TYPE Original Research
PUBLISHED 14 August 2025
DOI 10.3389/friot.2025.1623733

OPEN ACCESS

EDITED BY
Navod Neranjan Thilakarathne,
University of Colombo, Sri Lanka

REVIEWED BY
Zaharaddeen Karami Lawal,
Universiti Brunei Darussalam, Brunei
Nethmini T. Weerawarna,
University of Colombo, Sri Lanka

*CORRESPONDENCE
Börge Kordts,
b.kordts@uni-luebeck.de

RECEIVED 06 May 2025
ACCEPTED 18 July 2025
PUBLISHED 14 August 2025

CITATION
Kordts B, Brandl LC and Schrader A (2025) Managing self-explaining ambient applications. *Front. Internet Things* 4:1623733. doi: 10.3389/friot.2025.1623733

Managing self-explaining ambient applications

Börge Kordts*, Lea C. Brandl and Andreas Schrader

Institute of Telematics, University of Lübeck, Lübeck, Germany

Introduction: The Internet of Things (IoT) plays a crucial role in realizing the vision of pervasivemcomputing. The variety of devices and services in the IoT often makes manual integration complex and tedious. To mitigate this challenge, self-organization has emerged as a promising approach. Furthermore, the concept of selfexplainability has been proposed to clarify user interactions with dynamically interconnected smart objects.

Methods: This paper explores related research in the domains of (a) service composition, (b) self-explainability, and (c) application management within smart environments. To address existing limitations, we propose a novel application management concept. Therefore, we extended a framework that utilizes self-explaining applications to generate user instructions dynamically that explain the control of these. A user study was conducted to assess the effectiveness and preferred options of modalities of these instructions.

Results: The findings indicate that the approach successfully tackles the research question concerning the integration of self-explaining applications into smart environments.

KEYWORDS
smart object guidance, self-reflection, self-organization, ambient applications, application management

1 Introduction

Advancements in technology over the past decades allowed for the vision of *pervasive computing* to continually pave its way into the reality of people's lives. Consequently, people's everyday tasks and actions are becoming increasingly enriched through access to digital information that is invisibly surrounding users, but is available when needed.

The Internet of Things (IoT) has the potential to realize this vision of pervasive computing (Ornes, 2016). By connecting multiple smart objects that are placed inside a pervasive environment, device ensembles can be formed to extend the abilities of their users and support them in their everyday lives. These device ensembles can be equipped with sensors to monitor the environment's current state, actuators to manipulate the environment, and interfaces for user interactions. They may be connected via wireless networks within the Internet of Things. This makes them capable of assisting users with their tasks and needs.

In this regard, Altakrouri (2014) proposes decoupling input and output capabilities of smart devices to form dynamic solutions tailored to specific situations, users, as well as their abilities and preferences. Generally, the author also suggests decoupling interaction to honor the high level of user mobility, diversity, different contexts, and changing resources.

However, it may be challenging to connect components of smart environments (such as multiple IoT devices, services, and interfaces) manually, due to the potential complexity of such systems and the variety of the technology involved. Self-Organization but also centralized approaches for an automatic composition have been researched to provide particularly more comprehensive services suitable for the user's tasks and goals (see Section 3).

This directly addresses the need to flexibly provide interaction components, to enrich the control of output components, and to deploy new applications within smart environments. Typically, all these actions should be performed based on certain constraints. For instance, the use of a certain camera sensor or a display connected to a device could be required. These components could be modeled logically as smart objects that provide sensor data or actuators. Some device types bundle these components so that they are more practically accessible as a whole. Tablets or smart phones, for instance, can make use of a specific set of sensors and actuators, including a camera, an orientation sensor, and a touch-sensitive display.

In doing so, we are able to create ambient interaction spaces that offer various interaction options to users to control the surrounding and applications running within these. Though, the dynamic nature of these systems may obfuscate not only the interaction between users and respective systems. More fundamentally, this may have a negative impact on their understanding of the whole system and its functionality. Ultimately, this could result in a decreased usability of such interactive systems.

To counteract this issue, self-explainability has been proposed. Some solutions are having a particular focus on explaining the interaction between a user and dynamically connected smart objects. The concept of self-explainability has been researched in the context of connected smart objects, services and applications in the IoT (see Section 3).

This goes hand in hand with the previous argument that the app concept is in general applicable not only to single devices but also to entire smart environments. Here, apps, while running on some computer device connected to the network, are accessible from the environment and are therefore perceived as if they were running in the environment (Kubitza, 2017). We refer to applications that make use of smart objects in some way and that run within smart environments as *Ambient Applications* (see Section 2). This term contrasts with the common usage of 'apps', which usually have no close relationship to the physical environment and generally focus on graphical output on a rectangular screen.

However, it has not been stated how smart environments are equipped with these self-explaining applications. In particular, the term application suggests the existence of an application store and associated installation mechanism for deployment. Generally, the metaphor of the app store has been argued to be an attractive idea in the context of IoT (Stastny et al., 2015). All in all, solutions are required that can easily provide connected components to meet the wide range of needs but also explanations as the dynamic nature of these systems may obstruct the interaction. Yet, current approaches and existing frameworks towards this end either lack comprehensive user instructions, do not cover the details of the required interaction, or need manual setup (see Section 3).

Hence, in this paper, we propose a concept to allow smart environments to manage self-explaining applications for these smart environments based on the hardware present in the respective surroundings. We also present an implementation of this concept based on the *Ambient Reflection* framework. The framework is capable of connecting smart objects and applications based upon their self-description and further providing instructions or interactive tutorials for these systems that may guide users.

Our main contributions presented in this paper are:

- A literature overview in the domains of service composition, self-explainability, and application management,
- A definition for the term Ambient Application,
- A novel application management concept extending the Ambient Reflection framework allowing to generate user instructions dynamically,
- A user study that includes participant observation, two questionnaires, and semi-structured interviews to assess the effectiveness and preferred options of modalities of these instructions.

These contributions address the following research questions:

1. How can packaged Ambient Applications be made available and how can they be brought into the environment?
2. To what extent are generated instructions for dynamically connected ensembles based upon the components' self-descriptions suitable to instruct users about interaction possibilities provided by the environment?

2 Ambient applications

While smart objects are always physical objects and consist of a hardware component, most smart environments make use of some sort of separate software components, e.g., services. Beyond that, mixed forms and hybrid devices exist that offer complex functionality, where some devices even offer to install additional software (sometimes referred to as skills or features). However, the majority of smart objects available on the market provide relatively simple input or output functionality which can be used to implement control systems without complex logic, like lighting or heating control, for instance.

Moreover, smart environments are typically equipped with software components that are neither directly bound to physical devices nor run on smart objects directly. Instead, they run on computing hardware provided elsewhere in order to realize more complex application scenarios by means of intermediate logic. In the literature, this intermediate logic is sometimes referred to as a service (cf. Goumopoulos and Mavrommati, 2020). However, when referring to services, the focus is not on the interaction of a user with a system of smart objects and this intermediary logic, but is generally aimed at sensor-based control of output devices typically based on different rule sets. This logic is therefore executed in the background, as the term service already suggests.

However, when combining software logic and smart objects with input and output functionality in the context of human-computer interaction, the term *application* comes to mind. A term that is linked to classic computer systems, in particular desktop systems. It directly relates to the concept of a user application as an application that responds to input from a user. If necessary, it forwards output elsewhere. Hence, we transfer the concept to the domain of smart environments.

Ambient applications have previously been briefly described as software components that interact with smart objects within smart environments (Kordts et al., 2022). Similarly, Kubitza (2017) describes applications that run in the environment without explicitly coining a term.

Here, we define the term as follows: *ambient applications* are characterized by their close relationship to a smart environment, whereby they are either controlled by components of the environment or have output options for controlling other ambient applications or smart objects. These are pure software components that have to be executed on some computing system, but the hardware is in principle interchangeable.

This perspective offers the general possibility of decoupling the interaction and the application or business logic. If the interfaces are designed accordingly, applications as well as input and output devices can be separated and, in principle, combined in different ways for the respective usage context. This division of components can take place on both a physical and a logical level.

Ambient Applications may offer interfaces with various imaginable modalities. These can be graphical in nature, similar to traditional user interfaces, but can also utilize other approaches. For example, non-rectangular projections on objects in the environment or interfaces based on audio, such as voice user interfaces, are conceivable.

Ambient applications can therefore be differentiated from common 'apps' on the basis of these aspects. They have a relationship to the physical environment and run in smart environments or are perceived as such. In addition, common 'apps' generally focus on a graphical user interface with rectangular screens, whereas ambient applications are also conceivable with other interaction modalities.

3 Related work

Although previous research regarding composition, explainability, or application deployment has been presented, the conjunction of these topics entails its own challenges. In the following, we present related work regarding each of these three aspects in order to finally discuss research gaps within each of the respective fields as well as in the overlapping areas.

3.1 Service composition in smart environments

The aim of software composition is to reuse software modules in order to reduce the complexity of designs and simplify the development of software systems. Though, in the context of smart environments, the actually available components and devices are often only briefly known before they are actually used. Therefore, predictions about actually available hardware and services are challenging. Hence, different approaches are necessary than in other fields.

Bellavista et al. (2018) describe a service composition middleware for smart environments that relies on a translucent composition model. The approach is based on customizable rule sets and algorithms to solve a constraint satisfaction problem used to model the composition. Towards this end, a brute-force approach or backtracking with a heuristic (realized by filter criteria) can be used. This approach stands in need of users to specify requirements. However, no deeper understanding of the system and technical details is required.

Moreover, opportunistic software composition based on machine learning methods has been proposed. For instance, Koussaifi et al. (2018) describe a solution for an opportunistic and automatic service composition using reinforcement learning. Their approach is based on an assembly engine that discovers existing components and decides on the connections without using a pre-established plan. Users are involved in the adaptation process in order to resolve conflict situations and provide feedback.

Delcourt et al. (2021) present a solution for an automatic and dynamic software composition allowing for an adaptation to the current situation and the user. The composition is based on reinforcement learning without explicit user needs or predefined assembly plans. Users are given the opportunity to intervene in the process by accepting, rejecting, or changing proposed solutions prior to deployment.

Recently, Delcourt et al. (2024) present insights into approaches to involve users in the machine learning process to build self-adapting smart environments. This includes guidelines for the design of such systems. Notably, making clear what the system can do, how well it performs and how it works are crucial aspects. Hence, explanations of the system play an important role. Moreover, the authors integrate these approaches into their case of opportunistic software composition (see above).

All described systems address the more general problem of software composition in the context of smart environments. Although aiming at the involvement of users into the software composition process, none of the presented approaches explicitly focuses on the user interaction with the resulting systems. In other words, there is no special focus on explaining the human-computer interaction itself, particularly in the sense of involved interaction techniques. Moreover, there is no emphasis on decoupling input and output. Though, it has been previously argued that a clearer focus on the interaction may be beneficial with respect to the composition, the specification process, or the understandability (Kordts et al., 2022).

3.2 Self-explainability in smart environments

There is only limited work presented in the literature that addresses self-explainability in the context of pervasive computing. Research in that respect primarily focuses on reasoning about the situation and causes within the self-adaptation process that led to this state.

For instance, Garcia Dominguez et al. (2019) as well as Parra-Ullauri et al. (2020) interpret self-explainability as the ability of a system to answer questions about decisions made in the past. The authors present an architecture for recording temporal data. This data can subsequently be used to explain and reason why a system exhibits its current behavior and which adaptations have been made. Ultimately, history-awareness makes it possible to analyze the impact that past history has on the respective decision process and potentially draw conclusions for future decisions.

Moreover, self-explainability has been proposed to help users understand the logic of control systems, like in smart environments (Fadiga et al., 2021). Thereby, users can be put in the position to better react when needed. The authors present a way to derive causal models from experiments on the environment and observational data. These models may be used to explain an adaptive system's behavior.

There are also approaches that tackle explaining the behavior of composed software modules and services for pervasive environments. Koussaifi et al. (2019) present an architecture that can be used to generate descriptions of composed systems for smart environments. For this purpose, generated user-oriented descriptions mainly consist of rules that explain the components and the applications. They are based on descriptions of and connections between the respective components. Beforehand, developers need to specify descriptions, while the connections are determined by the service composition.

More closely related, Fey and Drechsler (2020) present a conceptual framework for self-explainability using a model that is based on cause-effect chains. These chains are a result of transitively combined cause-effect relationships. Moreover, the authors propose a technical solution that allows to infer explanations on the functional level. Finally, the authors describe a case study of a robot controller that provides explanations. Since this approach requires additional code, the authors further discuss the trade-off between the degree of completeness in explanations and implementation cost.

Additionally, Fey et al. (2022) propose a conceptual framework and a generic explanation pattern that can be used to provide self-adaptive systems with self-explainability. The work is based on finite-state automata as a means of formalization that can be used to determine whether an involved entity is in need of an explanation. These explanations may in turn be used to determine further actions.

However, it has been argued that current approaches for explanations are insufficient for most users and that there is a need for user-centric intelligible explanations. Towards this end, Sadeghi et al. (2024) present a framework for generating such explanations for smart environments. Their approach is based on algorithmic explanation constructs covering facts or events that describe the logic behind the behavior of a system. In addition, they include contextual explanation constructs for facts or specifications that provide information related to this behavior. Cause-effect paths are then used to reason about the causes for the current state of the system. Finally, user-centric explanations are generated from these by determining the best-suited granularity of explanations for the individual user.

Moreover, King (2024) introduces a framework for smart objects capable of self-knowledge based on methods from generative artificial intelligence. A formal model of smart objects' states forms the basis of a synthesis of actions and explanations that yield and describe these states. Towards this end, a teacher large language model trains small language models suitable for edge devices.

Notably, most authors focus on explaining the behavior of software systems in smart environments to users with a particular focus on reasoning about the actions that led to the current state. Interactions between a user and the system, including explicitly intended control actions and the system's response, play only a subordinate role. Particularly, interacting with dynamically connected ensembles is scarcely addressed.

Recently, a system that provides self-explainability for adaptive systems in smart environments also including Ambient Applications has been described (Kordts et al., 2022). This work, in turn, is based on research conducted by Burmeister (2018). The authors focus on explaining involved components primarily from the perspective of the interaction while decoupling input and output.

In order to address self-organization and self-explainability for dynamic ensembles of smart objects and Ambient Applications, the authors present the *Ambient Reflection* framework. This framework consists of three core components. The Smart Object Library can be used to integrate smart objects or Ambient Applications into the framework directly. Software components that cannot make use of the library directly, like web applications, can use the Virtual Device Daemon (VDD) and the corresponding Virtual Device API. These components provide a bridge based on a web socket connection in order to integrate respective applications into the framework.

The third component of the framework is the Description Mediator which is used to connect smart objects and Ambient Applications based on their functionality. For this purpose, the Description Mediator provides a probabilistic brute-force algorithm as a best-effort approach to the general ensembling problem. It further provides descriptions of the thereby created ensembles and informs all involved components about both, the connections and the combined descriptions. See Figure 1 for an overview of the framework and its components.

The self-explainability of the framework is based on self-descriptions of the involved components that are based on a human- and machine-readable description language for smart objects and Ambient Applications. Towards this end, descriptions of applications cover the interfaces with respect to input and output and basic system behavior. This means that applications are modeled as black boxes without providing the entire internal system logic.

Notably, with reference to the definition of the term Ambient Application presented in Section 2, in addition to the relationship to the physical environment, the applications integrated in the framework are self-explaining and discoverable in the network. They further provide services for interaction-related invocations and miscellaneous messages.

However, the authors have not stated how smart environments can be equipped with self-explaining applications. Thus, the concept of managing applications with respect to the more general metaphor of an application store has not been particularly targeted. Yet, all these works provide a suitable foundation for the integration of self-explaining applications into smart environments.

3.3 App management for smart environments

Stastny et al. (2015) present three main challenges in adapting the app store metaphor to IoT scenarios based on a literature review, interviews, and a survey. Namely, (a) the difficulty of supporting the diversity in the software and hardware vendor market, (b) the tension between context awareness and the need for pre-

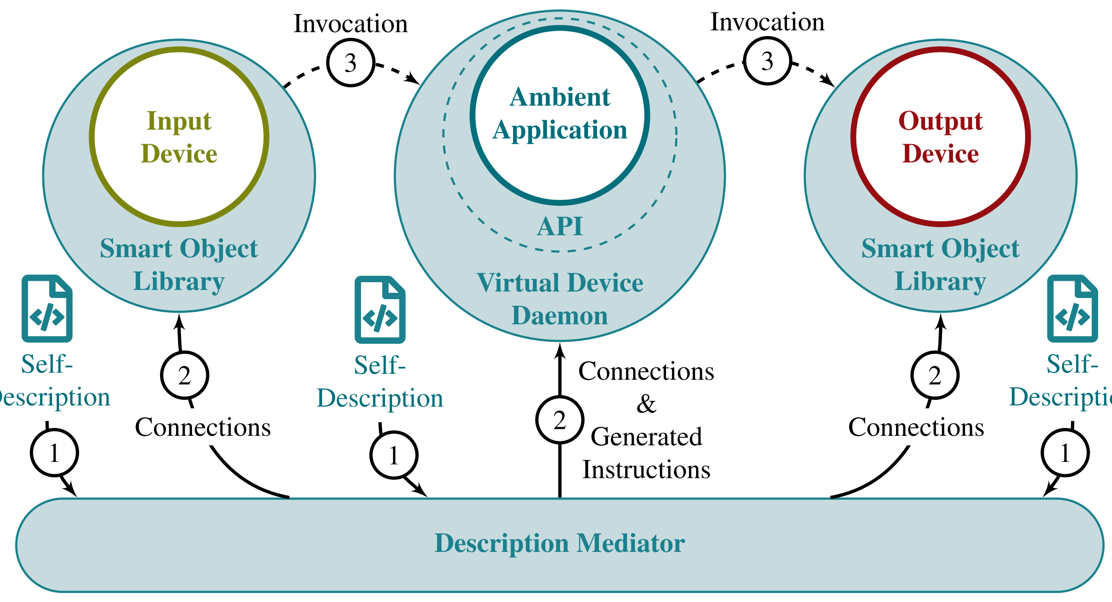


FIGURE 1
Overview of the components involved in the Ambient Reflection framework: Input and output devices utilize the Smart Object Library to send messages to connected devices and Ambient Applications. Meanwhile, ambient applications running in a browser use the Virtual Device Daemon via a JavaScript API for the same purpose. Involved components provide self-descriptions to the Description Mediator, which is responsible for setting up the connections. Upon discovery, the Description Mediator requests these self-descriptions (1). Using this information, it performs the ensembling process and generates corresponding instructions. Next, it sends the connection details to the relevant components and delivers the rendered instructions to components capable of displaying them (2). Finally, components can exchange messages without the Description Mediator (3).

configuration as well as pre-packaging, and finally, (c) the usability challenges related to the number of devices and apps. Based on these insights, the authors developed a proof-of-concept implementation of an IoT app store called *UbiBazaar* that allows the deployment of general applications to devices with specific characteristics.

Moreover, the usage of *Multi-Agent Systems* has been proposed to provide a decentralized solution for the deployment of applications in smart environments (Piette et al., 2016). The authors model the deployment on a given infrastructure based on graphs and apply a graph matching algorithm to find entities that can support running the application. Moreover, to ensure resource and information privacy, the authors propose a Multi-Agent System. Applications are managed by application agents during runtime. They guarantee the consistency of the application. In order to deploy or undeploy required functionalities of the application, they cooperate with one or more infrastructure agents. Infrastructure agents deal only with parts of the global infrastructure and do not share this information with other agents. These agents propose (partial) solutions for the deployment of functionality. They are also capable of deploying and undeploying functionalities of an application on request. Further agents provide information about the user or manage groups of infrastructure agents.

Kubitza (2017) introduces the concept of extensible smart environments based on an application store. These applications are then able to dynamically use the capabilities of available smart objects. They provide a unified schema for accessing sensors and actuators of heterogeneous devices from within these applications, which are running within a smart environment. Furthermore, they present a middleware and a runtime that implement this approach.

Malik et al. (2019) present a platform capable of providing virtual representations of physical objects, called *Virtual Objects*. This is based on an application store that can then be used for service composition. It allows for the sharing and discovery of Virtual Objects as well as the provision of micro-services associated with each Virtual Object uploaded into the store. Thus, the platform makes it possible to reuse Virtual Objects and respective services residing in multiple spaces in different contexts and scenarios.

These works provide valuable insights regarding the importance of mechanisms for deploying and managing applications within smart environments. Moreover, they present concepts and technical details of the implementation of respective systems. But none of the described publications incorporates the concept of self-explainability. Furthermore, there has been no particular focus on human-computer interaction when providing applications for smart environments. However, it has previously been argued that users would benefit from a focus on the interaction when providing and explaining ensembles of smart objects and applications.

The provision of applications that are part of an interactive system particularly highlights the following aspect: an application should not only run on a computer that is technically capable of executing it. Instead, applications should be dispatched to devices with computing capabilities that further satisfy certain constraints, e.g., devices with display capabilities or devices that can make use of specific input or sensing components. For instance, an application that makes use of a graphical user interface needs to be able to display it. Dynamically changing situations and the option for users to bring their own or in general further devices into the environment, requires a highly flexible solution for device and application management and their interconnection.

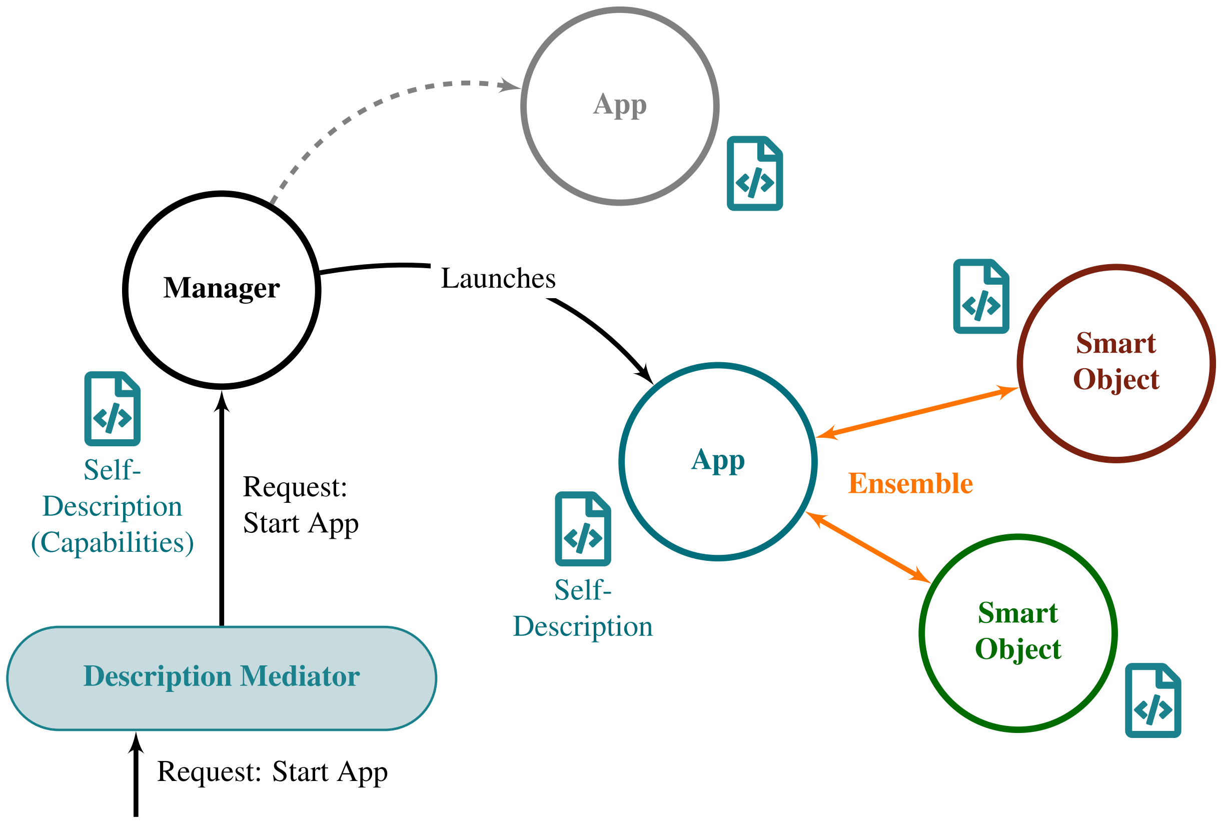


FIGURE 2
Overview of the provisioning of Ambient Applications using a manager. A user may request the Description Mediator to start an application, which, in turn, requests a manager capable of executing an application to do so. In principle, managers may be capable of launching multiple applications (indicated by the gray dashed line). Once started, applications can be connected to smart objects or other applications by the Description Mediator to form an ensemble.

4 A framework for managing self-reflecting ambient applications

As already indicated in the sections above, managing Ambient Applications in dynamically changing environments requires a discovery of physical devices capable of the execution that satisfy all given constraints in order to dispatch the requested applications to these devices. This also requires information about the capabilities of the device.

Ambient Reflection provides both a description language for smart objects and Ambient Applications that also covers device capabilities in a formal format as well as software components for the discovery and connection of respective elements in the network. However, as already explained, the framework is not capable of managing applications, but instead expects these to be prepared manually by users or administrators of the smart environment.

In the following, we present the extensions made to the Ambient Reflection framework that enable it to support the management of Ambient Applications. These extensions are an original contribution of this paper. As already described, the existing framework basically consists of three components. First, the Smart Object Library provides an easy way to integrate smart objects into the framework. Second, the Virtual Device Daemon and its corresponding Virtual Device API allow for an integration of web applications and other software that runs in restricted environments or software that cannot make use of the Smart Object Library. And third, the Description Mediator is used to connect smart objects or Ambient Applications.

Figure 2 depicts the application management process implemented with the components previously introduced. The following initial situation is assumed: A device runs a process with application management capabilities. This manager describes itself and the restrictions of the device on which it is running in a self-description. A call is made through the Description Mediator that prompts the manager to start an application. This application also has a self-description. Once the requested application is started, connections to other components in the environment can be established forming an ensemble.

After launch and connection, instructions for operating the system can then be generated based on the self-descriptions of all devices available in the ensemble. Managers could be bundled with devices that are intended to provide management functionality for smart environments, or they can be provided for existing infrastructure.

The following sections describe the components and their behavior in detail. Moreover, we introduce the new Limited Application Manager that handles devices capable of running only one Ambient Application at a time. This allows for the management of applications that use resources that should not be shared, such as display or audio capabilities.

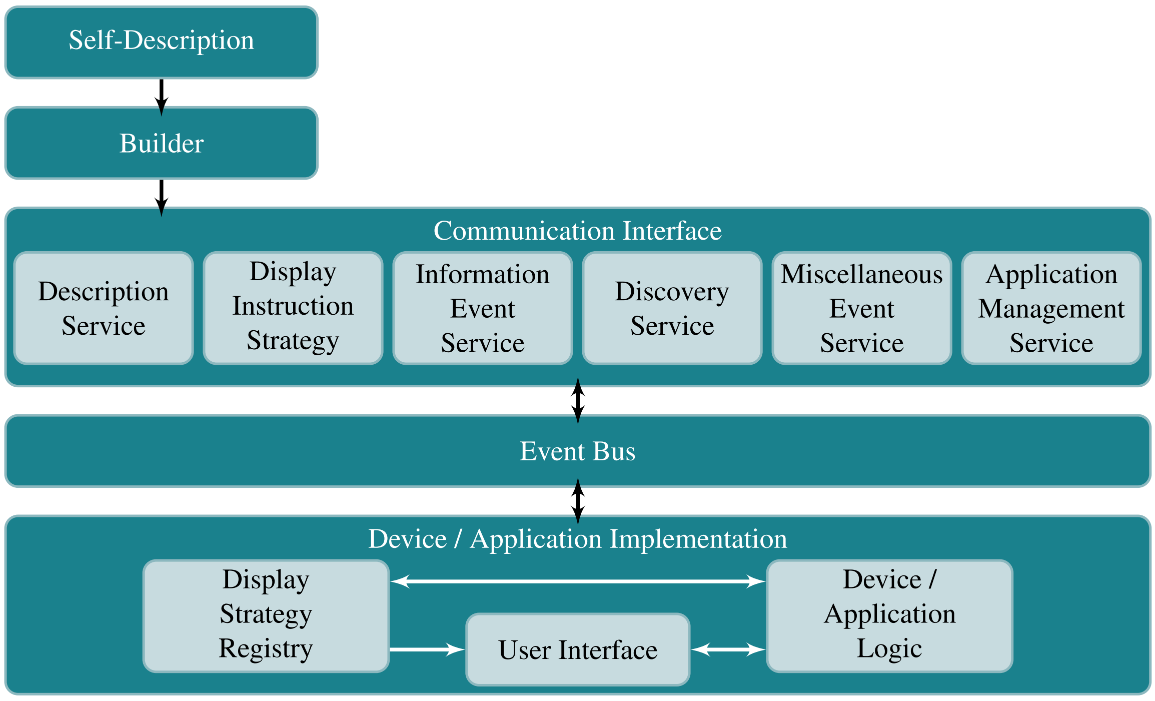

Self-Description
Builder
Communication Interface
Description Service
Display Instruction Strategy
Information Event Service
Discovery Service
Miscellaneous Event Service
Application Management Service
Event Bus
Device / Application Implementation
Display Strategy Registry
User Interface
Device / Application Logic

FIGURE 3
Structure of the smart object library (cf. Burmeister, 2018). An event bus is used to exchange messages between the communication interface and the device or application-specific implementation. The software component for the smart object or the Ambient Application is initialized by a builder based on a self-description. Various services for self-descriptions, configurations, interactions and non-interaction-related messages, as well as application management are provided via a communication interface.

4.1 Smart object library

In order to integrate devices into the framework, a programming interface called the *Smart Object Library* can be used. It manages all network communication within the framework and is founded on an event bus to handle invocations. A factory pattern for a simple setup and instantiation of smart objects can be utilized.

We extended the device library to provide mechanisms for the management of ambient applications. This includes a service for starting and stopping applications with specific environment and configuration options. Refer to Figure 3 for an overview of the structure of the library including the newly added application management service.

The extensions made to the framework also cover an extensible component for the management of apps. The application manager is supposed to pull the given version of an application from a defined and trusted repository or application hub. Actual implementations of application managers may inherit from this component and implement the respective methods to offer management functionality based on various conceivable foundations (e.g., based on virtualization, containerization, modified environments, or other concepts of sandboxing).

As a proof of concept, we implemented an application manager that executes Ambient Applications as subprocesses on the machine. This manager pulls the given version of an application from a defined and trusted application repository. We further introduced the option to configure the repository's origin allowing to select a custom data source. Obviously, the configuration should point to a trusted repository of application archives. The application is loaded as an archive that gets extracted after an integrity check based on a checksum comparison.

It is generally possible to start several applications on a single managed device. However, depending on the device and its configuration, some devices may only be able to run one application because resources must be reserved (e.g., the display or other actuators, ports that can only be used once, etc.). Whenever this is the case and there is already an instance of a component integrated into the framework, this instance will be shut down before the requested application is started. This avoids resource deadlocks.

Next, the application manager executes the pulled application using the given runtime environment and configuration for this application. Input and output channels for the launched application can be inherited from the parent component. Logs may be attached to existing ones at the same location by also inheriting the logging configuration. If the parent application that contains the application manager stopped components connected to the framework prior to the execution of the requested application, these components resume their work once the application terminates (via regular exit or via a requested shutdown).

Finally, we introduced a handler for browsing requests after launching an application. This becomes relevant when the requested application is web-based and makes use of the Virtual Device Daemon and its corresponding programming interface in order to be integrated into Ambient Reflection. When using this component, the web browser is instructed to open the given

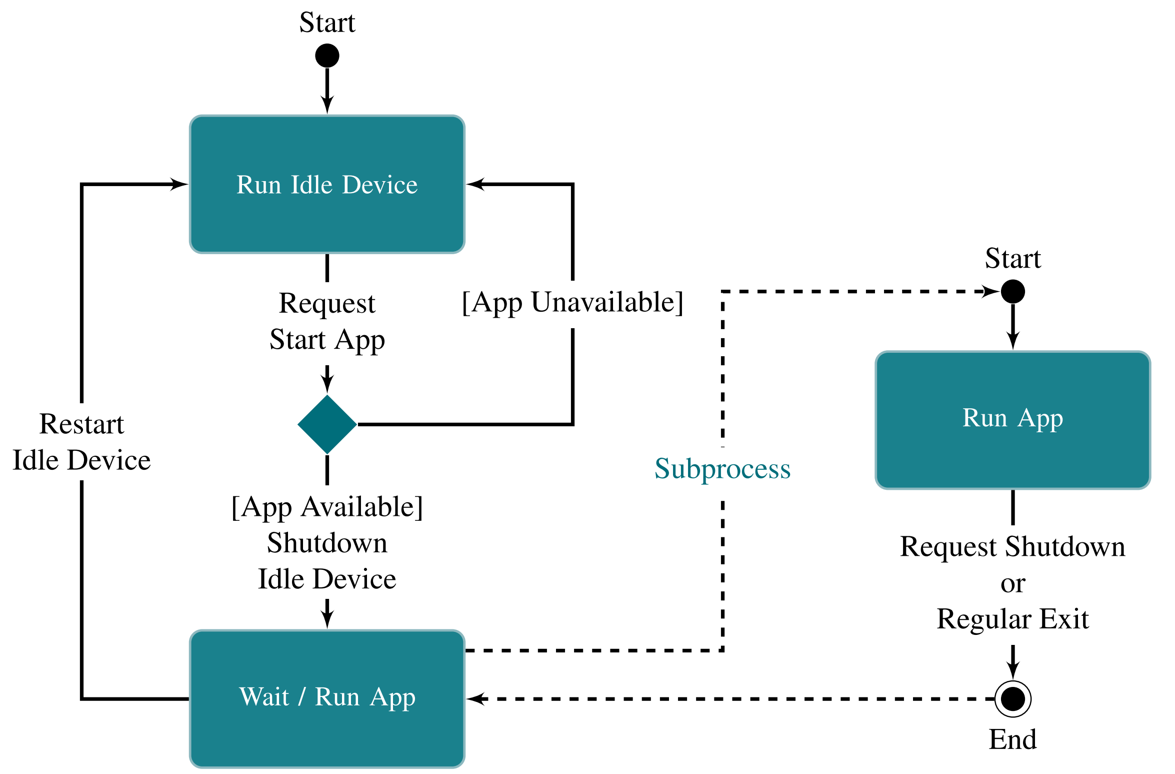


FIGURE 4
Lifecycle of a limited application manager/an idle device that launches Ambient Applications.

address. It is also possible to request the full screen or kiosk mode of the browser.

4.2 Virtual device daemon and API

The *Virtual Device Daemon* uses web sockets to act as a proxy for components that cannot access the Smart Object Library directly.

We extended the Virtual Device Daemon to also support application management. Since the daemon's implementation is based on the Smart Object Library, updating to our recent version of the library covered most aspects required for the management. Furthermore, we implemented extensions regarding the message exchange for network events (discovery and loss of smart objects/Ambient Applications) as well as application termination requests.

4.3 Limited application manager

In many situations, only one application can be applied to a device at a time, since applications may occupy relevant resources that are limited (e.g., foreground display capability, audio speakers, or, more basically, certain ports). For these cases, we propose using a device upgrade, where, at first, only a minimal instance is running in order to be discoverable and to provide the application management services that allow for the startup of an application. We also refer to this minimal instance as an *idle device*, since its main purpose is to accept a request for launching an application. Hence, the launch of an application serves as a device upgrade allowing the device to provide functionality and a user interface. This is similar to a protocol upgrade.

A managing component can then discover devices running these minimal instances and next, can request device information that includes respective capabilities. A distribution algorithm is able to assign requested applications to discovered devices capable of executing these based on the device information. Next, the selected device will be requested to start the assigned application (see Figure 4). First, the running minimal instance checks the availability of the requested application. If not available, the instance will respond with an error message and continue its work. Otherwise, the application will be pulled and prepared to be executed using the given runtime environment and configuration options. Then, the running minimal instance will be stopped in order to free used resources, and next, the requested application will be started as a subprocess. For this purpose, launched applications are configured to offer a service that allows for a termination of the application by the Smart Object Library. Unless explicitly specified otherwise, launched applications do not provide management capabilities. Additionally, the changes made to the framework make it possible to inherit the identifier of the parent for the started application.

If requested, a web application can also be opened using any installed browser. In doing so, web-based applications that are integrated using the Virtual Device Daemon and the corresponding API can also be started on the device and integrated into the framework. Once the started application

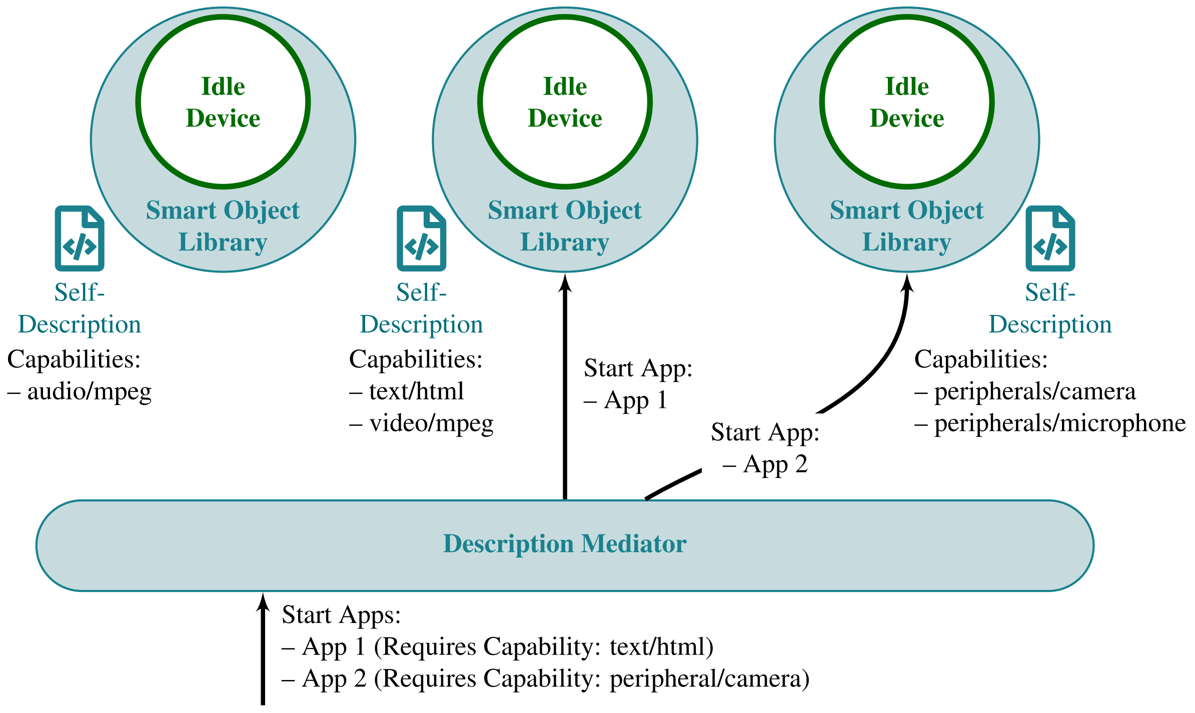


FIGURE 5
The distribution of Ambient Applications to application managers (in this case idle devices) capable of launching the applications. In this example, three limited application managers are running in a smart environment. Here, two applications are requested. The Description Mediator distributes these two applications to two of the managers.

terminates, the minimal instance will be restarted to be able to accept further launch requests.

4.4 Description mediator

The Description Mediator is the core component that is used to connect components based on their self-description and to provide instructions for these connected ensembles as well as to inform the involved components about both. The mediator merges descriptions of connected components into a description of the whole ensemble that serves as a foundation for the generation of user instructions.

Although devices may be requested to start applications using the respective communication protocol, this would require a component to discover the devices as well as their capabilities and to dispatch applications to these devices. We extended the Representational State Transfer (REST) interface of the Description Mediator to accept application launch requests (see Figure 5). It is possible to request the start on a specified device or to request a distribution on any device capable of executing it.

We also introduced a distribution algorithm that takes into account the properties of the device. This includes hardware as well as software capabilities. Software capabilities can be expressed using MIME types (Multipurpose Internet Mail Extension), a widely used format for the indication of media types in digital systems. We propose a format that is inspired by MIME types to describe hardware properties. This also covers peripherals connected to the devices that may be used. Although further criteria are conceivable, we implemented the first prototype of the distribution algorithm only based on the described capabilities.

4.5 Instruction rendering engine

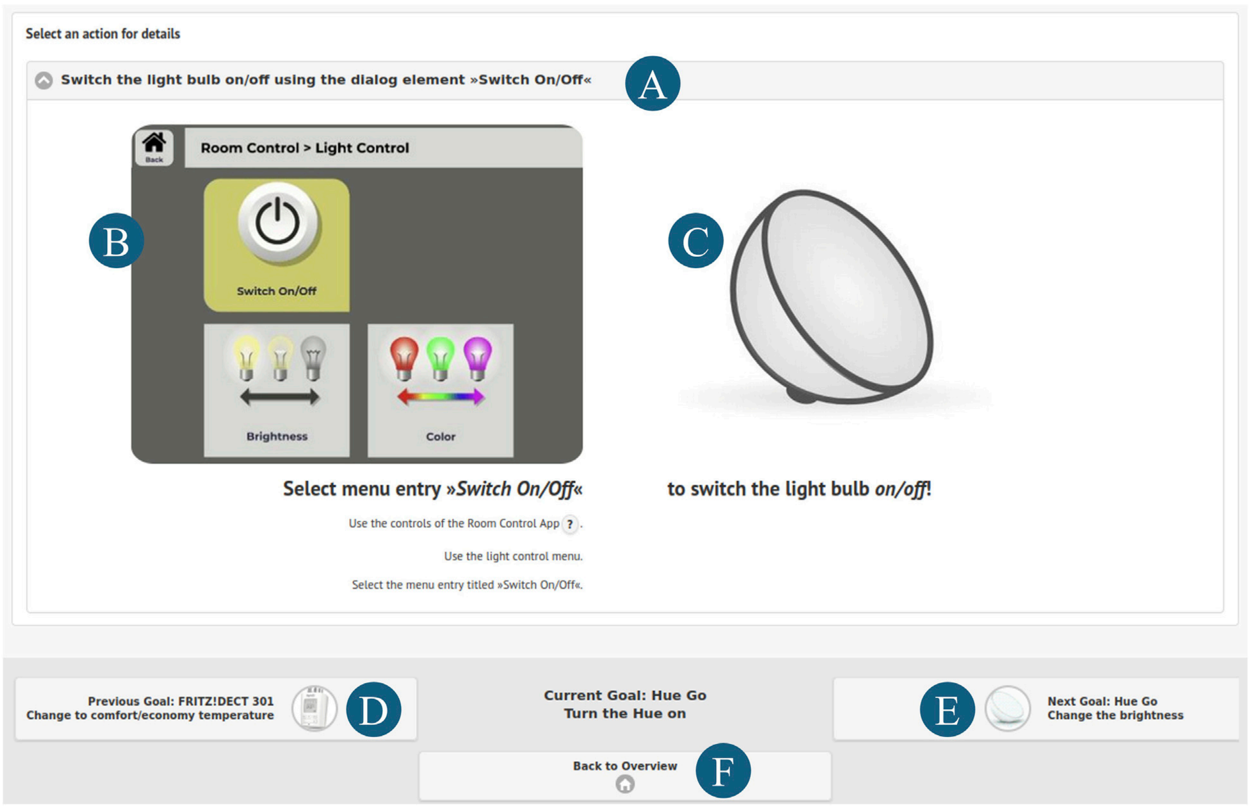
The Description Mediator merges the self-descriptions of the components involved in an ensemble into an aggregated description (see Burmeister, 2018). In addition, plain textual instructions, purely pictorial guides, and simple tutorials based on the Hypertext Markup Language (HTML) can be generated using respective rendering engines.

We extended the HTML rendering engine in order to allow for more detailed multimedia instructions that also fully support Ambient Applications (a screenshot is depicted in Figure 6). In doing so, we enabled browser-based guides that integrate textual descriptions with graphical representations of required user input actions and corresponding system responses. Particularly, animations of the required interaction steps can be provided. Consequently, purely textual instructions can be created, as well as instructions that are enriched with images or animations. For multi-step instructions, an overview page provides access to detailed explanations of each task, with navigation elements in the header and footer for easy access.

The following section describes a user study conducted to evaluate the quality of the generated instructions.

5 Evaluation of the self-explainability

In order to investigate the aspect of self-explainability provided by the described framework, we conducted a laboratory study with students. In general, there is a plethora of conceivable combinations of smart objects and Ambient Applications for various application


Select an action for details
Switch the light bulb on/off using the dialog element »Switch On/Off«
A
Room Control > Light Control
Back
Switch On/Off
Brightness
Color
B
C
D
E
F
Previous Goal: FRITZ!DECT 301
Change to comfort/economy temperature
Current Goal: Hue Go
Turn the Hue on
Next Goal: Hue Go
Change the brightness
Back to Overview

Select menu entry »*Switch On/Off*« to switch the light bulb *on/off*!

Use the controls of the Room Control App ?.

Use the light control menu.

Select the menu entry titled »Switch On/Off«.

FIGURE 6
A screenshot of rendered multimedia instructions for controlling a smart light by using a given Ambient Application. These instructions were generated from the self-descriptions of the involved components. The top of the figure shows a collapsible **(A)** which could be expanded to provide a more thorough explanation of this interaction step. The details view contains an image of the application element **(B)** and an animation of the reaction of the output device **(C)** Additionally, text instructions of the required input actions and the corresponding system response are depicted. The navigation bar is situated at the bottom of the screen. It allows switching to the previous instruction **(D)** the next instruction **(E)** and navigating back to the overview of the entire ensemble **(F)**.

domains. However, in order to make a claim to transferability and a contribution to general validity, we have selected three representative application scenarios that cover relevant application domains of smart environments. One focuses on a smart home application, and two scenarios address the medical domain.

5.1 Study design

We conducted a quasi-experimental mixed-design user study that combined qualitative and quantitative instruments. The purpose of this study was to evaluate the clarity, understandability, and some design choices of the instructions generated by the described Ambient Reflection framework.

Participants were asked to perform tasks using the ensembles and, subsequently, to evaluate the system based on their experience. During the process, we collected objective performance data and recorded subjective data in subsequent interviews and questionnaires. The procedure includes two questionnaires, a participant observation, and a semi-structured interview.

The purpose of this procedure is to answer the following research questions associated with our second research question regarding the self-explainability of our proposed framework:

(a) Assuming that self-descriptions of the components of a typical ensemble are given. Is it possible to generate instructions based on these in order to effectively convey the control and system response of the ensembles to users?
(b) In which of the scenarios is an integration of the instructions into the respective application preferable to a separate display on a different device?
(c) Are pure textual instructions preferred to animated or illustrated instructions?

5.2 Sample

We recruited students from the University of Lübeck for the user study. We sent invitation emails and also personally contacted the students.

Persons who are not fluent enough in German or English to understand and answer the items on the questionnaires or to understand the instructions generated by the system were excluded.

In total, we recruited twelve participants, seven male and five female. The average age was 25 years (with a standard deviation of 3.6 years). Seven participants studied computer science, two human medicine, and three health and healthcare sciences.

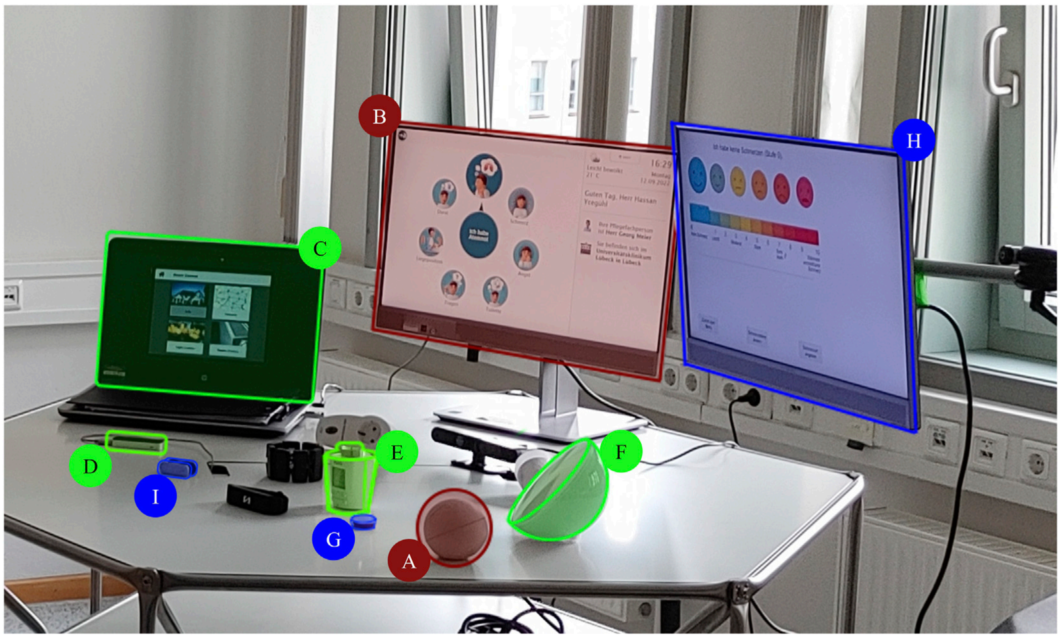


FIGURE 7
Laboratory set-up for the evaluation of self-explainability. The first ensemble with the ball-shaped interaction device **(A)** and the Augmentative and Alternative Control Application **(B)** is highlighted in red. The second ensemble, which includes the room-control application **(C)** with the hand tracking gesture controller **(D)** the smart thermostat **(E)** and the smart light **(F)** is highlighted in green. The smart button **(G)** the pain application **(H)** as well as the touch-sensitive gesture controller **(I)** are all part of the third ensemble, which is highlighted in blue.

Most of the participants had not interacted with any of the interaction devices involved prior to the experiment. However, they knew about the concept of smart objects in general, e.g., smart lights.

5.3 Study procedure and instruments

During the study, objective and subjective data were collected. This includes sociodemographic, performance, interview, and survey data.

At the beginning of the study, participants were introduced and provided with detailed information about the research subject and the study procedure. They then completed a sociodemographic questionnaire including items on previous experience with smart environments before proceeding to the practical phase of the study. The participant observation was divided into three phases, each of which was followed by interview questions. Finally, we asked the participants to complete a questionnaire to evaluate the instructions of the system. The laboratory setup for the study is shown in Figure 7.

5.3.1 Sociodemographic questionnaire and previous experience

The questionnaire covers general information, such as age, gender, recent occupation, study program (if currently studying), and a self-assessment of English language skills. In addition, items related to previous experience with the interaction devices involved and items related to the affinity for technology interaction (ATI) as published by Franke et al. (2018) were included.

5.3.2 Participant observation

The participant observation covers three parts, each asking the participants to learn the control of an ensemble. In total, participants were asked to perform five tasks using three ensembles (see Table 1). The tasks were presented in a fixed order since they were designed to progressively increase in difficulty and ensemble complexity.

We first asked the participants to make use of an ensemble to control an application for Augmentative and Alternative Control (AAC) using a ball-shaped interaction device (BIRDY, cf. Kordts 2023). BIRDY is controlled using tilt and press gestures. The application addresses care scenarios where patients are unable to express their needs, but may rely on a system to assist with communication. Users navigate through a circular menu to select and thereby express needs.

The next three tasks were to use a room-control application to control smart lights and smart thermostats. The user navigates through a grid menu to select elements for controlling the lights and heaters. The application in turn was controlled by a camera-based hand tracking gesture controller (Leap Motion).

Finally, a touch-sensitive gesture controller worn between index and middle finger (LITHO) and a smart bluetooth button (Flic) was used to control a pain documentation application for smart hospitals. This application is supposed to help with pain management in hospital wards, particularly in intensive care units. Users navigate through a wizard, selecting the pain level and location.

For each task, the participants were told to read the instructions in order to find out how the respective ensemble could be controlled. They were also asked to identify the devices involved in each ensemble and to use the identified devices to perform the actions

TABLE 1 The tasks, the required components and the minimum number of input actions required to solve the interaction task using the given ensemble during the participant observation. Note, that the instructions were always displayed on the same monitor (smart white board).

Task	Involved components	Description	Inputs
1	Ball-Shaped Interaction Device (BIRDY), AAC Application	Read the instructions to find out how to control the communication application. Identify the required devices (selection on a table) and perform the necessary actions to select the menu element *pain*	3
2	Hand Gesture Controller (Leap Motion), Room-Control Application	Read the instructions to find out how to control the room control application. Identify the required devices (selection on a table) and perform the necessary actions to access the *light control* menu	2
3	Hand Gesture Controller (Leap Motion), Room-Control Application, Smart Lights (Philips Hue Go)	Read the instructions to find out how to change the color of the lamp. Identify the required devices (selection on a table) and perform the necessary actions to set the lamp color to blue	3
4	Hand Gesture Controller (Leap Motion), Room-Control Application, Smart Thermostat (FRITZ!DECT 301)	Read the instructions to find out how to change the temperature of the heater. Identify the required devices (selection on a table) and perform the necessary actions to set the temperature to a value of your choice	≥ 8
5	Smart Button (Flic), Touch Controller (LITHO), Pain Documentation Application	Read the instructions to find out how to control the pain documentation application. Identify the required devices (selection on a table) and perform the necessary actions to document that you have severe pain (6 out of 10) in your left leg	11

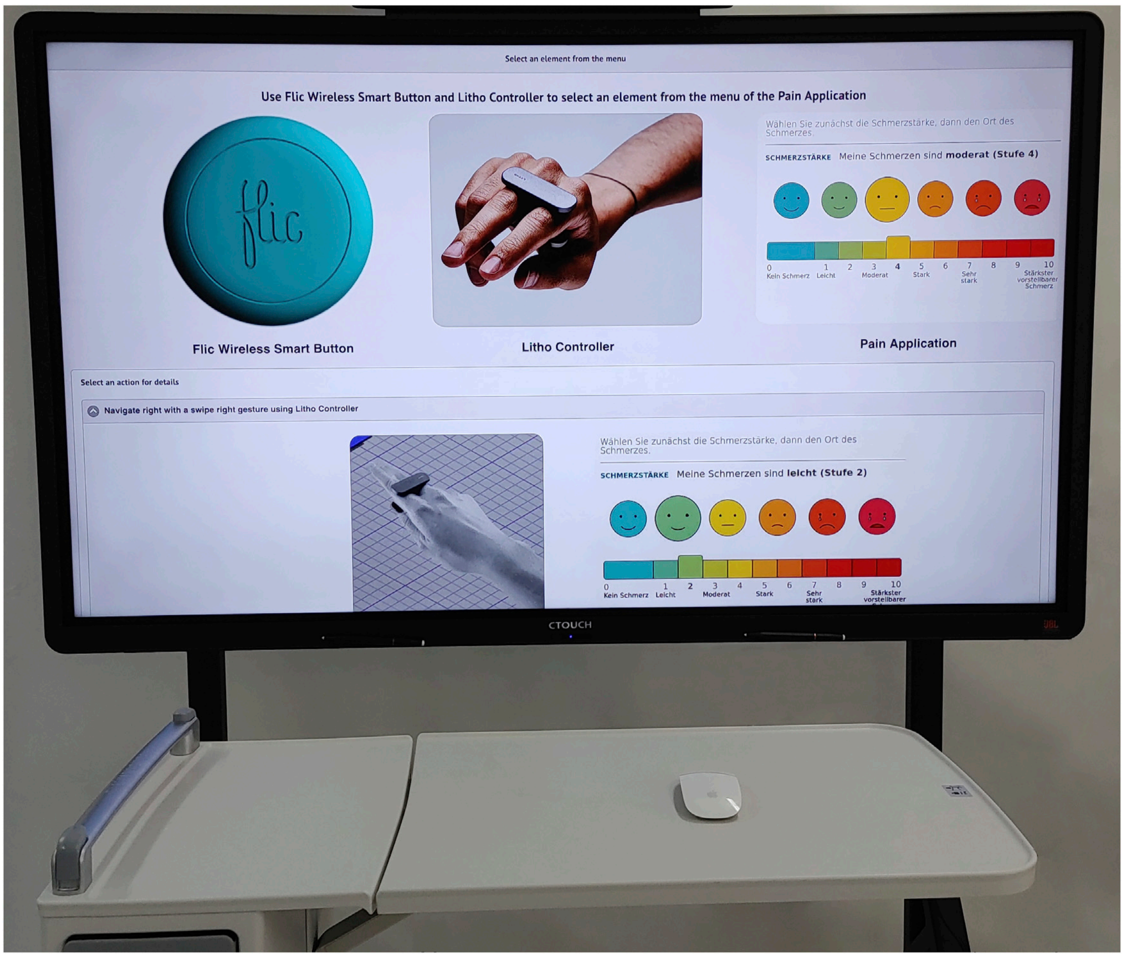

Select an element from the menu
Use Flic Wireless Smart Button and Litho Controller to select an element from the menu of the Pain Application
Wählen Sie zunächst die Schmerzstärke, dann den Ort des Schmerzes.
SCHMERZSTÄRKE Meine Schmerzen sind moderat (Stufe 4)
0 Kein Schmerz
1 Leicht
2
3 Moderat
4
5 Stark
6
7 Sehr stark
8
9
10 Stärkster vorstellbarer Schmerz
Flic Wireless Smart Button
Litho Controller
Pain Application
Select an action for details
Navigate right with a swipe right gesture using Litho Controller
Wählen Sie zunächst die Schmerzstärke, dann den Ort des Schmerzes.
SCHMERZSTÄRKE Meine Schmerzen sind leicht (Stufe 2)
CTOUCH

FIGURE 8
Laboratory set-up of the screen with instructions for the ensembles displayed (in this case instructions for the fifth task).

necessary to complete the given tasks. Towards this end, we presented generated instructions for each ensemble on a separate display (rectangular smart white board) and allowed the participants to read and understand these instructions (see Figure 8). Participants interacted with the instructions using a mouse. We provided animations for the required input actions with most components within these instructions. Only the interaction with the Leap Motion controller and the Flic button had still images without any animation. We also provided textual descriptions of the input actions for all components.

For the task of identifying the devices involved in the respective ensembles, a selection of devices was presented on a table, with applications displayed on monitors placed around the table. There were more devices on the table than were required for the tasks in order to avoid the selection becoming obvious due to devices already assigned (see Figure 7).

We used a wizard-of-Oz approach, meaning that participants did not actually control the applications, but rather the applications were remotely controlled by the investigator. For incorrect inputs, participants were told that their input had no effect.

In general, participants were asked to verbally express their intentions and intended actions to make the procedure transparent. Furthermore, participants could ask questions or make comments at any time during the process.

Key aspects assessed included error count and error type, task completion time (including reading the instructions, identifying the involved components and interacting with the ensemble), task progression, and how often the participants looked at the instructions. Any issues encountered were recorded in a problem log.

5.3.3 Semi-structured interview

After solving the task with each of the ensembles, we asked three interview questions to identify possible problems, to discuss the place of the presentation of the instructions and to give room for further remarks. We asked whether there are any uncertainties and at what points they occur. We also asked where in the room or on which device the participants would expect the instructions to be displayed.

5.3.4 Survey data

Finally, participants were asked to complete a questionnaire to assess the comprehensibility of the generated instructions.

6 Results

The provided aspect of self-explanability helped the participants to solve the tasks. In addition, the instructions were considered to be very understandable and particularly helpful in the learning process of controlling the ensembles. However, the participants also pointed out some ambiguities and suggested improvements. The main findings of the study are presented below.

6.1 Affinity for technology interaction

The sample had a high affinity for technology interaction of 4.57 (with a standard deviation of 1.25 and Cronbachs alpha of 0.96) compared to the expected mean of the German population of 3.6 (with a standard deviation of 1.08) reported by Franke et al. (2018).

6.2 Participant observation

The participants solved all tasks posed successfully, although several interpretation mistakes were made during the second and fifth task. Mean errors for the five tasks can be seen in Figure 9.

Some participants had problems interpreting illustrations of the gesture control of the Leap Motion controller as well as the Flic button. Moreover, participants did not notice the perequisite of the LITHO controller to additionally place the thumb on the device when swiping. Though, most participants noticed their mistake after another, more thorough look at the instructions and particularly reading the provided instruction texts. Others attempted a trial-and-error-process to find the correct gesture.

All but one participant were able to identify the involved devices and applications on the first attempt.

Furthermore, the navigation path within the applications was mostly clear. Mistakes were made due to testing gestures. A minor ambiguity was remarked on the fourth task. Some of the participants were not sure how to return from a submenu to the main menu.

Figure 10 depicts the processing times.

6.3 Interviews

In the interviews, five participants explicitly stated that they found the instructions or their structure clear. In particular, the illustrations and animations were positively mentioned by two of the participants. However, one participant pointed out that animations are not necessary for every illustration.

Two participants saw ambiguities in the design of the instructions. This was particularly the case with the collapsible menus, that are used to group parts of the instructions and hide details about the interaction execution and system responses. Participants did not realize that these elements could be expanded to display their contents. In addition, four participants noted that some wording in the instructions was too vague or misleading. In particular, the instructions for using the Flic button were ambiguous, stating that the finger must be moved toward the device. Instead, the button needs to be pressed.

Two of the participants stated that their interpretation errors with respect to the control of the Leap motion were caused by the selected illustrations of the gestures. Three participants commented negatively on redundancy in the instructions, especially for controlling the smart lights. Two of the participants had wished for more information about the menu structure, which is only scarcely addressed due to the black-box view. In addition, four participants said that their misunderstanding was caused by not reading or looking at the instructions carefully–sometimes relying only on the headings. To improve the clarity of instructions and control of some of the devices, three participants suggested step-by-step tutorials.

Eleven of the participants could imagine displaying the instructions on a separate device, while ten of the participants

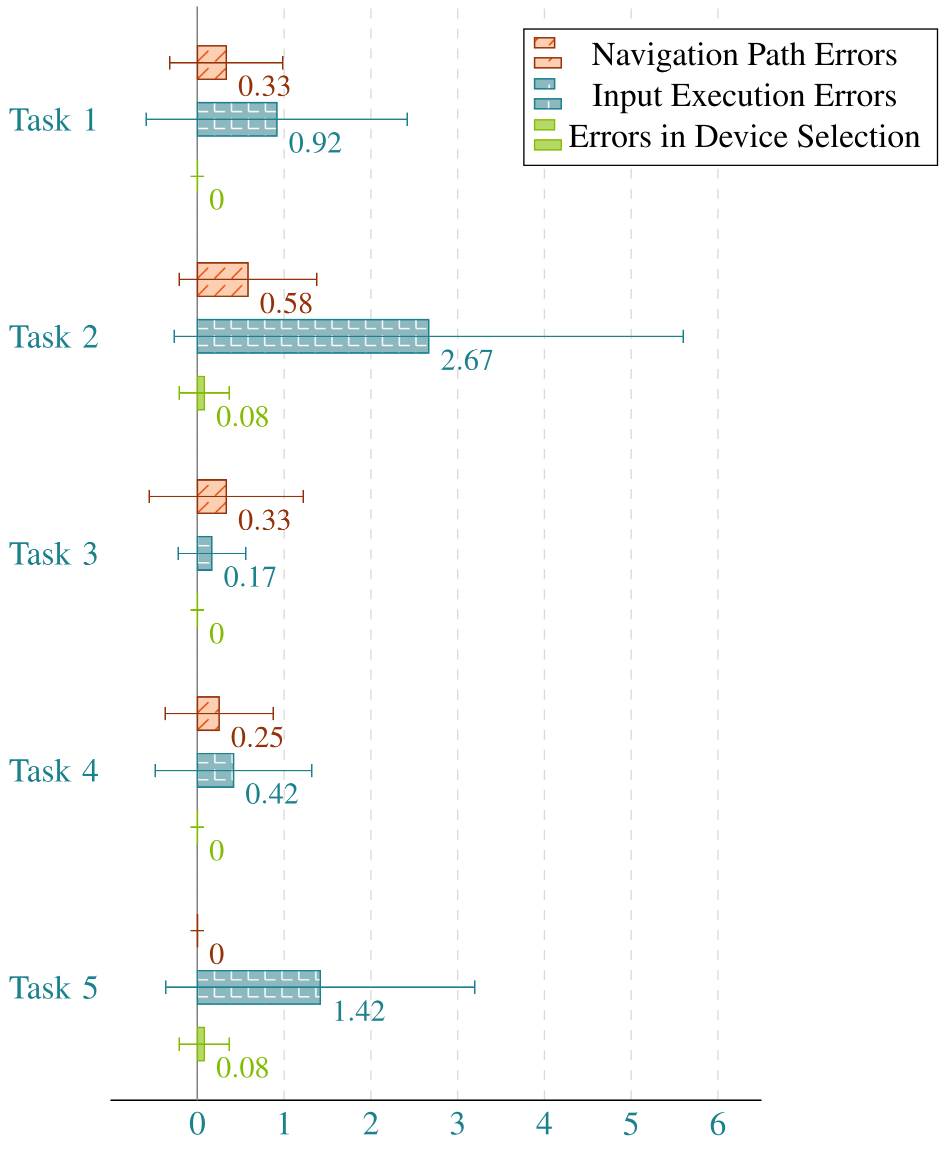


FIGURE 9
Observed errors in the performance of the tasks. The bars represent the average number of errors (for example, a total of 4 navigation errors were observed across all 12 subjects in task 1). The additional markings illustrate the standard deviations. Input errors refer to errors in the execution of the required gestures or input actions that therefore do not result in an input using the designated input device.

could also imagine integrating the instructions into the respective application. Regarding the medical context, participants tended to use a separate device, whereas integration into the applications was considered suitable for smart home applications.

Concerning the integration of instructions into applications, the need to switch between instructions and the application was seen as a potential barrier. Possible forms of integration were seen in help menus or instructions presented prior to using the application itself. When presenting instructions on a separate device, participants noted that these devices should be close to or at least in line of sight of the user.

In addition, a participant noted that applications for the medical field should take into account that personnel should also know how to use the ensembles. Therefore, the participant suggested separate instructions for the staff.

6.4 Questionnaire

The self-explainability of the smart environment(s) was described as very well intelligible. Particularly, the assistance during the learning process of the control of the ensembles was positively noted.

Descriptions of system responses were considered to be well understood. When asked whether animations, illustrations or text descriptions were more helpful in understanding the input actions, there was no consensus among the participants. In general, animations and illustrations were rated as more helpful than text descriptions. However, all three types of presentation were considered helpful.

Language clarity was rated slightly lower than content clarity. In addition, participants stated that repeated descriptions of similar

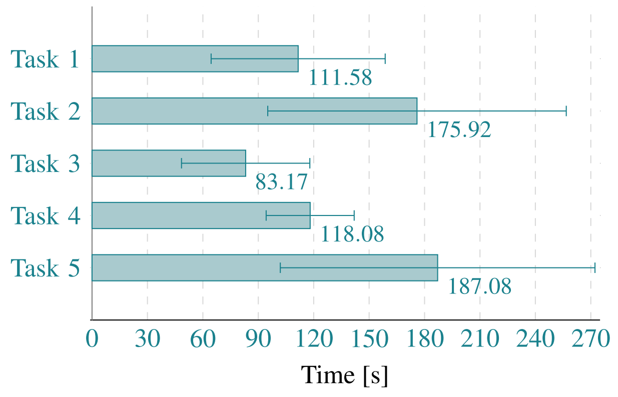


FIGURE 10
The average time in seconds required by the participants to solve the tasks posed. The additional markings illustrate the standard deviations.

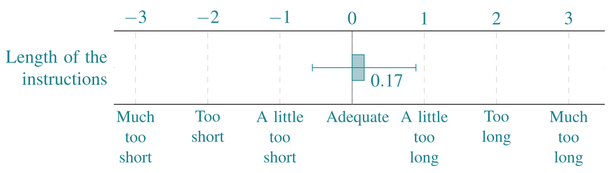


FIGURE 11
Responses to the questionnaire to evaluate the length of the generated instructions for the ensembles used in the course of the study. The bar represents the average values of the responses and the additional marker reflects the standard deviation.

objectives could have a negative impact on the clarity of the instructions. However, the overall size of the instructions was judged to be adequate.

The results of the questionnaire are depicted in Figures 11, 12.

7 Discussion

Several key findings can be derived from the described results. The high success rate and the reported clarity of the instructions, as well as their structure, underline the assistance provided by the generated instructions. In general, all participants were able to learn the interaction effectively within an appropriate timeframe.

In principle, no direct conclusions can be drawn from the processing times. However, in combination with the observation and the comparison with each other, the following interpretations can be made. The required time to solve the given tasks correlates with the interaction mistakes and can also be explained by the consequence to take a more thorough look at the generated instructions. The test subjects were observed to learn the control basics of the ensembles increasingly quickly using the instructions, but particularly struggled with the exact details of the gesture execution in task five. In addition, it was observed that test subjects were able to recall a control once it had been learned (concerning the consecutive tasks two to four). This observation is also consistent with the measured execution times.

Due to the black-box concept, navigation paths within the applications are not covered by the instructions. Only instructions for control options, such as how to control the menus, are covered by the generated manuals. In this respect, the successful navigation through the menus is notable. An explanation could be given by the low complexity of the menu structures involved in the applications. However, we argue that a thorough design of the applications should generally support the understanding of menu structures and required navigation paths.

The high variance in responses regarding whether animations, illustrations, or text descriptions helped understanding input actions may be related to different types of learning. In addition, the specific wording of the texts and the choice of animations and illustrations may have played a role. Regardless of the type of graphical representation implemented, the varying quality of the visualization of the devices may have introduced bias.

The chosen approach therefore entails some limitations detailed in the following, particularly regarding:

- participant diversity,
- size of the study population,
- system autonomy (regarding the wizard-of-Oz approach),
- embeddedness of instructions and realistic scenarios.

Only students were surveyed in the study. They belong to the group of young adults who have grown up in the digital world (i.e., digital natives). Biases in their views and judgments of the instructions due to age distribution, affinity for technological interaction, and educational background cannot be ruled out. While the study population represents a relevant user group of smart homes, other user groups may face different issues.

Limitations also lie in the size of the study population, the recruitment process (selection bias) and the methodological procedure based on interviews, where social desirability may have caused biases. In general, semi-structured interviews may have been influenced by uneven enquiries that cause an interviewer bias. To avoid this bias, we asked open questions and avoided suggestive questions.

A possible bias may also have resulted from the fact that the test subjects did not actually control the systems, but were controlled by the experimenter (according to the wizard-of-Oz principle). Consequently, the system autonomy was not fully tested, so that a bias in the response behavior regarding the comprehensibility of the input actions to be performed is also conceivable. This is particularly true with regard to the actual gesture recognition by the devices.

For the sake of simplicity, traditional applications were essentially used for the evaluation, with very limited integration into the environment. The primary concern here was the understandability of the ensembles and not the realistic use of the applications. In addition, monitors and more or less classic applications are also conceivable in smart environments. In the course of the evaluation, the instructions were also displayed on a rectangular monitor and could be operated by touch or mouse.

This leaves room for future studies that deal with an actual system autonomy and the comprehensibility of generated instructions with a focus on a stronger integration into the

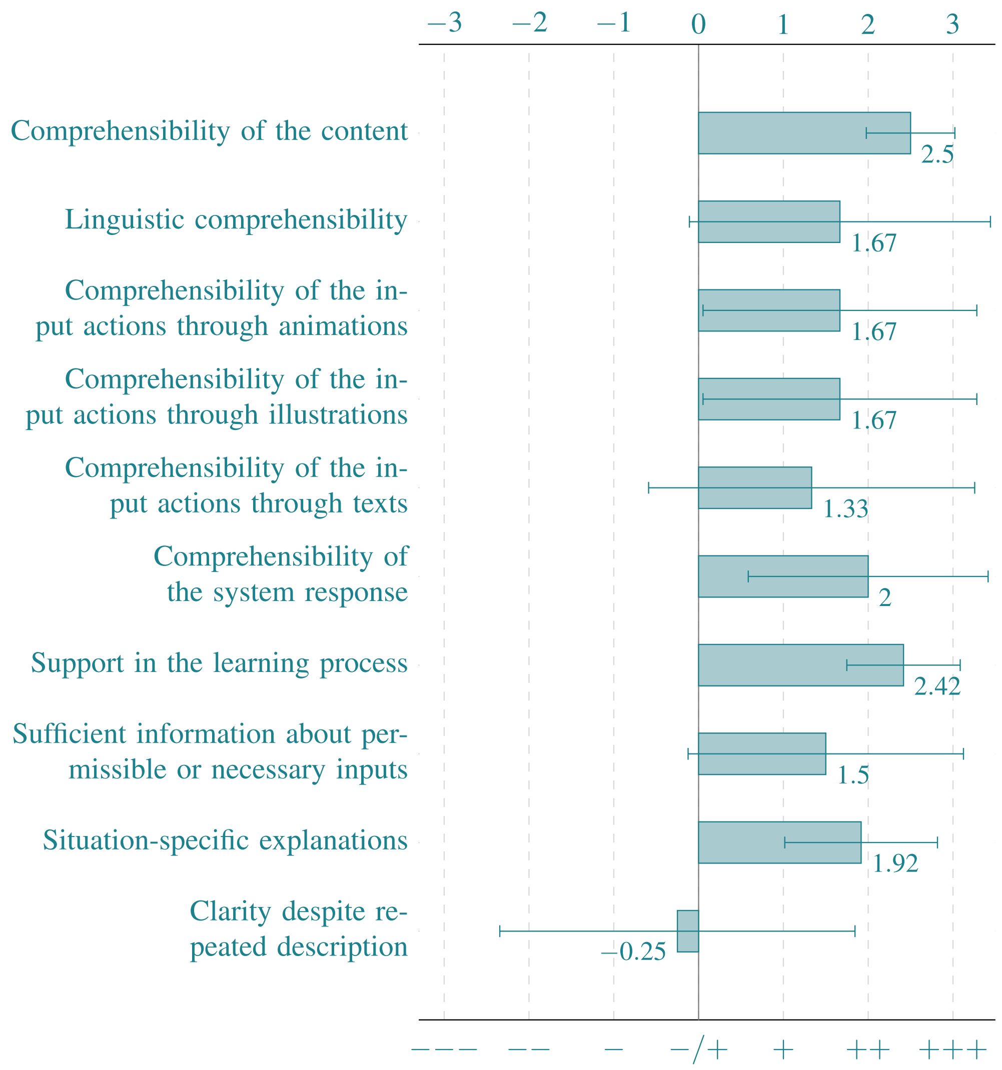


FIGURE 12
Responses to the questionnaire to evaluate the generated instructions for the ensembles used in the course of the study. The bars reflect the average values of the responses, while the additional markings show the standard deviations.

environment. For example, through non-rectangular projections or instructions in other formats, such as audio instructions.

8 Conclusion

In this paper, we introduced a framework for the management of Ambient Applications (see Section 2 for a definition of the term) within smart environments. The framework makes use of an extension of the Smart Object Description Language (SODL) in order to provide self-descriptions for involved smart objects as well as Ambient Applications. Devices capable of executing Ambient Applications are provided with a self-description that covers their ability to manage applications, as well as further capabilities, like connected peripherals or the ability to render specific media formats. Moreover, the concept of a device upgrade allows for the management of applications on restricted devices that can only execute one application at a time. The framework is also capable of distributing applications to discovered devices within the environment that are capable of executing them. Additionally, our system enables dynamic connectivity between smart objects and applications, allowing them to control one another. Furthermore, it can generate usage instructions for these interconnected devices and application networks.

Overall, the presented concept of our system addresses the first research question, where self-explaining Ambient Applications come from and how they can be brought into the environment. The management of Ambient Applications is a first step towards a more general application store concept. An application store would allow users to publish, install, update, and remove self-explaining applications in a smart environment in a usable way. This contributes to the concept of a marketplace for sharing and thereby fostering innovations in the domain of the Internet of Things (Kortuem and Kawsar, 2010).

We further conducted a user study to answer our second research question, to what extent the generated instructions for dynamically connected ensembles provided by our framework are suitable to guide users. By using generated instructions, participants were able to solve all tasks posed successfully. Even complicated input actions were understood. Moreover, participants stated that they found the instructions clear and positively noted the assistance during the learning process. Some illustrations and formulations were criticized, which were initially misinterpreted by several participants. These aspects should be addressed by using adequate illustrations or animations of the required actions within the self-description of the components. Furthermore, the wording of some input actions should be improved within the framework. Limitations of the study approach include a small sample size, the restricted diversity of the participants, the application of the wizard-of-Oz principle, and the chosen way to display the instructions. However, based on the results, it can generally be stated that the framework is able to generate comprehensible instructions for even complex ensembles, provided there is a good self-description of the components involved. Yet, there is room for future studies addressing the shortcomings of our study approach.

In conclusion, our framework can be used to manage self-explaining and self-organizing smart interaction spaces that connect smart objects as well as Ambient Applications. Users may easily launch Ambient Applications that are integrated in a plug-and-play manner. This contributes to our vision of dynamic interconnections within a landscape of present smart objects and applications as well as devices brought by the users and applications installed at will. All of this is adapted and explained to the users and their respective situations, preferences, and needs.

While our framework provides a distribution algorithm that takes device capabilities into account, several further criteria are conceivable. For instance, the positions of the devices may play a role when distributing the applications. Furthermore, (indirect) interconnections between the device and other components may be taken into account. Towards this end, graph-based modeling of the infrastructure and graph matching to identify entities that can support running the application may present a suitable approach (cf. Piette et al., 2016).

Our work may lay the groundwork for further future research. One such direction is the development of a comprehensive application store for Ambient Applications, particularly one that provides a suitable user interface that integrates well into the physical environment. The choice of adequate display and interaction options is an open research question in this regard. Furthermore, deploying at scale for real-world application scenarios is a logical next step. For example, implementing various scenarios in the context of teaching, particularly focusing on providing Ambient Serious Games (cf. Brandl et al., 2023) and other interactive teaching methods, such as station learning, shows potential. Additionally, providing Ambient Applications could support workshop methods like World Café (Brown and Isaacs, 2005). Finally, investigating other instructional formats, particularly non-visual ones, as well as comic-based manuals and situational assistance based on Augmented Reality during operation, is of interest.

Data availability statement

The datasets generated by the survey research during and/or analyzed during the current study are available in the Zenodo repository at: https://doi.org/10.5281/zenodo.16739014.

Ethics statement

Ethical approval was not required for the studies involved humans because it is focusing on the quality and preferences regarding the design of our software solution. The study was conducted in accordance with the Declaration of Helsinki. Ethical review and approval were waived for this study due to anonymously data collection and the informed consent of all participants to the processing and publication of the data. Besides socio-demographic data, no personal data were collected at any time. The participants received no incentive to participate in the study. The study was conducted in accordance with the local legislation and institutional requirements.

Author contributions

BK: Conceptualization, Validation, Software, Methodology, Visualization, Writing – original draft, Writing – review and editing. LB: Conceptualization, Software, Validation, Writing – original draft, Writing – review and editing. AS: Conceptualization, Writing – original draft, Writing – review and editing.

Funding

The author(s) declare that no financial support was received for the research and/or publication of this article.

Acknowledgments

Parts of the content of this work first appeared in the first author's doctoral thesis written in German (Kordts, 2023). This applies to the section on the term Ambient Application, the Instruction Rendering Engine, and the study carried out to assess the self-explainability of our framework. However, the implementation of application management using the Ambient Reflection framework is presented for the first time in this paper. We thank all those who participated in our study.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/friot.2025.1623733/full#supplementary-material

References

Altakrouri, B. (2014). *Ambient assisted living with dynamic interaction ensembles.* Germany: University of Lübeck.

Bellavista, P., Corradi, A., Foschini, L., and Monti, S. (2018). Improved adaptation and survivability *via* dynamic service composition of ubiquitous computing middleware. *IEEE Access* 6, 33604–33620. doi:10.1109/ACCESS.2018.2842683

Brandl, L. C., Kordts, B., and Schrader, A. (2023). "Technological challenges of ambient serious games in higher education," in *Proceedings of the MuM'23 workshops on making a real connection and interruptions and attention management*.

Brown, J., and Isaacs, D. (2005). *The world café: shaping our futures through conversations that matter*. San Francisco, CA: Berrett-Koehler Publishers.

Burmeister, D. (2018). *Selbstreflexive geräteverbünde in smarten umgebungen.* Germany: University of Lübeck.

Delcourt, K., Adreit, F., Arcangeli, J.-P., Hacid, K., Trouilhet, S., and Younes, W. (2021). "Automatic and intelligent composition of pervasive applications - demonstration," in *19th IEEE international conference on pervasive computing and communications (perCom 2021), (kassel (virtual), Germany)*.

Delcourt, K., Trouilhet, S., Arcangeli, J.-P., and Adreit, F. (2024). The human in interactive machine learning: analysis and perspectives for ambient intelligence. *J. Artif. Intell. Res.* 81, 263–305. doi:10.1613/jair.1.15665

Fadiga, K., Houzé, E., Diaconescu, A., and Dessalles, J.-L. (2021). "To do or not to do: finding causal relations in smart homes," in *2021 IEEE international conference on autonomic computing and Self- organizing systems (ACSOS) (washington DC, USA: iEEE)*. doi:10.1109/ACSOS52086.2021.00030

Fey, G., and Drechsler, R. (2020). "Self-explaining digital systems: technical view, implementation aspects, and completeness," in *Advanced boolean techniques: selected papers from the 13th international workshop on boolean problems*. Editors R. Drechsler and M. Soeken (Cham: Springer International Publishing), 1–20.

Fey, G., Fränzle, M., and Drechsler, R. (2022). "Self-explanation in systems of systems," in *2022 IEEE 30th international requirements engineering conference workshops (REW)*, 85–91. doi:10.1109/REW56159.2022.00023

Franke, T., Attig, C., and Wessel, D. (2018). A personal resource for technology interaction: development and validation of the affinity for technology interaction (ATI) scale. *Int. J. Human–Computer Interact.* 35, 456–467. doi:10.1080/10447318.2018.1456150

Garcia Dominguez, A., Bencomo, N., Parra Ullauri, J. M., and Garcia Paucar, L. H. (2019). "Towards history-aware self-adaptation with explanation capabilities," in *2019 IEEE 4th international workshops on foundations and applications of self* systems (FAS*W)*, 18–23. doi:10.1109/FAS-W.2019.00018

Goumopoulos, C., and Mavrommati, I. (2020). A framework for pervasive computing applications based on smart objects and end user development. *J. Syst. Softw.* 162, 110496. doi:10.1016/j.jss.2019.110496

King, E. (2024). *Continuous discovery and goal-oriented control of smart devices in Mobile environments.* USA: The University of Texas at Austin.

Kordts, B. (2023). *Selbstreflexive Smarte Umgebungen Im Intensivkontext*. Germany: University of Lübeck.

Kordts, B., Gerlach, B., and Schrader, A. (2022). Self-organizing and self-explaining pervasive environments by connecting smart objects and applications. *Technologies* 10, 15. doi:10.3390/technologies10010015

Kortuem, G., and Kawsar, F. (2010). Market-based user innovation in the internet of things. *2010 Internet Things (IOT)*, 1–8. doi:10.1109/IOT.2010.5678434

Koussaifi, M., Trouilhet, S., Arcangeli, J.-P., and Bruel, J.-M. (2018). "Ambient intelligence users in the loop: towards a model-driven approach," in *Software technologies: applications and foundations.* Editors M. Mazzara, I. Ober, and G. Salaün (Cham: Springer International Publishing), 558–572. doi:10.1007/978-3-030-04771-9_42

Koussaifi, M., Trouilhet, S., Arcangeli, J.-P., and Bruel, J.-M. (2019). "Automated user-oriented description of emerging composite ambient applications," in *31st international conference on software engineering and knowledge engineering (SEKE 2019) (lisbonne, Portugal)*, 473–478.

Kubitza, T. (2017). "Apps for environments: running interoperable apps in smart environments with the meSchup IoT platform," in *Interoperability and open-source solutions for the internet of things* (Cham: Springer), 158–172. doi:10.1007/978-3-319-56877-5_10

Malik, S., Ahmad, S., and Kim, D. (2019). A novel approach of IoT services orchestration based on multiple sensor and actuator platforms using virtual objects in online IoT app-store. *Sustainability* 11, 5859. doi:10.3390/su11205859

Ornes, S. (2016). The internet of things and the explosion of interconnectivity. *Proc. Natl. Acad. Sci.* 113, 11059–11060. doi:10.1073/pnas.1613921113

Parra-Ullauri, J. M., García-Domínguez, A., García-Paucar, L. H., and Bencomo, N. (2020). "Temporal models for history-aware explainability," in Proceedings of the 12th System Analysis and Modelling Conference (New York, NY, USA: Association for Computing Machinery), 155–164. doi:10.1145/3419804.3420276

Piette, F., El Fallah Seghrouchni, A., Taillibert, P., Caval, C., and Dinont, C. (2016). "A multi-agent middleware for deployment of ambient applications," in *Enablers for smart cities* (John Wiley and Sons, Ltd), 65–106.

Sadeghi, M., Herbold, L., Unterbusch, M., and Vogelsang, A. (2024). "SmartEx: a framework for generating user-centric explanations in smart environments," in *2024 IEEE international conference on pervasive computing and communications (PerCom)*, 106–113. doi:10.1109/PerCom59722.2024.10494449

Stastny, S., Farshchian, B. A., and Vilarinho, T. (2015). "Designing an application Store for the internet of things: requirements and challenges," in *Ambient intelligence.* Editors B. De Ruyter, A. Kameas, P. Chatzimisios, and I. Mavrommati (Springer International Publishing), 313–327. doi:10.1007/978-3-319-26005-1_21