block. Alternatively, the user can define such CAPs through sketches and descriptions [26], programming-by-demonstration [25, 46], or by segmenting maps for location-aware applications [49]. Augmented Reality (AR) and Virtual Reality (VR) authoring techniques have recently been explored to provide users with spatial awareness and immersiveness. CAPturAR [92] supports users to author activity-based context-aware applications when referring to their past activities that are visualized using AR avatars. By defining proxemics with physical surroundings using a mobile-device-based AR authoring interface, ProGesAR [101] and ProObjAR [102] enable users to include locations, object movements, and gestures into CAPs. Further, Ivy [29] immerses users into the digital twin of an indoor environment to define logic connections among smart objects via visual programming, while Haidon et al. [36] enables a caregiver to author CAPs tailored to a resident's habits by selecting the remote resident's spatial-sensitive contexts that are mapped to the caregiver's local AR space using a semantic mapping approach.

Existing efforts have mainly targeted end-user challenges in authoring CAPs with specific context triggers and actions. Some prior research has supported users to manually test authored CAPs by either selecting the triggering contexts on a 2D GUI [26, 74] or by demonstrating the human activities and proxemics in AR [92, 101, 102]. However, prior works do not investigate whether the user-authored CAPs perform according to user expectations in diversified run-time scenarios. When such automation policies behave unexpectedly, users can lose trust and abandon these systems [64]. In this paper, we aim to address this issue by facilitating the validation and refinement of CAPs during authoring.

2.3 Validation of Context-Aware Policies

During real-world usage, end-users may encounter complicated contexts with diverse and unexpected edge cases [21, 54, 58]. Thus, validating CAPs under such varied scenarios and resolving discrepancies has been widely explored in professional context-aware application development. By adopting the approach of unit testing [37, 107], researchers have proposed multiple architectures and frameworks to help application developers design effective unit test cases to validate context-aware applications.

First, prior works propose data-driven approaches to generate diversified contextual scenarios to reflect the complex run-time conditions [93]. CASS [69] proposes an architecture to automatically generate virtual sensory data for context-aware system developers to rapidly test and identify conflicts in the context-aware systems. TestAware [55] allows developers to design test cases for mobile context-aware applications by downloading, replaying, and emulating contextual data on either physical devices or emulators. On the other hand, it is impractical to validate countless real-world scenarios. The number of test cases should be constrained while maintaining the effectiveness of the validation. Context diversity [87, 88] has been used to address this concern, where the distance between test cases is used to ensure diversity. Other model-based

approaches [34, 63, 76, 103] have been developed to efficiently generate test cases that can fulfill the validation requirements. Similarly, this concern has been addressed in the research area of adaptive systems. ScalAR [72] adopts the Genetic Algorithm to enable adaptive AR application authoring within a small scale of indoor layouts. It considers how to generate the most representative and diversified scenes with spatial and identity variations to help AR designers efficiently validate the AR element behaviors and identify failure cases. Inspired by these prior arts, we distill 3 major considerations for validating CAPs: (1) test cases should cover diversified and nuanced real-world scenarios, (2) test cases should be diverse but concise so that all potential scenarios can be captured with a limited number of test cases to ensure an efficient validation process, and (3) test cases should be able to effectively detect faults in applications.

Creating test cases that meet these criteria is a challenging task. Unlike professional developers, end-users may lack the expertise to carefully design suitable test cases with varying complexity that cover as many corner cases as possible. Further, unlike general context-aware applications and programs, user-authored CAPs are highly personal; as such, test cases should be personalized to each user while addressing diverse scenarios. We believe that an automatic approach to generating personalized and diverse test cases can address these challenges.

Furthermore, to fully support novice users in validating and refining their CAPs, how to present these test cases appropriately remains challenging as a majority portion of the contexts perceived by advanced AI modules are associated with the spatial information (e.g., locations [101], proximity [102], interactions [40, 92], and IoT states [41]). However, the above-mentioned CAP validation tools do not address the need to enable local users to interpret the test cases precisely. Since the test cases are designed by professional developers who have a clear expectation of the outcome and solution to iterate the context-aware functions. Yet, when end-users author a CAP in their local environment without any expertise in designing the test cases, the test cases should be delivered to the users in an easy-to-understand manner. Two difficulties then pop up if using the existing tools: First, as we mentioned in Section 1, spatial-sensitive contexts often cause ambiguity when the referred assets are duplicated in space, especially when a test case may involve multiple contextual elements simultaneously. Prior research [6, 7] has mentioned the inaccuracies caused by pure descriptive narrations. Meanwhile, since end-users do not have the expertise to design the test cases, feelings of non-confident and untrustworthy will be raised if no *feedforwards* can be provided to end-users [27, 85, 86].

Hence, in this paper, we endeavor to adopt an approach that can intuitively represent the test cases to end-users, and more importantly, facilitate the users to rapidly identify the errors in the existing CAPs by thoroughly interpreting the provided test cases.

2.4 Immersiveness in XR

Motivated by prior research, we believe Extended Reality (XR) could provide an effective medium to communicate test cases for validation to users. Enabled by spatial awareness of XR, prior works have used virtual augmentations adjacent to relevant physical entities to facilitate end-users to understand complicated contexts such as

human activities [13, 92], interactions [12, 40, 53, 91], and status of smart objects [41]. Further, while being immersed in XR environments, end-users can experience simulated scenarios free from the limitations of physical contexts such as user status and time. Users can revisit what happened in the past via in-situ XR visualizations of human activities [13, 30, 52] and interactions [73, 99], or virtually immerse into other times and spaces [72, 94, 108]. Finally, through immersive authoring [45], users can intuitively program and author XR applications from within the target deployment environment [38, 65, 66, 105], which enables a real-time and seamless transition between authoring and testing.

To this end, our work develops an XR authoring environment that enables users to (1) iteratively author a CAP, (2) immersively evaluate auto-generated unit test cases with complex contextual scenarios, and (3) directly manipulate test cases and observe simulated results to obtain feedback on whether a CAP performs as expected and make refinements as needed.

3 FAST-FORWARD REALITY

We present Fast-Forward Reality, a novel XR-based workflow that enables end-users to iteratively author accurate CAPs through unit testing. To this end, we develop a computational approach for generating diverse and personalized unit test cases and an authoring interface that provides users with effective CAP validation via immersive visualizations. In this section, we define a set of design goals, that guide the system development, and present a walk-through to illustrate the "author-test-refine" workflow for a specific CAP. Following this, we elaborate upon our technical approach and system implementation.

3.1 Design Goals

In this work, we aim to develop a system that helps end-users author accurate CAPs via real-time validation before deployment. While being immersed in the XR authoring environment, a user can experience multiple unit test cases that are personalized to their previous activities and routines. Through simulated *feedforwards*, the system provides intelligibility and enables refinement when outcomes do not match original expectations. Consequently, users can eliminate potential runtime inaccuracies that may surface during initial authoring. To effectively support users in the authoring and refinement process, we postulate that a system should fulfill the following **design goals**.

(1) **Personalization**: As addressed in prior CAP authoring systems [25, 26, 92], test cases should be closely related to the CAP a user is authoring and personalized to their context history. This enables the user to effectively refine CAPs by examining contextual scenarios that are likely to occur in their everyday life.

(2) **Diversification**: Prior works [55, 69] have addressed the need to diversify the test cases by proposing different frameworks and guidelines. Following their suggestions, to cover a wide range of possible scenarios, test cases should be diverse and capture edge cases that may cause unexpected outcomes.

(3) **Brevity**: Test cases should be concise such that users can clearly identify factors that cause inaccuracies. Further, the number of
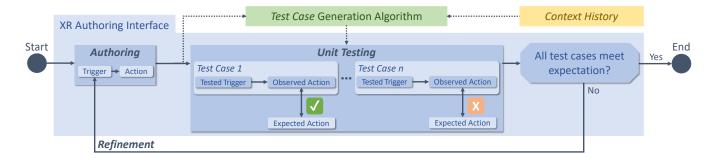
Figure 2: The authoring workflow of Fast-Forward Reality. While being immersed in the authoring environment, a user starts *authoring* to define a target action and initial *context instance*s. The system generates test cases from the user-authored CAP and context history. When the user starts *unit testing*, test cases are visualized to enable validation of the CAP's behavior. If unexpected instances are identified, the user *refines* the original CAP by editing the *context instance*s. The user repeats this process to iteratively refine the CAP until it meets their expectations.

test cases should be constrained such that they are tractable within a limited time [87, 88].

(4) **Interpretability**: As we discussed above, since end-users may not have professional programming expertise, it is cumbersome to interpret a combination of multiple space-sensitive contexts involved in a test case. And hence, it is difficult for end-users to imagine possible iterations of the CAPs. Thus, we propose to leverage the advantages of XR so that users can intuitively understand the complicated contexts and immersively experience the test cases that may happen in the future.

(5) **Seamlessness**: While being immersed in the XR environment, the interactions required for authoring and validating a CAP should be fluent and consistent to ensure seamless transitions in the *author-test-refine* workflow following the findings of prior immersive authoring and spatial programming works [29, 45, 66, 105].

3.2 Target Scenarios and System Walkthrough

In this paper, we identify the research scope where an increasing number and types of contexts can be detected and included in context-aware systems through modern AI modules (e.g., object detection, activity recognition), IoT communication, and AR spatial awareness. Prior works [25, 26, 49, 92, 101] have explored various systems to integrate different contexts into context-aware systems. We acknowledge their contributions and assume that it is straightforward to integrate these modules into a CAP authoring system. We follow example scenarios that are similar to the one shown in Figure 1 to illustrate the system design of Fast-Forward Reality in the next sections. Specifically, a user could live in an apartment with a layout similar to Figure 1. The user regularly wears a head-mounted XR device with real-time object detection and activity recognition capabilities. Meanwhile, smart objects such as smart lights, TVs, music players, and coffee makers are present in this space, while calendar events, dates, and weather information are intrinsically available in their XR devices.

Figure 2 illustrates the workflow of Fast-Forward Reality in a target scenario as mentioned above. Here, we explain it using the example shown in Figure 1, where an end-user tries to author a

CAP to control a smart TV. The user lives in a studio apartment and prefers to watch TV while eating. One day, as the user is watching TV while sitting on the sofa, the user decides to author a CAP to automatically turn on the TV during such situations.

Authoring: The user activates Fast-Forward Reality and begins the authoring process. By selecting in-situ icons that represent 'TV is on' and 'location is sofa' *context instance*s, the user authors an original CAP: *'turn on the TV when I am on the sofa'* (Figure 1a).

Unit Testing: Then, they start the validation process to ensure correctness. A computational algorithm examines the user's context history and the current CAP, and identifies that the user always 'eat' (activity) while 'watching TV', but rarely engages in other activities such as 'reading' or 'using the phone'. However, in the current CAP, the user has not included any activity *context factor*. Additionally, the context history indicates that when the 'TV is on', the 'music player' is typically 'off'. However, this case is not considered in the CAP either. Therefore, the system generates a *test case*: 'eating on the sofa while the music player is off', and visualizes it in the XR environment (Figure 1b). The user interacts with the immersive visualization and enacts the actions of sitting on the sofa and eating, while the music is off, and observes the TV turning on (Figure 1c).

Refinement: Following the highlighted relevant contexts in the *test case*, the user identifies their importance and *refines* the under-specified CAP by adding the two suggested *context instance*s (Figure 1d). The CAP is now more accurate such that the TV is not turned on when the user is engaged in other activities or listening to music. And, the user keeps validating more system-generated *test cases* based on the updated CAP until no refinements are needed. By using Fast-Forward Reality, the user has successfully authored and verified an accurate CAP to turn on their smart TV under desirable contextual scenarios.

3.3 Framework for Authoring

In this section, we elaborate on the framework adopted by the initial authoring and its connections with the context perception techniques. It addresses three concerns: (1) following prior research, how does Fast-Forward Reality detect and structure a user's everyday contexts, (2) during authoring, how does the user define the
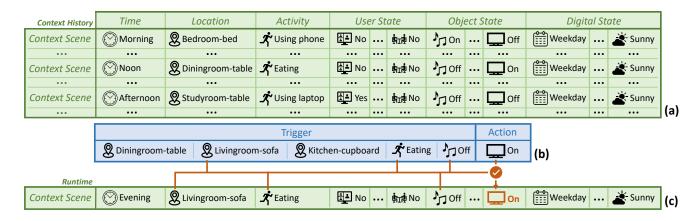
Context History	Time	Location	Activity	User State		Object State		Digital State		
Context Scene	🕐 Morning	📍 Bedroom-bed	🏃 Using phone	📺 No	... 🛏 No	🎵 On	... 🖥 Off	📅 Weekday	...	☀ Sunny
...	...	...	...	...	... ...	...	... ...	...	...	...
Context Scene	🕐 Noon	📍 Diningroom-table	🏃 Eating	📺 No	... 🛏 No	🎵 Off	... 🖥 On	📅 Weekday	...	☀ Sunny
...	...	...	...	...	... ...	...	... ...	...	...	...
Context Scene	🕐 Afternoon	📍 Studyroom-table	🏃 Using laptop	📺 Yes	... 🛏 No	🎵 Off	... 🖥 Off	📅 Weekday	...	☀ Sunny
...	...	...	...	...	... ...	...	... ...	...	...	...

(a)

Trigger					Action
📍 Diningroom-table	📍 Livingroom-sofa	📍 Kitchen-cupboard	🏃 Eating	🎵 Off	🖥 On

(b)

Runtime										
Context Scene	🕐 Evening	📍 Livingroom-sofa	🏃 Eating	📺 No	... 🛏 No	🎵 Off	... 🖥 **On**	📅 Weekday	...	☀ Sunny

(c)

Figure 3: The framework of authoring CAP using Fast-Forward Reality. (a) A user's everyday contexts are recorded as the *context history*, which contains a series of *context scene*s where each one is a collection of concurrently occurring *context instance*s of corresponding *context factor*s that can be detected in the environment. (b) In a user-authored CAP, the 'trigger' contains multiple *context instance*s from the same or different *context factor*s, while the 'action' is a *context instance* of a smart object that reflects the functional state (e.g., 'TV is on'). (c) During deployment, for all the *context factor*s included in the CAP, when the specified *context instance*s are present in the real-time *context scene*, the 'action' is triggered.

components of a CAP, and (3) upon deployment, how does our system determine when should the action of the CAP be executed. As discussed in Section 2, technical solutions have been developed to detect and identify increasing types of contexts that were proposed in early elicitation works [1, 26, 77]. We adopt previously proposed taxonomy of contexts [1, 24] and assume integration with modern AI modules to categorize the contexts we aim to leverage into the following **context factor**s:

- **Time** represents the time of a day, which is commonly used in CAPs: 'turn on all lights in the 'evening'. It is the most straightforward context that can be retrieved.
- **Location** of a user serves as a critical factor in a CAP (e.g., 'turn on the A/C when I enter the living room'). We assume that the user locations can be detected by external sensors [68, 95] or through computer vision with advanced head-mounted devices [61] in real-time.
- **Activity** belongs to the *status(activity)* category [24], representing both general activities and interactions with objects that are detectable via software [97] and hardware [106] based activity detection modules. Considering the structure of currently available activity detection networks and benchmarking datasets [14, 20, 32], we assume activities will be detected as discrete labels (e.g., eating, reading, etc.) given continuous sliding window time series of human skeleton or hand-object interaction as inputs.
- **User state** also belongs to the *status(activity)* category [24]. It represents the spatial-insensitive status of a user. Typical examples include: 'being alone', 'in a meeting', and 'feeling tired'. These contexts can be detected via AI networks or inferred from other contexts (e.g., how many cups of coffee have been drunk). Typically, the outputs are discrete nominal labels.
- **Object state** represents status of objects in the environment. For a smart object such as a coffee machine or a smart pill bottle, its physical states (e.g., not enough water, how many pills left) can be identified but not digitally manipulated. For other smart and IoT

devices, their active status can be altered by a CAP (e.g., 'turn on the TV'). Using IoT communication protocols such as Resource Description Framework [22], and hardware-based localization algorithms such as ultra-wideband [41], such discrete contexts can be detected during runtime.

- **Digital state** is related to *identity* and represents general but unique environmental information provided in general digital devices such as *temperature* and *weather*. Similar to the *Time*, these contexts can also be easily retrieved from the HMD.

For each type of *context factor*, we define **context instance**s as the nominal values (labels) that can be assigned to them. For instance, the location factor can either include object-oriented *context instance*s such as 'living room sofa' and 'dining table', or non-interpretable anchor IDs that are marked by AR devices. Activity can include labels from activity detection models (e.g., 'sleeping', 'walking'). Boolean 'yes' and 'no' states are used to indicate object states ('music player is on'). At any moment in a user's daily life, we define a **context scene** as a combination of *context instance*s, where each *context instance* belongs to one available *context factor* (Figure 3a). Whenever one of the *context factor*s changes, a new *context scene* is registered, and all the recorded *context scene*s form the user's personal **context history** (Figure 3a).

To enable end-user CAP authoring, we adopt the widely-used trigger-action programming paradigm [84]. Here, a user first defines an 'action' to manipulate a target object (e.g., *TV is on*), and then defines a 'trigger' by including multiple *context factor*s (e.g., *time* and *location*), and for each *context factor*, selecting the *context instance*s (e.g., *morning* and *afternoon*) (Figure 3b). During runtime, when a CAP is deployed, for all included *context factor*s, if one *context instance* is present in the current *context scene*, the corresponding smart function ('action') is executed (Figure 3c).
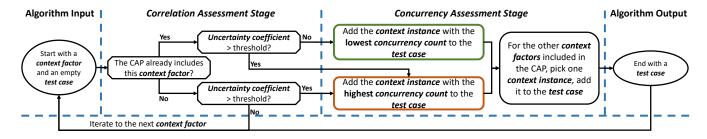
Figure 4: *Test case* generation algorithm. For each *context factor*, the algorithm starts with an empty *test case*. In the *correlation assessment stage*, we investigate whether the processing *context factor* holds a high *uncertainty coefficient* value with the target action, and whether it is already selected into the CAP. Depending on the conditions, the algorithm then processes the *concurrency assessment stage* to select the *context instances* into the *test case*. In one specific condition, the system will skip this process and directly iterate to the next *context factor*, while in other conditions, the algorithm outputs a *test case* and starts to process the next *context factor*.

3.4 *Test Case* Generation in Unit Testing

After a user has created an initial CAP, the main purpose of the unit test stage is to help reveal potential errors (the under-specified and over-specified scenarios discussed before), which are caused by the inaccurate selection of the contexts of interest following the framework introduced before. Note that resolving runtime errors of AI modules is out of this work's scope. Typically, most of the prior CAP authoring [25, 26, 92] and validation [55, 87, 103] systems assume the detection of the contexts is reliable and robust. For instance, an alarm CAP can be validated by changing a simulated time. However, it is impossible to test and refine the CAP when the time itself is wrong. Thus, the *test case* generation algorithm we propose in this paper only aims to resolve the inaccuracies in the user-authored CAPs, rather than the unpredictable AI errors. Yet, we will discuss the potential extension in the Discussion section.

With the above CAP authoring framework, we follow the unit testing approach and develop a computational approach to generate diverse **test case**s that help reveal potential inaccuracies in a CAP. A *test case* consists of a set of *context instances* available in the environment. Our *test case* generation algorithm addresses the design goals of *personalization*, *diversification*, and *brevity* (defined in Section 3.1) through three key strategies:

- **Strategy A - Correlation**: Investigate which *context factor*s largely affect the execution of the current CAP and include appropriate *context instance*s into *test case*s. By doing so, when visiting the *test case*, the user can notice the scenarios that are highly associated with their personal preferences.
- **Strategy B - Intention**: Add one (and only one) *context instance* for each *context factor* that is already in the CAP so that the *test case* is closely related to the user's original idea and realistically reflect a real-world condition where only one *context instance* of a *context factor* can happen at one moment.
- **Strategy C - Simplification**: Exclude irrelevant *context factor*s to avoid distracting the user so that they can focus on the *context factor*s likely to cause errors.

To implement *Strategy A*, as we assume that users will maintain some routines during everyday life, we compute **uncertainty coefficient** [79] to identify *context factor*s. Specifically, this is a conditional information entropy-based approach to asymmetrically

reveal that in the presence of a particular *context instance*, what is the probability that another *context instance* occurs. For instance, a user may watch TV at different times of the day, but is always eating and not using the phone while doing so. In this case, the *uncertainty coefficient* between the 'TV state' and 'time' is low, while 'activity' is high. To apply this in our system, once the user selects a target action, we loop over every detectable *context factor*, and (1) retrieve all past *context instance*s of both the target action and the current processing *context factor* from the user's context history and (2) calculate the *uncertainty coefficient* between the two lists using the *theils_u* function[1]. Further, since these coefficients only reflect the *context factor* level correlations, we also calculate the **concurrency count** for each *context instance* within the *context factor*: (1) fetch all *context scene*s where the target action *context instance* happened and (2) construct a dictionary of all the *context factor*s that saves the counts of *context instance*s that happen in these *context scene*s using the *Counter* function[2]. These *concurrency count*s will help the system determine which *context instance*s to include in the test case after deciding the *context factor*.

Figure 4 illustrates the *test case* generation algorithm. Given the target action and the initial *context instance*s, together with the precalculated *uncertainty coefficient*s and *concurrency count*s, we loop over every *context factor* that is available in the current environment to process the algorithm. First, in the **correlation assessment stage**, we examine (1) whether the *context factor* is already included in the CAP and (2) whether the *uncertainty coefficient* of the currently processing *context factor* is greater than an empirically set threshold (we will explain it in Implementation section). Four conditions will be available and we only process 3 of them in the following **concurrency assessment stage**.

(1) If a *context factor* shows a high correlation with the target action, however, is not included in the current CAP, we initiate a new test case and add the *context instance* that most frequently happens concurrently with the action in the test case. Typically, this case helps eliminate under-specified mistakes. For instance, a user always does specific activities while 'TV is on', leading to a high *uncertainty coefficient*. However, the user has not included

---

[1]https://github.com/shakedzy/dython/blob/master/dython/nominal.py
[2]https://docs.python.org/3/library/collections.html#collections.Counter

any activity in the CAP. So, the system includes the activity that the user mostly does when watching TV in the test case to remind the user of this critical factor in the CAP.

(2) If a *context factor* shows a low correlation with the target action, however, is already included in the current CAP, we initiate a new test case and add the *context instance* that rarely happens together with the action into the test case. Consider the same user, who would watch TV at different times, adds 'evening' into the CAP. The system warns the user about this potentially over-specified error by including 'time' when the user rarely watches TV in the test case.

(3) If a *context factor* shows a high correlation with the target action, and is included in the current CAP, we initiate a new test case if there is a *context instance* that happens more frequently. This condition also targets over-specified CAPs. For example, consider location holds a high *uncertainty coefficient* and the user has already included the dining table in the CAP. However, *concurrency count*s indicate that the user watches TV often while on the 'sofa'. Our system includes 'sofa' into the test case to nudge the user to generalize this *context factor* in the CAP.

(4) Note that the last condition (i.e., a *context factor* with low *uncertainty coefficient* that is not included in the CAP) will not lead to any test cases because we assume that this decision made by the user already meets their personal preferences.

Next, to realize *Strategy B*, for each CAP-involved *context factor* that is not included in the *test case*, the system randomly picks one *context instance*, such that the generated *test case* is consistent with what the user just authored. After iterating over all the N *context factor*s, the algorithm generates $M < N$ *test case*s, where M depends on the previously mentioned threshold. Meanwhile, if the current CAP involves n *context factor*s, each *test case* involves $m \in \{n, n+1\}$ *context instance*s. Finally, *Strategy C* is satisfied by composing *test case*s of only the most critical *context instance*s that may affect the user's decision and omitting low-priority instances. The user can focus on testing out the suggested *test case*s without needing to pay attention to irrelevant *context factor*s. By adopting this principled approach, the system can generate multiple *test case*s and present it to the user.

3.5 XR Authoring Interface

Our XR authoring environment ensures that Fast-Forward Reality follows the design goals (subsection 3.1 of providing *interpretability* during testing and enabling *seamlessness* of authoring and refinement. In our system, a menu is rendered on the user's non-dominant hand encapsulating all buttons for controlling the system (Figure 5a). The 'edit' and 'edit next' buttons allow for editing one of the existing CAPs. The 'new' button initiates the authoring of a new CAP. The 'start validation' button is used for switching between authoring and validation. Finally, the 'save' and 'delete' buttons allow users to store or remove a CAP.

When the user starts authoring a CAP, available *context instance*s are displayed in the XR authoring environment (Figure 5c-1). Users can interact with these elements to include them in the authored CAP. Based on the types of *context factor*s, we place spatial-sensitive *context instance*s directly in the XR environment. Specifically, the location and activity are illustrated using avatars with different poses

(Figure 5c-2); object states are represented using XR icons placed next to the corresponding objects (Figure 5c-3); spatially-insensitive contexts (time, digital state, and user state) are displayed on the user's non-dominant hand with 2D icons (Figure 5c-4). The user can select the checkbox above each *context instance* to add/remove it to/from the CAP. Meanwhile, each *context instance* is color-coded[3] to indicate different selection states: we use blue to represent unselected *context instance*s, pink for those included in the CAP, and red for those in a *test case*. Lastly, a panel is displayed on the user's non-dominant hand (Figure 5b) to illustrate the *context instance*s and activities included in the current CAP.

*Test case*s are generated automatically and made available to the user. When the user starts validation, a *test case* is visualized within the same environment to enable quick interpretation. Users can observe a *test case* as a 'fast-forward' simulation that reveals the consequences of a particular context scenario, thus providing them with *feedforward*. Typically, as illustrated in Section 3.4 and Figure 3, Fast-Forward Reality compares the real-time detection of the AI modules with the user-authored CAP, and visualizes the corresponding output in the XR environment. For instance, if all the conditions are met for a TV controller, the physical TV will be turned on in the environment. Since authoring and testing take place within the same environment, the user can immediately start refining their CAP by modifying the *context instance*s as needed. Note that to reduce users' visual and mental loads, all the XR UIs are only accessible to users during the author-test-refine flow. When users return to everyday life, Fast-Forward Reality will hide all XR visualizations but keep running at the backend.

3.6 Implementation

We use Meta Quest 2 [59] as the target platform for developing the authoring interface, and implemented the system using Unity3D (2020.3.16f1) [82]. Interactions with the interface are currently supported via handheld controllers; however, it is trivial to switch to free-hand interactions supported by Quest 2 and other XR devices [61]. To determine a proper threshold value used in the *test case* generation algorithm, and a suitable number of *context scene*s to capture in the context history, we conducted preliminary tests using a context history collection tool, which is described in the next section. Typically, we generated 5, 10, 15, 20 *context factor*s that may be relevant to a home environment. For each case, we collected 10, 20, ..., 50 *context scene*s. During data collection, we followed an abstract rule to control smart functions and collected diverse *context scene*s with a small amount of noise. Then, we calculated the *uncertainty coefficient*s of the smart function concerning all available *context factor*s. We empirically set the correlation value to be 0.5, then set the minimum number of *context scene*s collected in the context history to be 10 times the number of available *context factor*s in the environment. By doing so, we ensure that the number of *context factor*s that are higher than the threshold will be approximately less than 5.

4 USER STUDY

To evaluate whether the design of Fast-Forward Reality fulfills our design goals (Section 3.1), we conducted a systematic user study. In

---

[3]Additional textual labels can be included to ensure visual accessibility.

Figure 5: The XR-based authoring interface of Fast-Forward Reality. (a) The main menu rendered on a user's non-dominant hand enables users to 'edit' existing CAPs, 'add' a new CAP and author it, 'start validation' of unit *test case*s, 'delete', and 'save' the CAP. (b) An authoring panel displayed on the user's non-dominant hand indicates the action and triggers in the current CAP. (c-1) The immersive XR authoring environment. (c-2) Location and activity *context instance*s are represented using avatars with different poses. (c-3) Spatial states are placed in situ, next to corresponding smart objects. (c-4) The digital state and user state are rendered on the user's hand. All *context instance*s are color-coded to represent different conditions where blue color indicates that the *context instance* is not selected, pink represents that it is included in the current CAP, and red represents that it is included in the current *test case*.

this work, we propose an XR-based workflow that addresses the difficulties of validating complicated CAPs that cannot be straight-forwardly solved by traditional developer-oriented unit testing systems. In specific, since end-users do not have the expertise to design high-quality test cases while in the local environments, we design a computational algorithm that leverages the users' personal history to extract diverse test cases. Moreover, we adopt the immersiveness enabled by the XR technique to show the test cases via in-situ virtual content and support users to intuitively understand each test case by enacting each test case in XR via in-situ simulation. Therefore, the goals of this study include: (1) investigating whether the *test case*s generated by the system effectively meet users' needs in identifying potential inaccuracies of the CAPs, and (2) evaluating whether the XR-based environment helps users easily understand the test cases and identify potential improvements of the CAPs. Note that the main research scope of this work is not to compare whether our system outperforms prior approaches, instead, we aim to illustrate that the system can succeed in solving the research difficulties when dramatically increased spatial-sensitive contexts can be involved in CAP authoring.

4.1 Study Setup

For the study, we designed a virtual one-bedroom apartment (Figure 6a) as the home environment to conduct the user study. We also developed an interactive tool (Figure 6b) to enable end-users to elicit personal *context history* that is appropriate to the virtual home.

4.1.1 Context History Collection Tool. Our authoring process relies on the availability of a user's *context history*, containing *context scene*s that have taken place in the past. Given the constraints of our controlled study and privacy concerns, collecting participants' real *context history* was infeasible. To circumvent this issue, we developed a data collection tool where participants could quickly explicate their everyday activities to create their *context history* in the provided virtual home. We asked participants to specify *context instance*s temporally by prompting them to imagine their daily routines. Note that one of the key observations that has sparked HCI

interest and research on developing CAPs is that humans do follow some routines and have personalized preferences in their everyday lives. Thus, we aimed to provide an environment that resembled each user's personal living environment and asked them not to deliberately create random non-representative context histories, but to follow some general personal routines.

As shown in Figure 6b, the data collection interface is separated into four parts. In the top left corner of the UI, a 'new sequence' button allows users to create a new 'day' of living in the virtual apartment, while a clock illustrates the time. After pressing the 'new sequence' button, a user can select the two arrows next to the clock to change the time (Figure 6c-2); the numeric time is encoded into nominal values (e.g., evening). In the center, the floor plan (top-view) of the virtual apartment is displayed (Figure 6c-1). All available location *context instance*s are represented with selectable blue circular nodes. By selecting a node, the available *context instance*s of the two types of *context factor*s, activity, and object state are shown as lists. The participant can click an activity or an object state to indicate what will happen at the specified time. Further, available user states are listed below the clock for the user to select. All selectable *context instance* nodes are illustrated in Figure 6c-2; a textual description of the *context instance* appears when the cursor hovers over the corresponding node. User-specified *context instance*s are visualized along a timeline (Figure 6c-3). The participant can click a block/dot to delete the corresponding *context instance*. After specifying all *context instance*s at a specified time, the participant can change the time and repeat all operations until the 'current day' is finished.

Once the user presses the 'save sequence' button, for every time-step when the user specifies at least one *context instance*, a new *context scene* is created and stored. If the user does not specify any *context instance* of particular *context factor*s, they are set to default values in the *context scene*. By consecutively eliciting multiple days, the user successfully provides a *context history*.
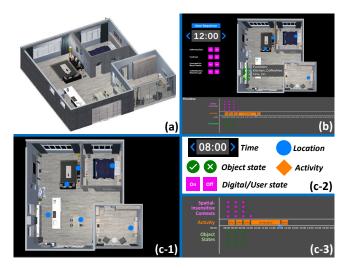
Figure 6: (a) The virtual environment used for the user study. (b) The *context history* collection tool. The top left corner below the 'new/save sequence' button contains spatial-insensitive contexts, *time* and user-specific *digital/user state*s (e.g., 'is working', 'feel tired', 'consume too many coffees', and 'not enough water in the coffee machine') (c-1) The floor plan of the target environment. By clicking each blue dot, pre-designated spatial-sensitive *context instance*s such as *object state*s and *activities* can be toggled together with their *location*s (e.g., 'coffee machine is on/off', 'stay', 'cooking', and 'eating'.) (c-2) The available *context instance*s that are visualized in different icons according to the *context factor*s. Typically, location is spatially overlaid on the floor plan, while available activities and object states are displayed as lists after selecting the location. (c-3) The timeline indicating all the recorded *context instance*s. Explanation of the *context instance* will appear when hovering on the icon. One can delete a *context instance* by clicking the block/dot.

4.2 Study Method

One of the major research questions we aim to investigate is the efficacy of Fast-Forward Reality and whether validation and refinement at author-time can improve the accuracy of CAPs before deployment. To inform our system design, we derived a set of five design goals. We conducted a within-subject comparative study to assess whether the test case generation algorithm can generate effective tests that fulfill the requirements of the first design goals 1–3 (*personalization*, *diversification*, *brevity*). Design goals 4–5 (*interpretability*, *seamlessness*) were assessed via a subjective questionnaire and interview.

Since the main goal was to prove the effectiveness of our *test case* generation algorithm, we designed a baseline system that adopted the same XR authoring interface, while turning off the *test case* generation capability. In specific, we provided the users with the same XR UIs as illustrated in Section 3.5. The only controlled factor was that during the test and refine phases, the users could manually toggle different *context instance*s and spatial movements to validate their CAPs. Note that, the users had full freedom to stop testing if

they felt the authored CAP was already precise enough. In the study, each participant was asked to author two CAPs using Fast-Forward Reality and the baseline system respectively in a virtual apartment (Figure 6a).

Participants: 12 participants were invited to the user study (8 males, 3 females, and 1 non-binary; mean age=27.83, SD=3.16). No specific background or experience was required during the recruitment. 9 participants had used smart objects (e.g., Nest Thermostats and Philips Hue), and 3 out of these 9 had authored everyday automation (e.g., Alexa) at least once. 11 participants had previously used XR applications (e.g., Pokemon Go and Beat Saber), while 5 users had developed XR applications. None of the participants had seen or used our system before the study.

Procedure: In the author-test-refine workflow, users do not complete the authoring in one shot, but continue modifying their CAPs until they feel satisfied with all the system-provided test cases. Thus, the most effective way to evaluate our system is to investigate the performance of the CAPs validated using the test cases. Overall, each user was asked to use Fast-Forward Reality and the baseline system to author 2 CAPs: (1) a CAP that controls a smart TV in the living room of the virtual apartment and (2) a CAP that controls a smart music player that can play music in the entire apartment. To provide the users with a more realistic feeling of living in the virtual apartment, we sent a pre-study survey to the users before they came, which included (1) the study background, (2) the virtual home environment floor plan (Figure 6a), (3) the 2 target smart functions, and (4) the available types of *context factor*s. Then, the survey asked the users what *context factor*s and *context instance*s would be considered when they wanted to control the two smart functions. We then prepared the virtual apartment for both the 2D tool and the XR authoring environment accordingly. Each user signed a consent form stating that no reimbursement was provided and the user preserved the right to terminate the study whenever they wanted. After a user arrived, the researcher introduced the background and definition of CAPs, and the user collected the *context history* using the 2D interface. Then, the user entered the XR authoring environment to complete the 2 authoring tasks using the 2 systems while considering data counterbalancing among all the users. During each task, we recorded the final authored CAP, completion time, number of iterations of the CAP (if a user added more than one *context instance* at one time, it was still marked as one iteration), and number of *test case*s created/validated. After each task, the user was asked to complete a Likert-type survey towards the feedback on the validation process. At the end of the study, the user completed another Likert-type questionnaire regarding the XR authoring environment and the entire usage experience, together with a Standard Usability Scale survey. Meanwhile, a conversation-like interview was conducted to collect the user's subjective feedback on the system.

4.3 Study Results

Based on the pre-study survey results, we included 8 types of *context factor*s in the virtual apartment. Time, location, activity, TV state, and Music player state were available for all the users, while 'is working', 'is alone', 'feel tired', 'weather', 'having a meeting', and 'playing video games' were provided for different users based on

their survey responses. Meanwhile, the available activities also varied (e.g., 'doing exercise' and 'smoking' were provided for some users). During the collection of the *context history*, we collected M=81.9 (SD=10.96) *context scene*s, and selected 75% of the data for the *test case* generation algorithm and 25% for accuracy evaluation discussed later.

We first report the analysis of the users' operations during the authoring processes. Note that these statistics and observations do not explicitly address our design goals, but provide implications of the usefulness of our system. Overall, the users finished each task using the baseline system with 5.0 (SD=1.86) minutes, while 6.25 (SD=1.82) minutes using Fast-Forward Reality. Within the entire authoring time, 2.25 (SD=1.60) minutes were spent for manually validating the CAPs, and using our system, 3.58 (SD=0.90) minutes were taken. On average, the users generated 1.25 (SD=1.06) *test case*s. We observed that using the baseline system, most users simply checked the CAP by selecting the *context instance*s they added during the validation stage. Using Fast-Forward Reality, 5.25 (SD=1.66) *test case*s were evaluated by the users. Note that although approximately 4 times more *test case*s were viewed, validation time was not greatly impacted. This can be attributed to the rationale that with the baseline system, users spent more time thinking about what *context instance*s could be used to validate the CAPs. Furthermore, the users conducted 1.0 (SD=1.28) iterations of the CAPs after the manual validation to add more constraints to the CAPs. We also observed that the users tried to make the CAPs more accurate by adding more *context instance*s even before the validation. *"I used IFTTT before, this was how I created a rule by just adding those important factors. It's always better to add more constraints to make it more accurate, right?" (P6)* In contrast, with the help of our system, 3.92 (SD=1.44) iterations were performed. Typically, we observed not only the 'addition' of the *context instance*s, but also the 'removal' of the *context factor*s after visiting some *test case*s. While these generalized observations partially substantiate our proposed approach, in the following analysis, we further investigated the accuracy of the CAPs and the users' subjective feedback systematically.

4.3.1 Test case Generation Evaluation. We first report the qualitative feedback on the *test case* generation algorithm from the 7-point Likert-type questionnaire. Typically, the users were asked to answer the same set of questions after experiencing Fast-Forward Reality and the baseline system respectively. The questions and results are shown in Figure 7a-1. Besides the general analysis, we conducted a Wilcoxon Signed-Rank Test to investigate whether the feedback on the *test case*s was significantly different between the two systems. This test approach specifically targeted within-subject non-parametric data while no normality test was needed.

As noted in the *personalization* design goal, the *test case*s should be closely related to the users' everyday activities (Q1) and the currently authoring CAPs (Q2). As shown in the results, our system was positively welcomed by the users in the relevance to the daily actions of the *test case*s (M=5.4, SD=0.79), and showed a significant difference (Z=-2.8, p=0.004) compared with the baseline system (M=3.3, SD=1.44). *"One [test case] where I ate food in the study room was definitely what I would do. It was one of my personal preferences." (P4)* Regarding the relevance to the CAP, our system received a more decent rating (M=5.5, SD=0.80), compared with

the baseline system (M=2.8, SD=1.29). A significant difference was identified in this question (Z=-3.1, p=0.002). *"Using the [baseline system], I had no idea of how to pick more [context instance]s since the only idea I had was already added in the policy. But the [Fast-Forward Reality] could give me some good examples, such as 'having a meeting', and 'working on my laptop in the music player task', which was definitely what I would care about." (P3)* Another key motivation when generating *test case*s is whether they could help users realize potential mistakes in the CAPs (the *diversification* and *brevity* design goals). The users highly appreciated that the *test case*s could contribute to the refinement against the CAPs (Q3: M=6.1, SD=0.90, and Q4: M=5.9, SD=0.79). These ratings were significantly higher than the baseline system (Q3: M=2.4, SD=1.00 and Q4: M=2.3, SD=1.23): for Q3, Z=-3.1, p=0.002, and for Q4, Z=-3.1, p=0.002. *"I liked the idea of showing some other potential conditions of one trigger. For the music player, when I first authored it, I only added 'evening' because I was thinking about eating dinner, but [Fast-Forward Reality] also reminded me of cooking in the morning, which would also happen." (P11)* "The system not only informed me which factor I should consider, but I could also directly use that case in my policy, such as it suggested me adding not working and not listening to music into the TV policy." (P1) "What made me surprised was that your system led me to think more about my policy. When I saw that eating at the dining table case, I started to think maybe I should add doing workouts in the living room to the TV policy as well." (P7)* Last but not least, the users felt more confident after they validated the CAPs using Fast-Forward Reality (Q5: M=5.8, SD=1.06), and the baseline system received a significantly lower rating (M=2.7, SD=0.98) with Z=-3.0, p=0.003. *"Actually, only after I used your system did I realize how bad it was when I created that music player rule using the [baseline system]." (P2)*

Besides the users' subjective feelings, we investigated whether the performance of the CAPs was improved after the validation. Given a user-authored CAP, we used the rest of the *context history* as the ground-truths to calculate: (1) precision = $TP/(TP + FP)$, (2) recall = $TP/(TP + FN)$, and (3) F-score = $2 \cdot precision \cdot recall/(precision + recall)$, where TP represents 'true-positive' indicating the target action is expected to be executed while the CAP successfully triggers it; TN represents 'true-negative' where the CAP accurately stays silent when the target action should not be triggered; FP (false-positive) means the CAP mistakenly trigger the target action but the user does not need it; FN (false-negative) means the action should be executed, but the CAP does not work. We also conducted a paired t-test after the data passed the normality test for the significance analysis (Precision: |Kurtosis|=0.88<2.0 and |Skewness|=0.36<2.0; Recall: |Kurtosis|=0.88<2.0 and |Skewness|=0.36<2.0; F-score: |Kurtosis|=0.88<2.0 and |Skewness|=0.36<2.0). The results are shown in Figure 7a-2.

Using our system, the precision of the CAPs reached 90.6% (SD=0.01), which was significantly higher than that of using the baseline system (70.5%, SD=0.08): t(11)=-2.51, p<0.05. Similarly, the recall was significantly increased from 32.1% (SD=0.05) to 83.3% (SD=0.03) with t(11)=-7.50, p<0.005. Further, the overall F-score was improved from 38.2% (SD=0.02) to 85.4% (SD=0.01) with t(11)=-10.10, p<0.005. The statistical analysis indicated that by using our system, the users could author more accurate CAPs via the additional validation stage. Meanwhile, we noticed that using our system, the recall
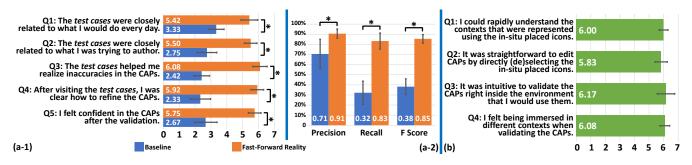
Figure 7: (a-1) The survey results of the quality of the *test case*s generated by the users using the baseline system and by Fast-Forward Reality. (a-2) The accuracy results of the CAPs authored by the users using the two systems. (b) The subjective feedback on the XR authoring environment.

and F-score were greatly increased if compared with the precision. According to the definition of precision and recall, the numbers of the FN reduced to a greater extent than that of the FP when using Fast-Forward Reality to validate the CAPs. One main reason was that when using the baseline system, no assistance was provided for the validation, thus many users added additional constraints that made the CAPs more over-specified. On the other hand, when using our system, we observed that after visiting some *test case*s, the users not only added more *context instance*s to the CAPs, but also removed some *context factor*s that were originally included. *"When using that [baseline system], there was only one scenario in my mind, so I just created that specific CAP When I used [Fast-Forward Reality], those suggestions let me realize the CAP was too constrained." (P3)* Another important observation was that although the overall accuracy was significantly increased, the CAPs rarely reached 100% accuracy. It was because in everyday life, a person could not strictly follow one single CAP as the routine. There always exist scenarios that have never happened before. We will discuss it in the Limitation section in terms of how to address this issue.

4.3.2 XR Authoring Environment Evaluation. To address the design goals 4–5 (*interpretability* and *seamlessness*), Fast-Forward Reality leverages XR to build an immersive authoring environment for end-users to experience *test case*s and iterate the CAPs. Using a 7-point Likert-type survey, we evaluated whether the users welcomed the features of the authoring interface. The results are illustrated in Figure 7b.

We received complimentary feedback on the feature allowing users to try out the *test case*s while being immersed in the XR environment (M=6.1, SD=0.67). *"Because I'll use these policies in the physical environment, it's a good idea to let me try out my [CAPs] in the same environment. When I looked at those virtual icons right above the TV and sofa, I could easily understand what they meant." (P4)* Meanwhile, all the users agreed that it was necessary to validate the CAPs before the real-world deployment (M=6.2, SD=1.19). *"I felt much more confident about my CAP when I could see the TV was on after I sat on the sofa. That instant feedback was realistic." (P5)* Such comments were aligned with the findings in prior works in *feedforward* simulation [27, 86], where users tend to visualize the realistic outcomes instead of descriptions to gain trust against the digital applications. Placing virtual icons that represent the corresponding affordance and functionalities of the smart objects received positive

feedback (M=6.0, SD=0.60). *"For those spatial contexts, it's better to show them in the environment. Otherwise, if I have many smart lights, it's difficult to understand which light I'm referring to using pure texts." (P2)* Users also commented on the comparison between in-situ icons and 2D icons for spatial-insensitive contexts. *"I wonder if those icons can also be attached to the corresponding objects. For example, the 'having-meeting' context can be attached to my calendar. Then, I can fully focus on the physical world when I do the test." (P9)* In addition, the immersive operations provided by our system were also highly accepted by the users (M=5.8, SD=0.83). *"Because the icons were inside the environment, I didn't have to switch between different platforms to create my CAPs." (P10)* Several users who had programming experience (P11, P7, and P1) also pointed out that our system was aligned with the trend of spatial programming [92, 105] for enabling non-experts to join the application development. *"If the program will be used in 3D, it is more intuitive to create and test it in 3D as well. It is a promising way to attract people who do not have coding expertise to create such CAPs in the future." (P11)* Last but not least, a decent Standard Usability Scale score with M=86.0 and SD=6.77 further proved the overall usability of Fast-Forward Reality.

4.4 Discussion

Fast-Forward Reality has been proven effective in facilitating non-expert users to author error-free CAPs via properly generated *test case*s in XR. Given the current scope of the CAP authoring, we discuss more insights and concerns we have distilled from the design process and user study results that inspire future research along the novel author-test-refine workflow of the CAP authoring area.

4.4.1 Scalability of the system and validated CAPs. In the current implementation, we illustrate our system in a virtual environment where spatial-sensitive *context instance*s and corresponding UIs are placed in situ. On one hand, our system could easily adapt to the corresponding AR environment. Specifically, when introducing the framework of context awareness in Section 3.3, we clarified that the scope of this paper lies in the assumption that users live in AI-powered smart environments so that all types of the *context factor*s could technically be registered and detected in the physical environment. While the registration is out of this paper's scope, prior works [18, 41] have addressed this need. Hence, for all *context*

*instance*s and UI icons that require in-situ visualization (e.g., an activity happening at a sofa), Fast-Forward Reality could display them at the corresponding physical locations. Meanwhile, *object state*s and *digital state*s are also available to display in the physical environment or anchored to a fixed place (e.g., coffee maker status, TV status, calendar events).

Moreover, we would discuss the possibility of deploying one CAP across different environments, which has been addressed by prior authoring systems [42, 72, 100]. Following the current *context factor* framework, one straightforward answer is that the CAP could be correctly run as long as all the involving *context instance*s are present in the new environment. Prior work [72] enables designers to validate the AR application performance by providing scenes where corresponding spatial and semantic associations are absent. Such an idea could be leveraged in the *test case* generation process by showcasing scenarios when each *context instance* is not available. However, the participants raised concerns when the researchers asked about more scenarios they would use the authored CAPs. *"Actually, I was thinking of using the music CAP when I went back to my parent's home as well. But I realized it would be a lot different. I may create a different CAP." (P6) "It depends on my preference. Some CAPs are just for home, I don't even want to turn it on somewhere else. But some CAPs, like more general ones, I would expect that to work all the time." (P12)* As CAPs are highly associated with humans' daily routines, any addition and deletion of *context instance*s represent the specific intention. Therefore, how to balance between scalability and personalization for authoring CAPs still requires further studies.

4.4.2 Mixed errors beyond CAP authoring. The main contributions of this work lie in the validation workflow and the algorithm to generate high-quality unit *test case*s that help identify and eliminate over/under specified errors. However, while researchers in the AI domain have been continuously working on improving the AI model performance and robustness, detection errors are inevitable under varied scenarios [47], and this issue has also been explored by other context-aware system research [50, 100]. In this paper, the user study has proven that Fast-Forward Reality could effectively avoid user-caused errors during CAP definition. It was conducted in a virtual environment and the accuracy of the CAPs was calculated by running the CAPs in the user-collected *context scene*s where all the *context instance*s were assumed to be correctly detected. Yet, we recognize the importance of bridging the gap between the validation process and the imperfect performance of the current AI technologies. Users would lose trust in our system if a CAP performs wrongly due to an object detection mistake even though the logic of the CAP is properly validated using our system. Intrinsically, the AI errors do not belong to the logic of any CAP. Thus, it is impractical to directly introduce such uncertainty into the validation workflow, as users could refine nothing logically on the CAP even though they know it may fail due to AI mistakes. However, we propose two potential methods to mitigate the mixed error issue for our system and future CAP authoring systems. First, sensing errors could happen during the *context history* collection, which reduces the correctness of the *test case* generation algorithm. We could enable users to eliminate wrong detection data so that the generated *test case*s are entirely reliable. Inspired by CAPturAR [92],

we could allow users to revisit the past activities and contexts with XR visualizations, then either delete or correct the wrong *context history*. Further, to deal with the sensing errors during execution, we notice that Explainable AI (XAI) has become a popular topic that allows for explaining AI errors to end-users to increase user trust in AI [5]. We envision an integration between our system and an XAI agent that can pop up the list of detected contexts when a CAP does not perform as expected to inform the user that such a mistake either comes from the CAP authoring or AI detection mistakes [4, 50].

Another uncertainty resides in the users' daily behaviors. A person never strictly follows a routine in complicated everyday life. Although our system improved the CAP accuracy via validation, the F-score of the CAPs could never reach 100%. One way to further improve the accuracy could be authoring multiple CAPs and adopting XAI approaches as discussed previously. Additionally, we envision the user wearing an advanced XR-HMD all the time for the detection of the contexts. Thus, the specific *context scene* that causes the mistake could be either recorded as a corner case and treated separately in the future or used for improving the AI-based models. We could also improve the *test case* generation algorithm by taking into consideration these *context scene*s that cause deployment errors.

4.4.3 Effectiveness of XR interfaces in CAP authoring and validation. As discussed in Related Works, 2D-based interfaces are prevalent in commercial CAP authoring tools and professional context-aware application development. With the advances of the XR technology, only a few works [92] explored the advantages of utilizing this emerging technology in CAP authoring tasks. In this paper, the positive feedback received on the innovative XR-based testing approach, along with XR's immersive capabilities, suggests that the symbiosis of XR and CAP remains a promising area for future research. On one hand, the spatial awareness of XR contributes to a more intuitive representation of spatially sensitive contexts such as *object state*s and *activities*, which empowers users to intuitively include contexts from the attaching physical objects or locations. Further, the immersive visualization of different *context instance*s enables users who do not have expertise in the concept of unit testing to rapidly understand imaginary contextual scenarios, and identify failures. This *feedforward* idea supported by XR successfully bridges the temporal gap between authoring and usage of a CAP, and enables the novel author-test-refine workflow introduced in this paper. We believe that the fusion of XR technology with CAP authoring not only enriches the process with greater intuitiveness and immersion but also facilitates understanding complicated real-world contextual combinations during authoring. This synergy, therefore, merits further research exploration, promising significant advancements in the field of CAP development. Additionally, unlike conventional 2D-based UIs which have undergone extensive refinement over the decades, the design of XR interfaces still needs large-scale and long-term studies to lower the user friction associated with this relatively new technology. In the next section, we outline various concerns that could guide further improvements of the XR interfaces in CAP authoring.

5 LIMITATION AND FUTURE WORK

More complicated contexts and CAPs. In this work, we primarily focus on the authoring of one single CAP following the framework discussed in Section 3.3. The quantitative results of the user study proved that with the additional validation stage, the users could successfully iterate the CAPs to make them more accurate.

One concern is that the current design of Fast-Forward Reality follows the mainstream CAP authoring tools that leverage nominal context factors (e.g., discrete labels output by Machine Learning modules). Yet, prior works such as CAPturAR [92] and DART [56] have adopted the programming by demonstration metaphor to enable the definition and detection of non-nominal contexts (e.g., a 'clip' of human activity, a time series of audio signals that represent people having group conversations). We envision integrating similar features in Fast-Forward Reality where users could directly demonstrate specific *context factor*s rather than selecting them one by one. However, how to generate test cases that help identify errors requires further research. For instance, how the system knows the semantics of a demonstrated 'clip' and finds the counter-examples from the *context history* requires either more sophisticated Machine Learning solutions or a systematic formative study to distill new design guidelines.

Our system supports the authoring of multiple CAPs targeting different smart functions while potential errors can also be eliminated via the validation (e.g., if a CAP controls the music player state, and a user is authoring another CAP for the TV, the *test case* generation algorithm would suggest the user considering the music player if these two object state *context factor*s are highly correlated. Yet, when the user creates multiple CAPs to control the same smart function (e.g., play different genres of music under different contexts), how to inform the user of the potential conflicts would be an issue, which has also been addressed by prior works [60, 81]. One straightforward solution is to directly include the other CAPs that control the same function into the *test case*s, and let the user to edit all the CAPs accordingly. Leveraging AI-based approaches such as associate rule mining [2] and decision trees to enable users to manage the priority among existing CAPs could be another solution.

Most of the users agreed that the current trigger-action metaphor and the provided types of the *context factor*s fulfilled their needs of authoring CAPs. Some users mentioned considering time-sensitive contexts such as 'I was having a meeting, then make a cup of coffee'. Since a user's *context history* is collected sequentially, it is feasible to calculate the uncertainty coefficients between any two *context factor*s in three temporal domains: past, present, and future. In this way, the system could enable users to author and validate time-sensitive CAPs. Meanwhile, by using the immersive authoring environment, prior arts [13, 92] visualize a user's past activities with XR animations. Instead of showing static XR icons, we could further show dynamic *test case*s indicating time elapses.

Different levels of immersiveness. While being situated in the XR authoring environment, the users welcomed the capability to test each case by acting it out immersively. One user mentioned that: *"Currently, everything is inside one home environment, I was considering some contexts like 'go-to-office' or 'while-driving'." (P11)* In addition, P2 raised an interesting feature that *"I was wondering if I could see more than one test case at the same time. Maybe showing me some miniature layouts."* Considering the diversity of the available contexts and the different user backgrounds, providing different levels of immersiveness to visit *test case*s and author CAPs would be necessary. For instance, prior works have shown the capability to immerse users into different virtual scenes [72, 99], or even showing different room layouts at one time [67] using conventional non-XR UIs. While the current system could be adapted to desktop/mobile devices (e.g., visualize CAPs with 2D UIs, visualize multiple test cases with duplicated room layouts, and use an on-screen controller to move the virtual camera within the environment during testing), further studies are required to investigate whether the benefits of such adaptation would compensate for the reduced effectiveness of the *feedforward* actions for understanding the *test case*s. Especially, for novice users who are not familiar with the ideas of under/over-specified CAPs and unit tests, we need more studies to evaluate whether the conventional-UI-based design would introduce additional mental loads.

Lower the friction of using the system. We received complimentary feedback on the immersiveness enabled by the in-situ placed XR contents and the intuitive system operations. Some users (P1 and P3) mentioned visualization issues when some XR contents were overlaid and clustered from some specific view directions. Meanwhile, we envision more smart objects and functions would be available in future smart home environments. To reduce users' visual loads, adaptively displaying the 3D contents when users move closer to them [44] or pay attention to different contents [51, 70], and adding a filter function to solely show contexts of a specific type of object would be a future improvement of the system. Furthermore, *test case*s generated by the system received positive feedback during the user study. Enabling users to act out each *test case* facilitates them to understand whether the CAP needs to be modified. With these highlighted features being kept, we could add pre-processed automation to further reduce users' workloads. Ranking *test case*s according to the uncertainty coefficients would be one improvement according to P9's suggestion. Users could pay more attention to those scenarios that would more likely happen in real life, and rapidly walk through the *test case*s that are listed at the end. We could also design a more advanced criterion that measures the importance of the generated *test case*s (e.g., if a user edits the CAP based on a *test case*, to what extent could the current CAP perform more accurately).

6 CONCLUSION

We presented Fast-Forward Reality, a novel workflow that supports an end-user to validate the CAPs with diverse *test case*s via *feedforward* simulation in XR. We first identified that existing authoring processes can result in *under-specified* or *over-specified* CAPs that cause unexpected behavior, leading to annoyance and frustration. In order to address this issue, we proposed an 'author-test-refine' workflow by leveraging the unit test approach in the software programming area. In specific, using the pervasively collected everyday context record and adopting the frameworks proposed by prior context-aware application validation works, we designed a computational approach to generate multiple unit *test case*s that not only are tailored to the user's personal routines but also help reveal

mistakes in the authored CAPs. Then, while being immersed in an XR-based authoring environment, users can experience *test case*s through in-situ visualization that conveys *feedforward* contextual scenarios, enabling intuitive validation of CAPs. A user study was conducted to evaluate the effectiveness of our *test case* generation algorithm and the design and functionality of the XR authoring environment. The high accuracy of user-authored CAPs and positive feedback on system features validated the overall performance and usability of the system. As an increasing amount of complex contexts can be digitalized into the digital space, we believe that our work can inform and provide inspiration for future investigation on how authoring systems can assist non-expert users to create error-free intelligent automation and policies that enhance the quality of life and work.

REFERENCES

[1] 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.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 207–216.

[3] Alexa Routines 2022. Alexa Routines. https://www.amazon.com/alexa-routines/.

[4] Stavros Antifakos, Nicky Kern, Bernt Schiele, and Adrian Schwaninger. 2005. Towards improving trust in context-aware systems by displaying system confidence. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*. 9–14.

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* 58 (2020), 82–115.

[6] Dana H Ballard, Mary M Hayhoe, Polly K Pook, and Rajesh PN Rao. 1997. Deictic codes for the embodiment of cognition. *Behavioral and brain sciences* 20, 4 (1997), 723–742.

[7] Adrian Bangerter. 2004. Using pointing and describing to achieve joint focus of attention in dialogue. *Psychological science* 15, 6 (2004), 415–419.

[8] Victoria Bellotti and Keith Edwards. 2001. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human–Computer Interaction* 16, 2-4 (2001), 193–212. https://doi.org/10.1207/S15327051HCI16234_05

[9] Sanjay Bhati, Harshid Soni, Vijayrajsinh Zala, Parth Vyas, and Yash Sharma. 2017. Smart medicine reminder box. *IJSTE-International Journal of Science Technology & Engineering* 3, 10 (2017), 172–177.

[10] Valentina Bianchi, Marco Bassoli, Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, and Ilaria De Munari. 2019. IoT wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet of Things Journal* 6, 5 (2019), 8553–8562.

[11] Michael S Borofsky, Casey A Dauw, Nadya York, Colin Terry, and James E Lingeman. 2018. Accuracy of daily fluid intake measurements using a "smart" water bottle. *Urolithiasis* 46, 4 (2018), 343–348.

[12] Yuanzhi Cao, Xun Qian, Tianyi Wang, Rachel Lee, Ke Huo, and Karthik Ramani. 2020. An exploratory study of augmented reality presence for tutoring machine tasks. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.

[13] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 521–534.

[14] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7291–7299.

[15] Alberto Huertas Celdrán, Félix J García Clemente, Manuel Gil Pérez, and Gregorio Martínez Pérez. 2014. SeCoMan: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications. *IEEE Systems Journal* 10, 3 (2014), 1111–1124.

[16] Seungho Chae, Yoonsik Yang, Heeseung Choi, Ig-Jae Kim, Junghyun Byun, Jiyoon Jo, and Tack-Don Han. 2016. Smart advisor: Real-time information provider with mobile augmented reality. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 97–98.

[17] Yi-Ting Chiang, Kuo-Chung Hsu, Ching-Hu Lu, Li-Chen Fu, and Jane Yung-Jen Hsu. 2010. Interaction models for multiple-resident activity recognition in a smart home. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3753–3758.

[18] Meghan Clark, Mark W Newman, and Prabal Dutta. 2022. ARticulate: One-Shot Interactions with Intelligent Assistants in Unfamiliar Smart Spaces Using Augmented Reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 1–24.

[19] Michael H Coen et al. 1998. Design principles for intelligent environments. *AAAI/IAAI* 547 (1998), 554.

[20] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. 2018. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 720–736.

[21] Ismayle de Sousa Santos, Rossana Maria de Castro Andrade, Lincoln Souza Rocha, Santiago Matalonga, Kathia Marcal de Oliveira, and Guilherme Horta Travassos. 2017. Test case design for context-aware applications: Are we there yet? *Information and Software Technology* 88 (2017), 1–16.

[22] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. 2000. The semantic web: The roles of XML and RDF. *IEEE Internet computing* 4, 5 (2000), 63–73.

[23] Anind K Dey. 2001. Understanding and using context. *Personal and ubiquitous computing* 5, 1 (2001), 4–7.

[24] Anind K Dey, Gregory D Abowd, and Daniel Salber. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction* 16, 2-4 (2001), 97–166.

[25] 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.

[26] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *International conference on pervasive computing*. Springer, 254–271.

[27] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. 2002. But How, Donald, Tell Us How? On the Creation of Meaning in Interaction Design through Feedforward and Inherent Feedback. In *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (London, England) *(DIS '02)*. Association for Computing Machinery, New York, NY, USA, 285–291. https://doi.org/10.1145/778712.778752

[28] Yegang Du, Yuto Lim, and Yasuo Tan. 2019. A novel human activity recognition and prediction in smart home based on interaction. *Sensors* 19, 20 (2019), 4474.

[29] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. 2017. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In *Proceedings of the 43rd Graphics Interface Conference*. 156–162.

[30] Andreas Rene Fender and Christian Holz. 2022. Causality-preserving Asynchronous Reality. In *CHI Conference on Human Factors in Computing Systems*. 1–15.

[31] Aurel-Dorian Floarea and Valentin Sgârciu. 2016. Smart refrigerator: A next generation refrigerator connected to the IoT. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, 1–6.

[32] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. 2022. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18995–19012.

[33] Shalom Greene, Himanshu Thapliyal, and David Carpenter. 2016. IoT-based fall detection for smart home environments. In *2016 IEEE international symposium on nanoelectronic and information systems (iNIS)*. IEEE, 23–28.

[34] Tobias Griebe and Volker Gruhn. 2014. A model-based approach to test automation for context-aware mobile applications. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. 420–427.

[35] Jens Grubert, Tobias Langlotz, Stefanie Zollmann, and Holger Regenbrecht. 2016. Towards pervasive augmented reality: Context-awareness in augmented reality. *IEEE transactions on visualization and computer graphics* 23, 6 (2016), 1706–1724.

[36] Corentin Haidon, Hélène Pigot, and Sylvain Giroux. 2020. Joining semantic and augmented reality to design smart homes for assistance. *Journal of Rehabilitation and Assistive Technologies Engineering* 7 (2020), 2055668320964121.

[37] Paul Hamill. 2004. *Unit test frameworks: tools for high-quality software development*. " O'Reilly Media, Inc.".

[38] Fengming He, Xiyun Hu, Jingyu Shi, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. Ubi Edge: Authoring Edge-Based Opportunistic Tangible User Interfaces in Augmented Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–14.

[39] Sumi Helal, Bryon Winkler, Choonhwa Lee, Youssef Kaddoura, Lisa Ran, Carlos Giraldo, Sree Kuchibhotla, and William Mann. 2003. Enabling location-aware pervasive computing applications for the elderly. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications,*

2003.(PerCom 2003). IEEE, 531–536.

[40] Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J Quinn. 2021. Adaptutar: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[41] Ke Huo, Yuanzhi Cao, Sang Ho Yoon, Zhuangying Xu, Guiming Chen, and Karthik Ramani. 2018. Scenariot: Spatially mapping smart things within augmented reality scenes. In *Proceedings of the 2018 CHI Conference on human factors in computing systems*. 1–13.

[42] IFTTT 2022. IFTTT. https://ifttt.com.

[43] Panagiotis Kostopoulos, Athanasios I Kyritsis, Michel Deriaz, and Dimitri Konstantas. 2016. F2D: a location aware fall detection system tested with real data from daily life of elderly people. *Procedia computer science* 98 (2016), 212–219.

[44] Wallace S Lages and Doug A Bowman. 2019. Walking with adaptive augmented reality workspaces: design and usage patterns. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 356–366.

[45] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. 2004. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 172–181.

[46] 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.

[47] Deyi Li and Yi Du. 2017. *Artificial intelligence with uncertainty*. CRC press.

[48] Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information systems frontiers* 17, 2 (2015), 243–259.

[49] Yang Li, Jason I Hong, and James A Landay. 2004. Topiary: a tool for prototyping location-enhanced applications. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 217–226.

[50] Brian Y Lim and Anind K Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*. 13–22.

[51] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 147–160.

[52] David Lindlbauer and Andy D Wilson. 2018. Remixed reality: Manipulating space and time in augmented reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[53] Ziyi Liu, Zhengzhe Zhu, Enze Jiang, Feichi Huang, Ana M Villanueva, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[54] Chu Luo, Jorge Goncalves, Eduardo Velloso, and Vassilis Kostakos. 2020. A survey of context simulation for testing mobile context-aware applications. *ACM Computing Surveys (CSUR)* 53, 1 (2020), 1–39.

[55] Chu Luo, Miikka Kuutila, Simon Klakegg, Denzil Ferreira, Huber Flores, Jorge Goncalves, Mika Mäntylä, and Vassilis Kostakos. 2017. TestAWARE: a laboratory-oriented testing tool for mobile context-aware applications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–29.

[56] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 197–206.

[57] Luca Mainetti, Luigi Patrono, Andrea Secco, and Ilaria Sergi. 2016. An IoT-aware AAL system for elderly people. In *2016 International multidisciplinary conference on computer and energy science (SpliTech)*. IEEE, 1–6.

[58] Santiago Matalonga, Felyppe Rodrigues, and Guilherme Travassos. 2015. Challenges in testing context aware software systems. In *9th Workshop on Systematic and Automated Software Testing*. sn, 51–60.

[59] Meta Quest 2 2022. Meta Quest 2. https://www.oculus.com/experiences/quest/.

[60] Fereshteh Jadidi Miandashti, Mohammad Izadi, Ali Asghar Nazari Shirehjini, and Shervin Shirmohammadi. 2020. An empirical approach to modeling user-system interaction conflicts in smart homes. *IEEE Transactions on Human-Machine Systems* 50, 6 (2020), 573–583.

[61] Microsoft HoloLens 2 2022. Microsoft HoloLens 2. https://www.microsoft.com/en-us/hololens.

[62] Diaa Salama Abdul Minaam and Mohamed Abd-ELfattah. 2018. Smart drugs: Improving healthcare using smart pill box for medicine reminder and monitoring system. *Future Computing and Informatics Journal* 3, 2 (2018), 443–456.

[63] Aamir Mehmood Mirza and Muhammad Naeem Ahmed Khan. 2018. An automated functional testing framework for context-aware applications. *IEEE Access* 6 (2018), 46568–46583.

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

[65] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. Xrdirector: A role-based collaborative immersive authoring system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[66] Michael Nebeling, Shwetha Rajaram, Liwei Wu, Yifei Cheng, and Jaylin Herskovitz. 2021. Xrstudio: A virtual production and live streaming system for immersive instructional experiences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[67] Roy Oberhauser. 2022. VR-Git: Git Repository Visualization and Immersion in Virtual Reality. In *Proceedings of the the Seventeenth International Conference on Software Engineering Advances*. 9–14.

[68] Ian Oppermann, Matti Hämäläinen, and Jari Iinatti. 2004. *UWB: theory and applications*. John Wiley & Sons.

[69] JoonSeok Park, Mikyeong Moon, Seongjin Hwang, and Keunhyuk Yeom. 2007. CASS: A context-aware simulation system for smart home. In *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*. IEEE, 461–467.

[70] Ken Pfeuffer, Yasmeen Abdrabou, Augusto Esteves, Radiah Rivu, Yomna Abdelrahman, Stefanie Meitner, Amr Saadi, and Florian Alt. 2021. ARtention: A design space for gaze-adaptive user interfaces in augmented reality. *Computers & Graphics* 95 (2021), 1–12.

[71] pytest 2023. pytest. https://docs.pytest.org/en/8.0.x/.

[72] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, Ananya Ipsita, and Karthik Ramani. 2022. ScalAR: Authoring Semantically Adaptive Augmented Reality Experiences in Virtual Reality. In *CHI Conference on Human Factors in Computing Systems*. 1–18.

[73] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, and Karthik Ramani. 2022. ARnnotate: An Augmented Reality Interface for Collecting Custom Dataset of 3D Hand-Object Interaction Pose Estimation. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[74] Raf Ramakers, Kashyap Todi, and Kris Luyten. 2015. PaperPulse: An Integrated Approach to Fabricating Interactive Paper. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 267–270. https://doi.org/10.1145/2702613.2725430

[75] Abhishek Roy, SK Das Bhaumik, Amiya Bhattacharya, Kalyan Basu, Diane J Cook, and Sajal K Das. 2003. Location aware resource management in smart homes. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003)*. IEEE, 481–488.

[76] Michele Sama, David S Rosenblum, Zhimin Wang, and Sebastian Elbaum. 2008. Model-based fault detection in context-aware adaptive applications. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. 261–271.

[77] Bill Schilit, Norman Adams, and Roy Want. 1994. Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications*. IEEE, 85–90.

[78] Shortcuts 2022. Shortcuts. https://support.apple.com/guide/shortcuts/create-a-new-personal-automation-apdfbdbd7123/5.0/ios/15.0.

[79] Henri Theil. 1970. On the estimation of relationships involving qualitative variables. *Amer. J. Sociology* 76, 1 (1970), 103–154.

[80] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 543–555. https://doi.org/10.1145/2901790.2901817

[81] Rahmadi Trimananda, Seyed Amir Hossein Aqajari, Jason Chuang, Brian Demsky, Guoqing Harry Xu, and Shan Lu. 2020. Understanding and automatically detecting conflicting interactions between smart home IoT applications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1215–1227.

[82] Unity3D 2022. Unity3D. https://unity.com/.

[83] 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.

[84] 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.

[85] Jo Vermeulen, Kris Luyten, and Karin Coninx. 2012. Understanding complex environments with the feedforward torch. In *International Joint Conference on Ambient Intelligence*. Springer, 312–319.

[86] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 2013. Crossing the bridge over Norman's Gulf of Execution: revealing feedforward's true identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1931–1940.

[87] Huai Wang, WK Chan, and TH Tse. 2014. Improving the effectiveness of testing pervasive software via context diversity. *ACM Transactions on Autonomous and*

Adaptive Systems (TAAS) 9, 2 (2014), 1–28.

[88] Huai Wang, Ke Zhai, and TH Tse. 2010. Correlating context-awareness and mutation analysis for pervasive computing systems. In *2010 10th International Conference on Quality Software*. IEEE, 151–160.

[89] Junbo Wang, Zixue Cheng, Lei Jing, Yota Ozawa, and Yinghui Zhou. 2012. A location-aware lifestyle improvement system to save energy in smart home. In *4th International Conference on Awareness Science and Technology*. IEEE, 109–114.

[90] Liang Wang, Tao Gu, Xianping Tao, Hanhua Chen, and Jian Lu. 2011. Recognizing multi-user activities using wearable sensors in a smart home. *Pervasive and Mobile Computing* 7, 3 (2011), 287–298.

[91] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 552–567.

[92] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 328–341.

[93] Zhimin Wang, Sebastian Elbaum, and David S Rosenblum. 2007. Automated generation of context-aware tests. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 406–415.

[94] Zeyu Wang, Cuong Nguyen, Paul Asente, and Julie Dorsey. 2021. Distanciar: Authoring site-specific augmented reality experiences for remote environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[95] Ron Weinstein. 2005. RFID: a technical overview and its application to the enterprise. *IT professional* 7, 3 (2005), 27–33.

[96] Mark Weiser. 1999. The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review* 3, 3 (1999), 3–11.

[97] Konlakorn Wongpatikaseree, Mitsuru Ikeda, Marut Buranarach, Thepchai Supnithi, Azman Osman Lim, and Yasuo Tan. 2012. Activity recognition using context-aware infrastructure ontology in smart home domain. In *2012 Seventh International Conference on Knowledge, Information and Creativity Support Systems*. IEEE, 50–57.

[98] Chao-Lin Wu, Yi-Show Tseng, and Li-Chen Fu. 2013. Spatio-temporal feature enhanced semi-supervised adaptation for activity recognition in IoT-based context-aware smart homes. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, 460–467.

[99] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Spacetime: Enabling fluid individual and collaborative editing in virtual reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 853–866.

[100] Wenhua Yang, Chang Xu, Yepang Liu, Chun Cao, Xiaoxing Ma, and Jian Lu. 2014. Verifying self-adaptive applications suffering uncertainty. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. 199–210.

[101] Hui Ye and Hongbo Fu. 2022. ProGesAR: Mobile AR Prototyping for Proxemic and Gestural Interactions with Real-world IoT Enhanced Spaces. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[102] Hui Ye, Jiaye Leng, Chufeng Xiao, Lili Wang, and Hongbo Fu. 2023. ProObjAR: Prototyping Spatially-aware Interactions of Smart Objects with AR-HMD. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.

[103] Lian Yu, Wei-Tek Tsai, and Gian Perrone. 2016. Testing context-aware applications based on bigraphical modeling. *IEEE Transactions on Reliability* 65, 3 (2016), 1584–1611.

[104] Daqing Zhang, Tao Gu, and Xiaohang Wang. 2005. Enabling context-aware smart home with semantic web technologies. *International Journal of Human-friendly Welfare Robotic Systems* 6, 4 (2005), 12–20.

[105] Lei Zhang and Steve Oney. 2020. Flowmatic: An immersive authoring tool for creating interactive scenes in virtual reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 342–353.

[106] Yang Zhang, Yasha Iravantchi, Haojian Jin, Swarun Kumar, and Chris Harrison. 2019. Sozu: Self-powered radio tags for building-scale activity sensing. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 973–985.

[107] Hong Zhu, Patrick AV Hall, and John HR May. 1997. Software unit test coverage and adequacy. *Acm computing surveys (csur)* 29, 4 (1997), 366–427.

[108] Zhengzhe Zhu, Ziyi Liu, Youyou Zhang, Lijun Zhu, Joey Huang, Ana M Villanueva, Xun Qian, Kylie Peppler, and Karthik Ramani. 2023. LearnIoTVR: An End-to-End Virtual Reality Environment Providing Authentic Learning Experiences for Internet of Things. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.