Addressing the Need to Capture Scenarios, Intentions and Preferences: *Interactive Intentional Programming in the Smart Home*

Mathias Funk [1,*], Lin-Lin Chen [1], Shao-Wen Yang [2], and Yen-Kuang Chen [2]

[1] *Industrial Design Department, Eindhoven University of Technology, Eindhoven, the Netherlands*
[2] *Intel Corporation, Santa Clara, California, USA*

The Internet of Things (IoT) and connected products have become part of the advance of ubiquitous technology into personal and professional living spaces, such as the smart home. What connectivity and distributed computing have made possible, is still programmed only according to more or less simplified rule systems (or in traditional code); the mapping between what end users intend or would value and what can be expressed in rules is not straightforward. This article analyzes the temporal, preferential, technical, and social complexity of mapping end-user intent to rules, and it suggests new concepts to better frame information that needs to be captured to create smart-home systems that better match users' intents. We need a new approach aimed at first capturing end users' intentions and potential usage scenarios, then providing this information to a control system that learns to resolve intentions and scenarios for available devices in the context. The new approach should deconstruct and rebuild IoT-related programming at a higher level of abstraction that allows end users to express long-term intentions and short-term preferences, instead of programming rules. Based on related work, a first-person perspective and analysis of current smart-home programming practices, the concept of Interactive Intentional Programming (IIP), is introduced and discussed.

Keywords **–** End-User Programming, IoT, Smart Things, Systems Design, Domain Modeling.

Relevance to Design Practice **–** Smart systems often cannot realize meaningful behavior because they lack contextual information from end users. This article introduces Interactive Intentional Programming as a new approach to designing systems that capture semantically rich, high-level usage scenarios, intentions, and preferences in a novel domain model that can enable adaptive, smart behavior through machine intelligence.

Citation: Funk, M., Chen, L.-L., Yang, S.-W., & Chen, Y.-K. (2018). Addressing the need to capture scenarios, intentions and preferences: Interactive intentional programming in the smart home. *International Journal of Design, 12*(1), 53-66.

Introduction

The last few years have witnessed tremendous growth of the Internet of Things (IoT), including connected and smart devices for the consumer market. A succession of notable products have been introduced and each has marked the advance of technologies into the home, including NEST, a learning thermostat, in 2011; SmartThings Hub, originally launched as a KickStarter project in 2012; Philips Hue, an intelligent LED lighting system, in 2012; and the rivals Amazon Echo and Google Assistant, smart speakers with intelligent personal assistant capabilities, in 2016. Through such products, together with the availability of different types of sensors, smart outlets, smart appliances and devices, it has become possible for consumers to *program* and automate their homes to some extent. Accordingly, smart-home technologies have moved out of well-controlled laboratory environments (De Ruyter & Aarts, 2004; Intille, 2002; Kientz et al., 2008; Offermans, van Essen, & Eggen, 2014) and into the messy and nuanced everyday lives of ordinary people. Connected products are designed with programming interfaces as the primary way to interact with them, thereby blurring the line between *product* and *tool*. The traditional principle of designing products ready for use has faded and been replaced by productized toolboxes that require end users to customize and finish the design themselves to fit their individual contexts and lifestyles.

In this article, we first review the extensive body of work related to end-user programming in the smart home, and then analyze the particular issues, based in part on a first-person user experience, leading to a set of new concepts for better capturing relevant information for the design of smart homes. We outline how these concepts help solve common issues and conclude with an outlook on future steps.

Received May 15, 2017; **Accepted** February 5, 2018; **Published** April 30, 2018.

***Corresponding Author:** m.funk@tue.nl

Related Work

In the following, we review the issues related to end-user programming in the actual contexts of smart homes. While end-user programming or development (EUD) is a large and diverse field, we focus here on issues that emerge when applying EUD to a private, sensitive context of everyday living. Furthermore, this style of EUD requires a mix of manual and screen-based tasks on different devices that are often scattered throughout the home environment.

Tools and Technologies

End-user programming tools for the home environment have been created and investigated in academia (Davidoff, Lee, Yiu, Zimmerman, & Dey, 2006; Dey, Sohn, Streng, & Kodama, 2006; Humble et al., 2003; Newman, Elliott, & Smith, 2008; Offermans et al., 2014; Truong, Huang, & Abowd, 2004). However, they have largely remained at the proof-of-concept level, and today most end users program their smart homes by using trigger–action programming (Ur, McManus, Pak, & Littman, 2014; Ur et al., 2016) to develop rules in the form of *if trigger, then action*. A prime example is the website IFTTT (https://ifttt.com) which stands for *if this, then that*. IFTTT offers services that enable end users to create a new *recipe* by following simple steps to select the event trigger and the corresponding action. These recipes can then be shared with other users via the IFTTT website. Similarly, SmartThings (https://smartthings.com/) offers an interface that allows end users to define trigger conditions and actions. Similarly to IFTTT, SmartThings offers predefined templates for users to easily configure rules. Differently from IFTTT, SmartThings allows some actions to be taken under certain conditions, e.g., preconfigured modes (such as Home, Away, and Night) or selected periods (such as from 9 p.m. to 6 a.m., or from sunset to sunrise). If the standard templates do not meet users' needs (e.g., combining multiple sensor inputs), SmartThings allows users to write their own programs in the Groovy programming language. Stringify (https://stringify.com/) is yet another programming interface that offers trigger conditions and actions. Differently from IFTTT or SmartThings, Stringify uses a drag-and-drop graphical user interface instead of a textual (programming) interface.

Behind the end-user programming interface there is also middleware for smart home spaces. However, most state-of-the-art middleware solutions are not yet friendly enough for end users to turn their needs into rules. They include Samsung SmartThings, Apple HomeKit (https://apple.com/ios/home/), Google Assistant (https://assistant.google.com/), Axeda Machine Cloud (http://ptc.com/axeda), Microsoft HomeOS (https://www.microsoft.com/en-us/research/project/homeos-enabling-smarter-homes-for-everyone/), Berkeley Building-Application Stack (Krioukov, Fierro, Kitaev, & Culler, 2012), Octoblu Integration of Everything (https://octoblu.com/), and Wyliodrin—The IDE for IoT (https://wyliodrin.com/). The aforementioned all require that an application provides a manifest to describe the desired services by specifying the corresponding sensors and actuators on a per-rule basis. There is a clear mismatch between users' ability to program and the current programming interfaces. An exception is the Wukong platform (Huang et al., 2015), which defines the abstraction of sensing and actuation to facilitate IoT programming, and, in effect, enables write-once-deploy-everywhere for proficient programmers. However, the imperative nature of current middleware solutions remains a major issue in end-user programming. Yet another problem in current middleware solutions is their inability to adapt to changes. IoT devices are rather unreliable by design, due to the requirements of low cost and low power consumption (e.g., battery power). Devices may even malfunction or be disconnected from the home network from time to time. Because smart-home programs should be deployed once and run forever, there needs to be a certain degree of redundancy in sensing/actuation capabilities.

Weaving Smart-Home Technologies into the Real Home

Programming the smart home is not a one-time task, but a continuous process usually undertaken by a single knowledgeable person—a guru (or hacker)—in the family. Mennicken and Huang (2012) describe four phases of growing a smart home: initial planning, preparing technical infrastructure, iterating until it fits, and reaching (temporary) stability. Woo and Lim (2015) investigated how participants in eight households in Korea

Mathias Funk is assistant professor in the Future Everyday group in the Department of Industrial Design, Eindhoven University of Technology, the Netherlands. He has a background in Computer Science and a PhD in Electrical Engineering. His research interests include complex systems design, distributed IoT applications, remote data collection, systems for musical expression, and design tools. In the past he has worked at the Advanced Telecommunications Laboratory (ATR) in Japan, at RWTH Aachen, at Philips Consumer Lifestyle, and at Intel Labs, Santa Clara, CA. He is also a co-founder of UXsuite, a high-tech spin-off from Eindhoven University of Technology.

Lin-Lin Chen is dean of the Faculty of Industrial Design at Eindhoven University of Technology. She is also a professor in the Department of Design at National Taiwan University of Science and Technology (NTUST). She received her BS degree from National Cheng Kung University in Taiwan and her PhD from the University of Michigan at Ann Arbor. She was dean of the College of Design at NTUST from 2004 to 2010, president of the Chinese Institute of Design from 2007 to 2008, and convener for the arts (and design) area committee of Taiwan's National Science Council from 2009 to 2011. She is the founding editor-in-chief of the *International Journal of Design* (SCI, SSCI, AHCI), current president of the International Association of Societies of Design Research (IASDR), and a fellow of the Design Research Society. Her research focuses on user–IoT interfaces, design innovation, and product aesthetics.

Shao-Wen Yang is a senior staff research scientist at Intel Labs, Intel Corporation. He received his PhD degree in computer science from National Taiwan University in 2011. He joined Intel Labs in Taiwan in 2013 as a resident scientist in the Intel–NTU Connected Context Computing Center and focused on research and development of Internet of Things technology and middleware. In 2016 he moved to California, and started to focus on the Internet of *Video* Things. He served on the technical program committee of the Design Automation Conference and has served as a guest editor for several international journals. His current research interests include various aspects of visual fog computing, especially ease-of-use frameworks for workload creation, and partitioning and orchestration of visual fog workloads. He has over 20 technical publications, and over 60 patents and pending patent applications.

Yen-Kuang Chen received his PhD degree from Princeton University, NJ, and is a principal engineer at Intel Corporation, Santa Clara, CA. His research areas span from emerging Internet of Things (IoT) applications to computer architecture that can embrace emerging applications. He is passionate about IoT and the smart home. He has more than 100 IoT devices in his home. He has 60+ US patents, 20+ pending patent applications, and 100+ technical publications. He is a fellow of the IEEE.

applied trigger–action style programming to create rules, and how these rules were adopted. They found that after the initial installation stage, the participants experimented with new rules, went through a process of incorporating the rules into their family daily routines, and finally reached a stage of either routinization or removal of unsuitable rules. Clearly, it is important to support all phases of the smart home development process, from initial setup to fine-tuning and adoption of smart home behaviors, and also to consider all family members including the passive users (Mennicken & Huang, 2012).

Mismatch between User Needs and Existing Tools

From studies on end-user programming in the wild (Demeure, Caffiau, Elias, & Roux, 2015; Mennicken & Huang, 2012; Woo & Lim, 2015) as well as exploratory studies into future smart-home programming (Bellucci, Vianello, Florack, & Jacucci, 2016; Ur et al., 2014), it emerges that simple trigger–action programming rules often cannot adequately express users' intentions and mental models. For example, end users usually do not think in terms of physical sensors, but of triggers at a more abstract level, e.g., activities or states. In an explorative study conducted by Ur et al. (2014), three categories of triggers were identified: (1) physical sensors (e.g., *motion is detected*), (2) triggers related to activities, locations, and states (e.g., *when no one is in the room*), and (3) fuzzy triggers that might involve machine learning (e.g., *when the water is too hot*). The latter two categories are not well supported by the current trigger–action-style programming tools. Another explorative study was conducted by Bellucci et al. (2016) using a toolkit (T4Tags 2.0) that contains tangible tokens. Common triggers that they found included *presence in a room*, *time*, *location*, and *environmental sensing*, while common actions included sending notifications or reminders, controlling lights, and controlling an appliance. *Time* was found to be an important trigger, but users were often not satisfied with the built-in temporal triggers (Demeure et al., 2015). To better support mental model accuracy, Huang and Cakmak (2015) recommended making a distinction between different types of triggers (states, events), as well as between different types of actions (instantaneous, extended, and sustained).

In order to work within the constraints of trigger–action programming tools, end users have to express the abstract triggers (e.g., whether someone is in the family room) by using simple rules with concrete sensors (such as whether specific physical motion sensors detect movements for a specific period of time). A *meaningful* smart-home behavior is thus broken up and described in several fragmented simple rules, which are linked to different physical sensors and actions. There are several problems with such ad-hoc solutions. First, without a connection to the original intention, each rule might not be easy to interpret or explain after a certain period of time. To manage the rules that they had created, end users in one study tried to associate meanings by adopting specific naming schemes (Demeure et al., 2015). For example, one participant named rules by noting the devices, the triggered actions, the destination state of the device, and higher-level goals, while another participant followed the schema DAYTYPE–ACTION–CONDITION where DAYTYPE can be weekday, weekend, or holiday. A better solution would be to offer support to capture this information, rather than relying on users' self-imposed naming schemes. Second, these simple rules are neither robust nor migratable, because they depend on the availability and functioning of specific sensors and actuators. If the specific sensor or actuator is not available (in a new household) or is broken (after some period of time), manual reprogramming of the rule is necessary, even if the original intention remains the same (see section on *First-Person Experience* below). HomeOS (Dixon et al., 2012), a smart-home operating system developed by Microsoft, addresses such issues by requiring that an application provide a *manifest* to describe the devices or services that it needs in order to function, providing a layer of abstraction above the physical devices.

Intentions and Conflicts

At a higher level, Mennicken, Vermeulen, and Huang (2014) identified three major challenges and directions for smart-home research—*meaningful technologies*, *complex domestic spaces*, and *human–home collaboration*—based on an extensive literature review, analysis of smart-home solutions, and filed studies. They argued that to achieve meaningful technologies it is important to support the *goals* and *values* of inhabitants. What are these goals and values of smart-home users? Based on need-finding interviews in the field, Takayama, Pantofaru, Robson, Soto, and Barry (2012) identified goals such as *peace of mind*, *ecologically conscious*, *saving money*, *entertain and impress others*, and *personalize the home*. They found that the satisfaction of *home automators* often comes from creating connections to the home and family members, rather than simply controlling the home. In another study, Mennicken and Huang (2012) summarized four main motivations for home automation: *modernization*, *positive feedback cycles*, *hobby*, and *saving energy*. The translation of such high-level needs into simple rules that connect rather crude sensors and actuators is extremely challenging.

Mennicken and Huang further noted that smart technologies could result in conflicting high-level goals; for example, providing *peace of mind* through remote connection to the home can conflict with *privacy* and *security* concerns. In addition, different household members might have conflicting values or interests over time; for example, between parents' *energy saving* and kids' *maximum comfort*. Not only do needs differ among different households and among different members of the same household, but also needs can be different for the same individual over time. In several aspects, they argued for the necessity to tag applications with high-level goals, and to provide high-level rationales behind automation tasks. All these factors point to going beyond task-level rules to better capture the intentions of the users. A common issue is conflicting rules that end users produce unintentionally (Woo & Lim, 2015). Unexpected results occur when rules that were created for different reasons become active for the same devices with conflicting instructions. Since there is currently no interface to help users specify how such conflicts should be resolved or give priority to certain rules, the users have to manually revise the

rules through trial-and-error processes to avoid possible overlap of activations. Hereinafter, we may refer to end users' *goals* and *intentions* interchangeably.

While supporting the intentions of end users in a smart home is important, it has long been recognized to be extremely difficult for a smart-home system to automatically infer users' intentions. As Don Norman (2009) commented: "If only the house could read the mind of its owner. It is this inability to read minds, or, as the scientists prefer to say, to infer a person's intentions, that defeats these systems" (pp. 121-122). There exists the problem of *mutual intelligibility* (Suchman, 2007, p. 80): not only is the smart-home system unable to decipher users' intentions, but also users are often left in the dark about what the system knows about them, how the system knows it, and what the system is doing about it (Bellotti & Edwards, 2001). One of the reasons why it is so hard for the smart home to understand users' intentions is because users can improvise—they can take *situated actions* (Suchman, 2007)—that are unpredictable from sensor data. A possible approach to resolving this intelligibility problem is to explicitly include people in the loop. Rogers (2006) argued for a shift from *proactive computing* to *proactive people*. This paradigm shift is also reflected in the *interactive machine learning* process (Amershi, Cakmak, Knox, & Kulesza, 2014) whereby users iteratively provide information to a learning system, allowing simultaneous usage and development of the learned model.

Preferences and Overrides

While users have long-term goals and intentions, they might also have short-term needs and preferences for the specific needs of the moment. For example, Alan, Shann, Costanza, Ramchurn, and Seuken (2016) implemented three different types of thermostats—*manual*, *direct learning*, and *indirect learning*—that consider tradeoffs between comfort and price, and then evaluated the three designs in a field study with 30 UK families for over a month. In the *direct learning* design, they explicitly designed the system to allow users to temporarily override the predicted optimal temperature setting, after which the system would wait for a certain time before taking back control and setting the temperature back to the learned optimal temperature setting. Moreover, the researcher provided a *boost* option that directly changed the temperature setting without influencing the learned user's model. The results were generally positive in that the end users were happy with the thermostat autonomously responding to real-time prices on their behalf, and they felt in control. This in-situ study indicates that smart climate-control systems can operate largely autonomously from a good user model, and this is accepted by end users as long as real-time control is possible through preferences to allow for meaningful overrides and adjustments of system behavior according to ad-hoc needs.

Summary

The above review of studies on end-user programming in the wild and exploratory studies into future smart-home experiences shows that although trigger–action programming has gained popularity over the last few years (Ur et al., 2014; Ur et al., 2016), simple rules are neither sufficient to express desirable home behaviors, nor are adequate technologies available and accessible to end users. As Mennicken et al. (2014) argued, future smart homes need to support *higher goals* and *values* of the users. Consequently, without clearly capturing these, it is very difficult to orchestrate smart-home behaviors to support users' needs, to resolve desired behaviors towards the often messy physical graph, and to mitigate conflicts or interference between overlapping behaviors. As argued by researchers such as Rogers (2006), end users of the smart home should play a more active role. Instead of having a smart system that imposes an interpretation of the current contexts and user intentions based on sensor data, it might be better to provide an interface for the end users to express their intentions within a certain scenario.

First-Person Experience of Smart-Home Programming

Over the last three years, one of the authors has installed more than 150 heterogeneous connected devices in order to gain first-hand experience of the Internet of Things in an actual personal home context. Because of a move to a new house and the need to replace the main hub, the researcher completed three rounds of installation and reinstallation, offering a rare case study on smart-home experience beyond the initial installation. In the first part of this section, we will give a chronological overview of the three installations and noteworthy issues that emerged in continual longitudinal use. The second part summarizes and compares the user's intentions and the programmed rules in different categories.

Chronological Overview

First installation: At the beginning of the year 2014, the researcher began to install sensors and smart devices into his home. By the end of 2014, about 50 devices had been installed and more than 50 smart rules had been developed during this first phase of experience using mostly SmartThings and IFTTT. The initial installation of the devices and rules was more troublesome than the researcher had expected. The researcher frequently needed to reinstall devices to make adjustments. In addition, it proved very difficult to tune the sensitivities of sensors to achieve the intended smart-home behaviors. Oversensitivity creates many false alarms, while undersensitivity fails to trigger the actions properly. Both are undesirable. Unfortunately, today's feedback systems for adjusting and calibrating sensors are poor inasmuch as (1) they offer a limited input and output interface, (2) users sometimes need to make changes in the physical world in order to see the effect in the cyber space, and (3) feedback is time-delayed, e.g., cool-down periods for motion sensors make tuning difficult or at least time-consuming.

Second installation: Around the end of 2014, the researcher moved to a new home. Consequently, the researcher reinstalled all sensors and smart devices physically into the new environment, which had a different layout. The researcher found that it was better

to place the devices in the corresponding room, which was not always straightforward to achieve. Because rules are associated with the device IDs, if a sensor and its associated actuators are not properly labeled and moved accordingly, the motion sensor in the dining room, for example, may erroneously trigger the light in the living room to turn on. Many more difficulties were encountered during the second installation process: Some physical-environment-specific rules from the old house could not be easily applied in the new house, as the spatial relationships among rooms and also between devices and rooms were different. For example, in the first house (Figure 1), because the dining room is next to living room and already receives sufficient lighting, only when people are eating at the dining table should the light turn on, and not just when someone walks through. In the second house, the dining room is next to the family room instead of the living room.

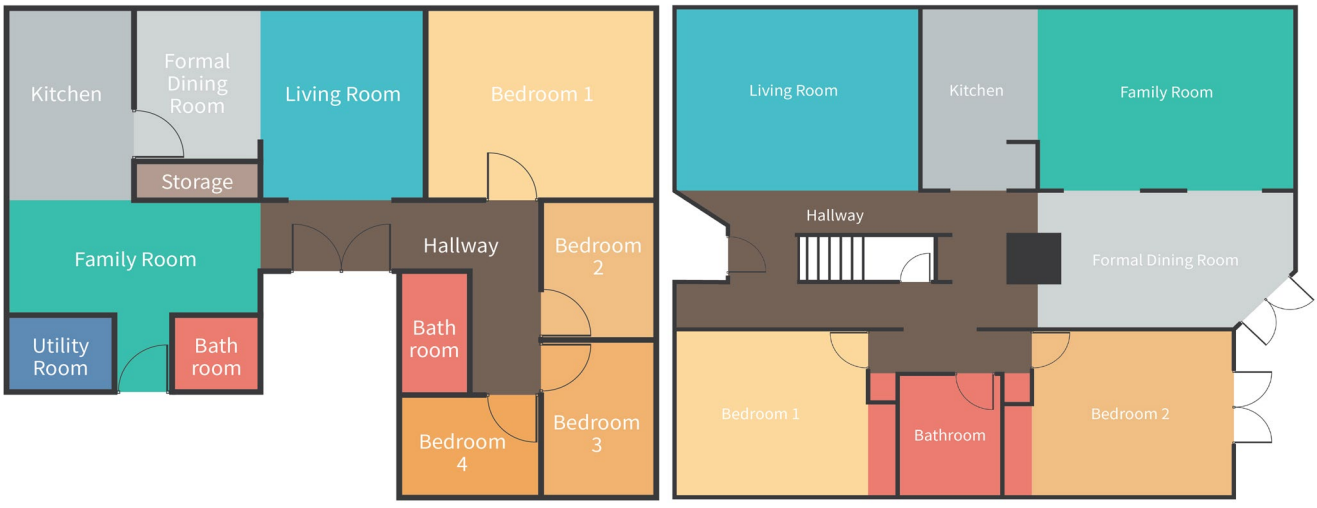
Third installation: In late 2015, the main smart-home hub was upgraded. However, because the new hub did not offer a good migration tool, the researcher ended up reinstalling all the devices to the new hub. By then, the family was used to the convenience of automatic behaviors, and would complain if the system failed to do things automatically. After two years, device failures started to occur. However, it was hard to debug a faulty device or system, due to the lack of interface support (e.g., display). In addition, it might be too tedious for smart-home users to deal with the faults of each individual device, often due to an insufficiently robust design that struggles with interruptions of wireless communication or battery power.

Additional installations: From late 2015 to late 2016, 50 more devices and 50 more rules were added. There were additional points of difficulty when these additional devices and rules were added. It was often cumbersome to adjust some of the old rules as some new devices and new rules can easily interfere with each other. For example, a light was programmed to be turned on when motion started and turned off when motion stopped. Later, a second rule was added, to turn the light on when the door was opened and turn it off five minutes after the door was closed. There was an instance when the light was turned off while users were active in the room because a user had opened and closed the door five minutes previously. Looking at the rules independently, they made sense, but together, unwanted behavior would occur. Depending on user preferences, turning on lights could have priority over turning off lights, but current systems do not notify the user about a potential interference, nor do they allow one to specify priority easily. Furthermore, because more customized rules (through multiple cloud services, e.g., SmartThings, IFTTT, Stringify) had been implemented to integrate more devices into the system, it became very hard to debug a problem as an actuation could be triggered by more than three different cloud services.

In summary, the change from the first to the second installation was due to moving house, while the change from the second to the third installation was necessary because of hardware upgrades. Both are common aspects of contemporary lifestyles and help illustrate the everyday problems that arise in the smart-home domain due to incompatibility, hardware and software faults, and configuration or usage problems that result from flawed interfaces and inferior usability and design.

User's Intentions and Implemented Rules

From the chronological overview, several general issues around setup and configuration, and also around maintenance and updating, become apparent. These issues are related to hardware failures, upgrading, physical installation, logical mapping to space, and the complexity of scaling a smart-home installation leveraging different external services. In this second part, we focus on programming and how users' intentions were mapped to rules, aiming to show how sensible user needs and high-level requirements cannot be expressed in rules in state-of-the-art smart-home technologies.



(a) First house (b) Second house

Figure 1. Floor plans of first and second installation sites.

From our first-person experience of smart-home programming, we identified different types of top-level intentions. Table 1 shows user intentions as reported by the researcher, divided into four categories: (1) *sustainability* (by saving water, energy, or other resources), (2) *security and safety*, (3) *convenience*, and (4) *privacy*. Note that there are overlaps, e.g., between 1 and 3 in the sense that turning off power for devices under certain conditions serves the purpose of *saving energy* and does so in a highly *convenient* way through full automation.

Looking at the rules actually implemented in the researcher's smart home, as shown in Table 2, two categories were identified: (1) *automatic control* and (2) *automatic notification*. Although remote control and remote monitoring are technically part of smart-home systems, they are not considered here because they lack crucial aspects of automation and delegation of control. Due to the complexity of rules and interoperability problems between different smart-home systems and their counterparts in the cloud, we note that most rules in Category 1 connect isolated subsystems to implicitly minimize interference with other parts of the smart home, for instance, the rules about room lighting or heating. Category 2 rules simply connect sensors to a notification service that alerts the home owner. While this serves the purposes of security, and of indication and notification, the quantity of potentially captured events can result in too few reports and a lack of information, or too many and thereby information fatigue.

Summary

Comparing Tables 1 and 2 reveals clear differences between users' intentions and what could be realized using rules connecting sensors to actuators or information displays. Although the rules cover many aspects of a smart home, at a low level the particular technologies demark often insular solutions, as few solutions are capable of supporting more complex uses spanning multiple devices or rooms of the smart home. For example, Table 3 shows, for a specific smart object in the user's home, how different rules and intentions can be applied to a single object and how the rules and intentions are interleaved or even overlap. This gives rise to two potential problems: (1) technologically, some of the rules might become complicated to create, test, and maintain, and (2) it will be very hard for someone else to understand or debug a system just by looking at the rules. For example, why did we not automatically turn on the dining-room light, and why is the time-out period longer between 7 p.m. and 10 p.m. (a semantic problem that indicates a need for higher-level information).

Analysis of Users' Intentions and the Corresponding Rules

With current tools, end users have to first distill their intentions into a simple automation scenario and then implement this scenario using the tools available. In practice, many more problems and issues emerge for the curious and courageous practitioner: even a perfectly designed and fine-tuned smart home needs to evolve with changing user needs—as expressed through preferences and overrides, but facilitated through learning and continuous adaptation. At a higher level, this is enabled by learning about users' preferences instead of operational rules. Only when we abstract the sensory and actuation capabilities can a smart-home system leverage information about redundancy, conflicts, and interference while still allowing users to be in control.

In the following, we expand the problem space to match the requirement that ideally, intentions and preferences should be directly recognized and programmable in technology. The problem space can be classified into three main categories: sensing activities and scenarios (*what is happening?*), capturing intentions and preferences (*how would the user like the environment to be?*), and contextual actuation (*what should happen in the environment?*). We will outline the categories by characterizing them in terms of design choices, involved technology, and, if possible, deficiencies of employed solutions.

Sensing Activities and Scenarios

Related research shows that users want simplicity in programming, and expect that a system should be able to automatically deal with different situations in a smart home in a smart way. Unfortunately, what can be programmed with current state-of-the-art technology

Table 1. User intentions.

1. Saving Energy/Resources	*2. Security/Safety*
• Reduce energy usage by turning off power (light, HVAC, plug/switch) • Reduce irrigation water • Get notified when power or water is used	• Arm the house's security system • Close/lock the door • Turn on the security cameras • Pretend that we are still in the house • Alert neighbors and authorities (turn on siren) • Shut down the electric/gas/water system • Get notified when something goes wrong
3. Convenience	*4. Privacy*
• Turn on the light • Turn on the HVAC • Turn on plug/switch • Disarm the security system • Open/unlock the door • Wake me up at the right time • Get information for daily routine, e.g., today's weather or traffic conditions	• Turn off the indoor cameras

Table 2. User-implemented rules.

1. Automatic control	*2. Automatic notification*
• Turn on the lights in the front porch and driveway - when there is motion, between sunset and sunrise • Turn on the lights in the living/family room - when the door is opened, between sunset and sunrise - when there is motion, between sunset and sunrise • Turn off the lights in the living room - between 7 and 10 p.m.: after 15 minutes without motion - at other times: after 5 minutes without motion • Turn off the lights in the family room, dining room, kitchen, front porch, driveway - after 5 minutes without motion • Turn lights on and off throughout the house to pretend that we are still at home - when we are not home, between sunset and 10 p.m. • Turn on all lights (and even change their colors) - when intruders are detected • Turn on the siren - when intruders are detected • Turn off the power plug for the TV - when everyone leaves the house • Turn off the power plug for the office - when I leave the house • Turn off the HVAC in the living/family rooms - when everyone leaves the house - when everyone goes to bed • Turn off the HVAC in my office - when I leave the house • Turn on the HVAC in the living/family room - when I arrive home - when I get up • Turn on the HVAC in my bedroom - 10 minutes before I get up • Close/lock the door - 5 minutes after we leave the house - 5 minutes after we leave it open or unlocked • Open/unlock the door - when I am coming back • Turn on the security cameras - when everyone leaves the house - when everyone is sleeping • Turn off the cameras - when I am coming home - when I wake up • Arm the security system - 5 minutes after everyone leaves • Disarm the security system - when any household member arrives • Reduce irrigation water - when it rains outside • Charge the electric vehicle - between 11 p.m. and 7 a.m.	• When the siren goes off • When a door is opened, and I am not home • When motion is detected (PIR sensor or video camera), and I am not home • When CO/smoke is detected • When CO_2 goes above a certain level • When water leakage is detected • When it is time to water/fertilize the garden • When the sprinkler is on • When the security system control box is moved or disconnected • When someone rings the doorbell • When I forget to close the door • At 6 a.m., the weather forecast and traffic condition • At the end of each month, the energy usage analysis

Table 3. User-implemented rules on a single object (example)**.**

Things	*Rules*	*Intentions*
Light in living room	Turn on, when there is motion in the same room, between sunset and sunrise; Turn on, when the door is opened, between sunset and sunrise;	**Convenience**
	Turn off, 15 minutes after the motion stops, when it is between 7 p.m. and 10 p.m.; Turn off, 5 minutes after the motion stops, when it is between 10 p.m. and 7 p.m.;	**Saving energy/resources**
	Turn on/off at random times, when owners are not at home, between sunset and 11 p.m. Turn on, when intruders detected	**Security**

still falls far short of such expectations. Here, we propose to distinguish between an *activity* and a *scenario*; the former refers to a sequence of user actions with a clear purpose, while the latter connects such activities to a context (or location) and goes beyond human activities. For example, reading is an activity, whereas reading in the bed and reading in the study are two different scenarios with probably very different intentions—trying to fall asleep, or not. Likewise, the end of an activity can be expressed as a scenario to allow for automation or notification. For end-user programming in the home, scenarios are more relevant and probably more stable over the long run, whereas activities without contextual information might be prone to many variations. Users in a smart environment may have complex needs that correspond to similar, yet slightly different scenarios. To take smart lighting in the living room as an example:

- When users are only walking through, they need minimal light, probably from a low light source.
- When users are chatting, possibly with guests, they need medium to maximal light.
- When users are watching TV, they need medium-to-low indirect light.
- When a user is reading a book, he/she needs maximal light, but in the form of a spotlight.
- When a user is reading from an illuminated tablet, he/she only needs low light.
- When users need to find lost keys or clean up spilled juice, they need maximal light, likely coming from the ceiling.

Thus, more and more scenario variants can grow out of an initially simple scenario of providing light in the living room. It is very difficult for conventional rule-based trigger–action programming to address the complexity of these scenario variants. A first reason is that scenarios and activities are not captured as such, but only as sensor-based triggers, which makes reuse difficult (or results in cloning triggers that might then interfere). Second, it requires complicated imperative rules to describe the scenarios, and complexity increases exponentially when new scenarios inevitably need to be added over time. What is needed is the ability to explicitly define a scenario and to allow the scenario to be fine-tuned, amended, or branched into new ones over time. We envision that a better user interface needs to be developed to allow multiple input combinations of scenarios mixed with user intentions.

Capturing Intentions and Preferences

Intentions can be seen as the long-term perspective of what we consider rationally the right things to do. They are expressed and formulated by end users in moments of clarity and family consensus, and from a more distanced view than usual daily life. For instance, New Year's resolutions carry traces of intentions and could be named intentions if they are accompanied by stronger commitments. In smart-home programming, intentions are rarely formulated explicitly. They are implicit to the planning and design of a smart-home system, and often surface only as generic intentions, such as saving resources, protecting life and assets, being in control, and enjoying convenience and comfort. Intentions can, however, be very instrumental for resolving conflicting rules in a meaningful way. For example, a common rule for smart lighting is to *turn on the light when the PIR motion sensor detects activity* and to *turn off the light when the PIR motion sensor detects no activity*. The intentions behind these rules might be convenience and sustainability. When the rules are isolated and the intentions are implicit, it is difficult to make proper trade-offs. By explicitly capturing the intentions behind the rules, it becomes possible for automated systems to reason about meaningful trade-offs for the users.

Moving from the long term to the short term, user preferences are an important concept to capture desired changes in automation—ad-hoc deviations that demark the flexibility of living spaces, give end users a strong sense of control over the automated system, and can be a source of data that the automated system learns from to adapt over time. With user preferences, conflict resolution becomes ever more complex. Not only do multiple users have conflicting preferences, even an individual user can have conflicting preferences with respect to long-term goals (e.g., losing weight) and short-term desires (e.g., ice-cream now!). In the smart-home setting, whether to open the blinds or to turn on the lights involves the conflict of whether to prioritize the intention of saving energy or that of privacy and security. Different users could also have conflicting perspectives: For the husband, turning off the lights means saving energy; for the wife, keeping the lights on makes her feel safe. There is clearly a need to record and learn users' preferences and intentions.

Contextual Actuation

In addition to sensing and triggering conditions, smart-home programming also needs to address smart actuations that can adapt from one scenario to another. It is desirable for smart-home programming to be able to be reused across like scenarios, becoming a template and allowing for adaptations during deployment. It is desirable to open up the possibility of fulfilling users' needs by different means. This can be achieved by separating the means from the objective. For example, the objective of *Get me some light* can be instrumented as the means of *Lights on* or *Blinds up* depending on the time of day, energy saving possibilities, and privacy concerns. What are needed are device-independent actuations, decoupling the physical layer from intentions. All of these occur when the physical system and system complexity grow over time with new devices, component replacements, upgrades, and upscaling in the smart home. Based on the aforementioned abstraction, smart-home programs should be able to adapt to changes by remapping the logical actuation to the currently available physical actuators. Finally, more subtle qualities of automated behavior, such as dimming instead of switching, perhaps depending on certain times when the users would like fewer disturbances, are very difficult for end users to program using current technologies, especially in a conditional or parameterized way.

Summary

The overview of the different areas of the expanded problem space shows that automated living and work spaces such as the smart home pose challenges that are neither fully understood, nor solved in practice, let alone in commercial applications of end-user programming systems. When end users program rules, they specify which concrete behavior is desired when a concrete event occurs. This is, however, insufficient for addressing the daily behavior of end users and their needs over time in the broader context of everyday life. Consequently, the possibilities for algorithmic decision making or even machine intelligence are very limited. Even if a user can program, writing trigger–action rules for hundreds of devices at home is a daunting task, potentially amplified by a collection of different platforms and interfaces. Current programming interfaces do not allow users to program at a proper level of abstraction and they inhibit reuse of de-facto-similar rules across devices, scenarios, and platforms (e.g., smart lighting rules that differ little from smart climate rules except for the particular sensors and actuators to be used). At the same time, low-level programming such as trigger–action rules depreciates fast when devices are added or upgraded, or simply fail the test of time. In the following section, we develop the expanded problem space towards a richer domain model to capture the right information.

Towards Capturing the Right Information for the Smart Home

We envision much better informed and much more capable home programming environments in which users should be able to delegate most automation functionality to smart, underlying control systems while retaining a certain level of control over elements that are critical to their experience. Ideally, users should be able to specify their intentions at a high enough level, so that smart-home solutions can figure out the right actuation means based on predetermined user preferences and machine learning.

To progress in this domain, we propose two radical measures: first, better capturing of information about scenarios and intentions, and second, the creation of feedback loops between interaction and captured information that facilitate adaption and learning over time. Covering both measures fully is beyond the scope of this article. However, we will address the first part in the following by introducing novel concepts for designing smart homes that lay the informational foundations.

Domain Model and Concepts

The system envisioned above operates on a *domain model* that takes users' needs as input, and then learns and adapts accordingly. Compared to direct mappings between sensors and actuators, i.e., concrete trigger–action rules (cf. Figure 2, left), a domain model consists of concepts that are more abstract, but at the same time more powerful as a structure to gather information about the context of the smart home (as a system), the context's users, and their intentions. In short, we aim at capturing and modeling information for designing a smart-home system, by using the following concepts (see Figure 2, right):

- ***Scenarios*** determine the current state of the smart-home system by abstracting from sensors in the environment towards states, i.e., what is happening in the world or what is currently the state of affairs;
- ***Intentions*** determine abstract behavior of the system and the general end-user priorities for realizing this behavior, i.e., which behavior of the smart home is desired, or what should happen given the current state of affairs;
- ***Preferences*** allow for ad-hoc control by end users in the environment by parameterizing behavior from a user interface, i.e., what small changes need to be applied to the way the system acts right now;
- ***Actuations*** discover and determine the right action, i.e., how a combination of states, intentions and preferences can result in the desired changes in the environment by using a set of suitable direct actuators such as lamps, audio devices, and locks.

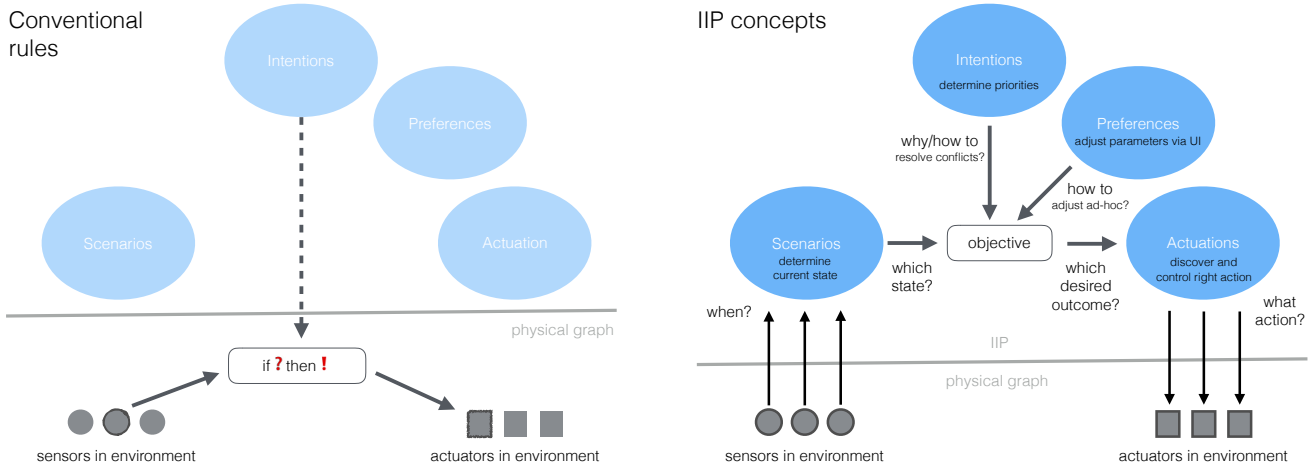


Figure 2. Comparison of conventional rules (left) **and new domain model with concepts** (right)**.**

While the concept of actuations is essential to the operation of a smart-home system leveraging the domain model, elaborating on the details of actuation is beyond the scope of this article. In the following, the first three concepts—scenarios, intentions, and preferences—will be explained with regard to their properties, how they differ from conventional end-user programming concepts, and how they relate to each other.

Scenarios

A scenario connects a context (e.g., a location) to sensor data indicating activities or system states. This means that scenarios represent an abstract combination of sensors measuring real-world activity in the moment and scenarios can therefore be understood as temporal states of a system. Common example scenarios in the smart-home domain are (1) *family on holiday*, (2) *evening*, and (3) *multiple family members in the living room*. These examples show that scenarios are combinable (e.g., 1 + 2 or 2 + 3), and can even form contradictory combinations (1 + 3) that need to be resolved by the system at a later stage. A scenario is activated when (sensor) event data (see below) matches the conditions of the scenario, such as the time of day being between 18:00 and 23:00 (activates Example Scenario 2), or motion sensors recording the number of people entering a room (activates Example Scenario 3). Such system states can possibly be specified with arbitrary granularity; however, most use cases only require states that last from minutes to hours (Examples 2 & 3), or in exceptional cases, days (Example 1). Likewise, the granularity of spatial locality can vary: scenarios can be linked to particular parts of the overall home context, such as rooms, corners, and working places; they can also extend further than the actual home to cover office or school presence (all depending on having relevant sensor data available) as long as they sufficiently describe relevant temporal states that can in the end be linked to meaningful system activity.

Compared to rule-based programming styles that are prevalent in the IoT domain, the use of highly semantic scenarios and the ability to represent and leverage more complex combinations of events and event data allow users to formulate semantic information about their everyday life in a more sophisticated way. Whereas rules can only capture device-specific data and connect them in simple ways, the domain model allows for a hierarchy of sensors to drive scenario activation from the highest semantics available. Conventional rules can capture conditions at device level efficiently, but are rather difficult for end users to translate towards activities—and such information is necessary for the smart home to act smart. Scenarios are constructed from semantic sensor data, which are not necessarily bound to specific sensors and therefore allow scenarios to be more portable, durable and robust over space and time, whereas rules break as soon as a specific device breaks or is temporarily unavailable. In summary, scenarios abstract from raw data, combine data towards the activation of a temporal state over a local context, and give meaning to this state.

Intentions

End users in a smart-home context usually have diverse and more or less well-articulated needs. Such needs can be very specific, but commonly follow a general notion of *quality of life*. Different aspects of this form a general set of intentions, such as convenience or comfort, saving resources (time, energy), feeling in control, feeling safe and secure, or aiming to avoid distractions or promote social interaction. While intentions are certainly all desirable (to differing extents), they are quite general and have a guiding function of abstract behavior. They can be internally resolved towards *objectives*, as shown in Figure 2, which are not directly exposed to the end user and match the physical graph with its actuators in the home. As stated above, intentions represent our "good," rational thinking that forecasts and plans. Capturing these thoughts is a novel approach and can often only be achieved through more complex interaction, examples that inspire, or defaults and templates that can be personalized. Figure 3 shows an example of two parallel intentions, *save energy* and *convenience*. The first intention has a higher priority than the second, which is internally resolved to two objectives, *switch off lights* (i.e., deactivate all lights, corresponding to *save energy*) and *welcome light* (i.e., activate a porch light, corresponding to *convenience*).

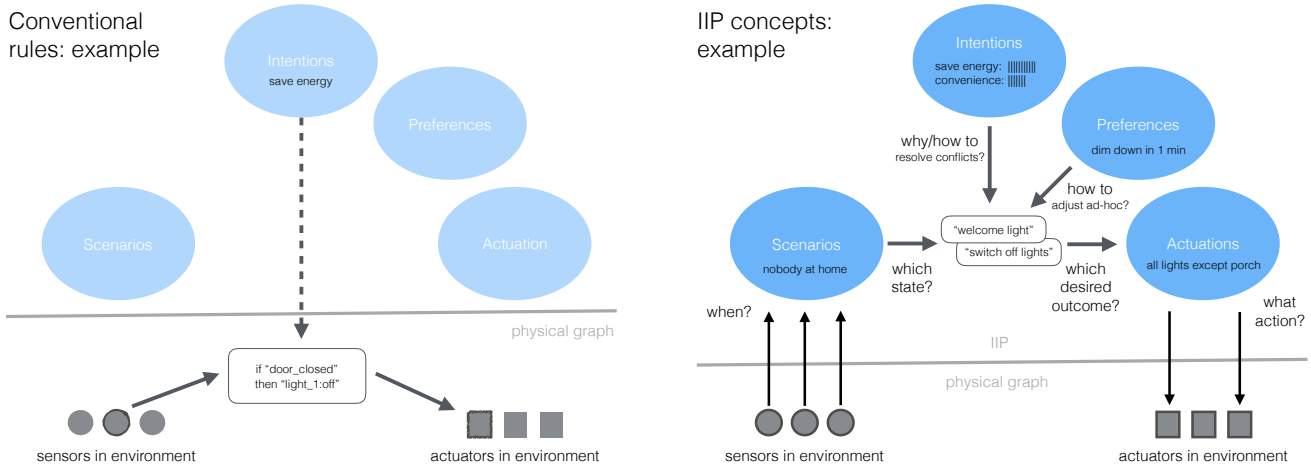


Figure 3. Comparison of lighting examples using conventional rules and domain-model concepts.
The right-hand part shows how parallel intentions are prioritized and resolved to parallel objectives and matching actuation.

Preferences

Still, everyday life will not always follow our plans, or guidelines derived from plans. Hence the counterpart to intentions is *preferences*. In contrast to long-term, rational thinking as represented by the concept of intentions, short-term preferences are crucial to capture user needs in the moment as shown by the study of different thermostats (Alan et al., 2016). Preferences essentially fit intentions to everyday life and shape how intentions are translated into actuation. Neglecting preferences would cause rapid ageing of the system's operationalization of intentions, and the initial fit with everyday life would be lost. The concept of preferences would be ideally mirrored by simple, targeted interfaces that allow users to tweak system behavior in the moment.

Preferences are not exclusively hedonic; they can be seen as real-time corrective measures from which the system can learn in order to adjust the way long-term intentions are translated into actuation. Preferences are therefore crucial (1) to tweak the system's behavior as a combination of intention-driven behavior and preference adjustments, and (2) to enable system learning and adaptation over time of intentions towards what inhabitants of the smart home actually prefer on a daily basis.

Chain of Behaviors

The above concepts form a chain from activity *sensing* to *scenarios* and system *states*, linking them to prioritized *intentions* and *preferences*, which finally determine *actuation* through *actuators*. Sensors and actuators, as low-level components, are omitted in this article. This chain is very different from other IoT programming approaches in the sense that it offers a way to express ambiguity and alternatives in intentions of one or more users, and to capture (ad-hoc) preferences in a consistent and highly semantic way. This is possible because this chain includes enough information for machine intelligence to reason and make decisions for the human users. Nevertheless, end users can interact with this intelligence through very context-specific and non-intrusive interfaces in the human environment. As an example, Figure 3 illustrates the simple case of light automation for saving energy: on the left-hand side, a rule is depicted that switches off the light when the door of a room is closed. On the right, a similar rule is shown with domain-model concepts that formulate a much broader use case, saving energy in the entire house, by switching all lights off in response to the absence of the inhabitants. This general behavior driven by the intention to save energy is mixed with a second intention, convenience, that will allow for all lights to be switched off except for the porch light, to allow returning inhabitants to safely enter the home. Although the concepts are individually not more complex than a simple rule, together they enable broader scopes of automation and intentionally programmed behavior.

Interactive Intentional Programming

In the first half of this section, we illustrated that a domain model can capture relevant information to address end-user needs for automation. In the remainder of the section, we will outline an approach to fill out and leverage this domain model: *Interactive Intentional Programming* (IIP). The approach essentially (re)structures the conventional *usage* phase into two sub-phases that correspond to domain modeling tasks performed in the respective sub-phases:

(1) In the *setup* phase, users specify (1) scenarios (high-level understanding of what is going on in the smart home) and (2) intentions for actuation (how the automated system should respond to active scenarios). This *intentional programming* part of IIP not only captures information about end-user intentions and preferences, but also aims at expressing how relevant particular intentions are in a certain scenario (prioritization).

(2) In the *use-and-refine* phase, users interactively fine-tune the scenarios and preferences through daily use. That is, in this *interactive programming* part of IIP, the domain model is meant to evolve dynamically in an interactive way as it explicitly allows for frequent re-specification of intent and the adjustment of preferences at any moment. With this plasticity, IIP caters for the needs of everyday smart-home usage, in which exceptions are the rule. That is, in IIP the domain model acts as a growing knowledge base that learns preferences from user interaction and determines how automation should react to sensed data and control actuation.

The nature of IIP is that it is essentially an *interactive* form of domain modeling that leverages formal concepts like *intentions*, but also *scenarios* and *preferences* for aspects of programming automation that are conventionally implicit or tacit.

In the following, we illustrate IIP with a first-person experience to demonstrate the *viability* of IIP through explicit modeling of scenarios, intentions and actuation.

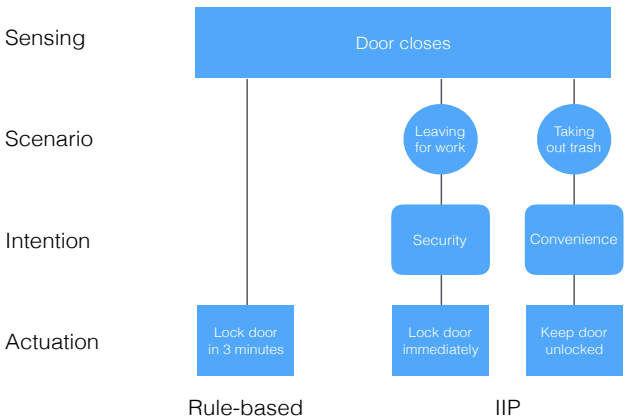


Figure 4. Scenario awareness (visualized as a change of behavior from sensing to actuation).

Scenario awareness is the capability of differentiating between a user's actions under a diverse set of scenarios. As illustrated in Figure 4, the apparent behavior of a user leaving the house (*door closes*) may be for work or taking out the trash can. In the scenario of *leaving for work*, the user has an intention of security, and accordingly would like the door to be locked immediately and securely; in the scenario of *taking out trash*, the user has an intention of convenience for leaving the door unlocked

briefly until the user gets back indoors again. The conventional rule-based approach will leave users no choice but to set up a fixed-interval timeout, e.g., three minutes, for both scenarios; as experience shows, our first-person user (see above) was locked out of the house from time to time with this implementation. IIP, on the other hand, makes it possible to differentiate between scenarios and then to generate actuations as fitting as possible.

Intention awareness is the capability of differentiating between intentions of one or more users, given one or more scenarios. In the left-hand part of Figure 5, one user is carrying out scenarios of (recipe) *reading* and *cooking* at the same time, with intentions of *comfort* and *health*, respectively; each of these intentions leads to two possible actuations. Being unaware of intention makes a smart-home system unable to address potential conflict in actuations, and it is apparent that IIP can differentiate among possible actuations and make a conflict-free actuation decision. The right-hand part of Figure 5 shows another example in which there is no apparent resolution to a conflict. There are two users, both in the scenario of *reading*. However, one user's intention is *privacy* whereas the other's is *energy saving*. Together, these intentions lead to conflicting combinations of actuations. Specifically, *privacy* makes sure the blinds are closed, whereas *energy saving* tries to keep the lights off; if both are implemented, the room will never have light in the nighttime. Intention awareness allows IIP to differentiate and resolve conflicts among intentions. For example, a conflict can be resolved by prioritization of users. If one user has greater authority than the others, that user's intention may override all others; conflict resolution can also be addressed by mitigating among multiple intentions.

Summary

The domain model and the IIP approach that we introduce here are responses to identified problems in the domain of smart-home automation. As explained above, we propose to tackle foremost the informational deficiencies of earlier approaches by capturing user needs as intentions and preferences, and users' everyday behavior as scenarios. The domain model concepts—scenarios, intentions, and preferences—allow for a domain modeling approach that provides high-level semantics to what a smart-home system should achieve in a particular context and time. Together, these concepts inform and resolve everyday situations in a chain of behaviors, through scenario and intention awareness. This can be leveraged in novel smart-home systems that reason based on the provided input aggregated in the domain model and, in addition, extend it by learning from everyday behavior and changes in preferences.

Discussion

In the previous section, we introduce a novel domain model with concepts that aim at developing the possibilities of programming the smart home through user-friendly declarative means, instead of low-level imperative rules that we show to be rather disconnected from user needs (cf. *Related Work* and *Analysis* sections). We also frame an approach, Interactive Intentional Programming (IIP), which relies on frequent interactions with end users to create and evolve concrete instances of the domain model for the particular contexts of the end users. The domain model captures information that allows an IIP system to become scenario and intention aware. It is important to note that the IIP approach purposely departs from conventional trigger–action programming and uses far more abstract means to capture information that is intended to be leveraged by advanced control systems and by a middleware layer that resolves semantics to concrete actuation. The scope of IIP is limited to the management of information flowing in and out of the domain model. Nevertheless, IIP clearly addresses the different challenges identified in the first part of this article: capturing activities and scenarios, eliciting intentions and preferences, and connecting them into a new form of actuation that leverages high-level semantics to solve common problems of rule-based trigger–action programming:

- Higher goals of strengthening connections among family members (Takayama et al., 2012) and seamlessly supporting family rituals (De Ruyter, 2003) are still beyond current smart-home solutions, but might come within reach with IIP.
- Conflict and interference of rules is an important aspect of smart-home automation that currently cannot be resolved within the realm of automation; it needs social interaction and mediation. IIP allows one to capture information at a high enough level such that conflicts become "visible" to the automated system and a resolution can be attempted,

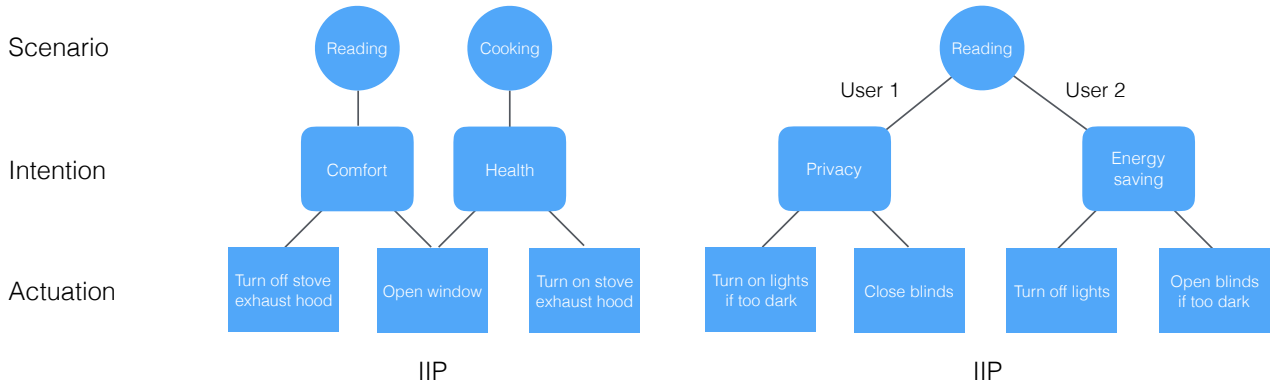


Figure 5. Intention awareness (visualized as a chain of behaviors omitting the sensing part); left-hand part shows intentions with overlapping actuation, right-hand part with conflicting actuation.

e.g., prioritizing different intentions such as *save energy* and *convenience* as shown in Figure 3 towards a setting that switches off all lights except a porch light for convenience.

- Independence of physical devices, i.e., IIP allows for more generalized smart-home programming that can adapt to different sets of physical devices and reroute actuation, e.g., light can be provided by different sources in the context depending on the preferences of different users that are present.
- Robust classification of human activities and system states is supported by IIP. For instance, referring to the three categories of triggers identified in Ur et al. (2014), the reported intentions mostly relate to activity triggers, but not yet to fuzzy triggers that require learning and adaption (e.g., user preferences under different contexts). In other words, scenarios can represent states and can help deal with more sensed complexity than can rules.
- Simple rules do not protect against information overflow and alert fatigue. As an example, a user wants to make sure that she receives information that guests are in her house. If she uses simple rules for that, sometimes she will receive a false alarm, or too many alerts in the case of a real emergency. In such cases, she would have to use multiple mobile apps to track down the root cause of the problem, i.e., debugging on the fly. In contrast, IIP allows one to specify intentions that control the amount of information that end users are exposed to.

It is important to note that using higher-level semantics and restricting access to IIP to largely declarative means may make end users feel they have less direct control over their home environment. This results from delegating real-time decision making largely to a control system that is instructed by scenarios, intentions, and preferences expressed in domain-model concepts. With conventional rule-based approaches, end users have to translate their intentions, combined with activity recognition through sensors, and encode them into atomic rules. In conventional rule-based systems, rules represent a direct connection between a sensor or event source, and an actuator that is triggered when the reading of the former meets a certain predetermined condition. End-user intentions are only indirectly represented in such rules, depending on the ability of end users to concretize their needs adequately. This is one of the most difficult tasks in programming, and inadequate training, expertise, or means may lead to conflicting, inconsistent, mutually interfering, or broken rules that ultimately do not satisfy user needs.

Finally, when we look again at the first-person experience of smart-home programming, there are also blind spots in the intended behavior, in the form of desired smart-home behavior that is not even considered relevant because its realization in rules is not deemed feasible or practical. A critical point for the first-person experience as shown above is that intentions and rules largely correlate and end users cannot really think of intentions before implementation. For the purposes of this paper, we had to reverse-engineer the intentions from implemented rules and since the implemented rules represent a subset of what would be possible with IIP, it is clear that intentions are underrepresented in everyday smart-home usage.

Conclusion

Designing for IoT and the use of connected products in the smart home is difficult, as related research shows. In this paper, we present additional evidence focusing on the long-term use of such technology as it scales, evolves, and continues to pose challenges for the brave practitioner. A central conclusion is that the temporal, preferential, technical, and social complexity of mapping end-user intent to rules needs new concepts to better frame information that needs to be captured to create smart-home systems that better match users' intentions. Based on the analysis of related work and an account of a multi-year first-person experience of smart-home programming, we identify two main areas for improvement: first, better capturing information about scenarios, intentions, and preferences, and second, the creation of feedback loops between interaction and captured information that facilitate adaption and learning over time. In this article, we argue for new concepts organized in a domain model and propose Interactive Intentional Programming (IIP) as a novel smart-home programming approach. The domain-model concepts are explained and contrasted against trigger-condition–action rules and similar end-user programming systems. The concepts are explained in context through several examples that illustrate how the challenges identified earlier can be successfully tackled with IIP. In the end, the proposed IIP approach deconstructs and rebuilds contextualized IoT programming with a higher level of abstraction that enables a substantial use of machine intelligence in the domestic environment, as a shift from directly programmed automation towards a better balance between the human and the system.

While research into the operationalization of the domain model and IIP is currently in progress, the concepts have matured sufficiently and will be supported by technical implementations in the near future. What also changes with approaches like IIP that leverage conceptually higher-level data is how we will design future smart-home products: we need to address different stages of technology adoption and use, we need to account for failure transparently, we need new interfaces to information and learning machines, and we need to carefully design the nuances of human–machine cooperation in personal environments. As a contribution to future design according to these principles, this article provides background, rationale, and conceptual building blocks.

Acknowledgements

This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 107-2633-E-002-001), National Taiwan University, Intel Corporation, and Delta Electronics. The authors would also like to thank Suzanne L. Thomas and Sangita Sharma for their great support and the many fruitful discussions.

References

1. Alan, A., Shann, M., Costanza, E., Ramchurn, S., & Seuken, S. (2016). It is too hot: An in-situ study of three designs for heating. In *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 5262-5273). New York, NY: ACM.

2. Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *AI Magazine, 35*(4), 105-120.
3. Bellotti, V., & Edwards, K. (2001). Intelligibility and accountability: Human considerations in context-aware systems. *Human–Computer Interaction, 16*(2-4), 193-212. doi:10.1207/S15327051HCI16234_05
4. Bellucci, A., Vianello, A., Florack, Y., & Jacucci, G. (2016). Supporting the serendipitous use of domestic technologies. *IEEE Pervasive Computing, 15*(2), 16-25.
5. Davidoff, S., Lee, M. K., Yiu, C., Zimmerman, J., & Dey, A. K. (2006). Principles of smart home control. In *Proceedings of the International Conference on Ubiquitous Computing* (pp. 19-34). Berlin, Germany: Springer.
6. Demeure, A., Caffiau, S., Elias, E., & Roux, C. (2015). *Building and using home automation systems: A field study.* Retrieved from https://hal.inria.fr/hal-01265223/document
7. De Ruyter, B. (2003). *365 days of ambient intelligence research in HomeLab.* Amsterdam, the Netherlands: Royal Philips Electronics.
8. De Ruyter, B., & Aarts, E. (2004). Ambient intelligence: Visualizing the future. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (pp. 203-208). New York, NY: ACM.
9. Dey, A. K., Sohn, T., Streng, S., & Kodama, J. (2006). iCAP: Interactive prototyping of context-aware applications. In *Proceedings of the International Conference on Pervasive Computing* (pp. 254-271). Berlin, Germany: Springer.
10. Dixon, C., Mahajan, R., Agarwal, S., Brush, A. J., Lee, B., Saroiu, S., & Bahl, P. (2012). An operating system for the home. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation* (pp. 25-25). Berkeley, CA: USENIX Association.
11. Huang, J., & Cakmak, M. (2015). Supporting mental model accuracy in trigger-action programming. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 215-225). New York, NY: ACM.
12. Huang, Z., Tsai, B. L., Chou, J. J., Chen, C. Y., Chen, C. H., Chuang, C. C., ... & Shih, C. S. (2015). Context and user behavior aware intelligent home control using WuKong middleware. In *Proceedings of the IEEE International Conference on Consumer Electronics – Taiwan* (pp. 302-303). Piscataway, NJ: IEEE. doi:10.1109/ICCE-TW.2015.7216911
13. Humble, J., Crabtree, A., Hemmings, T., Åkesson, K. P., Koleva, B., Rodden, T., & Hansson, P. (2003). "Playing with the bits" user-configuration of ubiquitous domestic environments. In *Proceedings of the International Conference on Ubiquitous Computing* (pp. 256-263). Berlin, Germany: Springer.
14. Intille, S. S. (2002). Designing a home of the future. *IEEE Pervasive Computing, 1*(2), 76-82.
15. Kientz, J. A., Patel, S. N., Jones, B., Price, E. D., Mynatt, E. D., & Abowd, G. D. (2008). The Georgia Tech aware home. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems* (pp. 3675-3680). New York, NY: ACM. doi:10.1145/1358628.1358911
16. Krioukov, A., Fierro, G., Kitaev, N., & Culler, D. (2012). Building application stack (BAS). In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (pp. 72-79). New York, NY: ACM. doi:10.1145/2422531.2422546
17. Mennicken, S., & Huang, E. M. (2012). Hacking the natural habitat: An in-the-wild study of smart homes, their development, and the people who live in them. In *Proceedings of the International Conference on Pervasive Computing* (pp. 143-160). Berlin, Germany: Springer.
18. Mennicken, S., Vermeulen, J., & Huang, E. M. (2014). From today's augmented houses to tomorrow's smart homes: New directions for home automation research. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 105-115). New York, NY: ACM.
19. Newman, M. W., Elliott, A., & Smith, T. F. (2008). Providing an integrated user experience of networked media, devices, and services through end-user composition. In *Proceedings of the International Conference on Pervasive Computing* (pp. 213-227). Berlin, Germany: Springer.
20. Norman, D. (2009). *The design of future things.* New York, NY: Basic Books.
21. Offermans, S. A. M., van Essen, H. A., & Eggen, J. H. (2014). User interaction with everyday lighting systems. *Personal and Ubiquitous Computing, 18*(8), 2035-2055.
22. Rogers, Y. (2006). Moving on from Weiser's vision of calm computing: Engaging ubicomp experiences. In *Proceedings of the International Conference on Ubiquitous Computing* (pp. 404-421). Berlin, Germany: Springer.
23. Suchman, L. (2007). *Human-machine reconfigurations: Plans and situated actions*. Cambridge, UK: Cambridge University Press.
24. Takayama, L., Pantofaru, C., Robson, D., Soto, B., & Barry, M. (2012). Making technology homey: Finding sources of satisfaction and meaning in home automation. In *Proceedings of the Conference on Ubiquitous Computing* (pp. 511-520). New York, NY: ACM.
25. Truong, K. N., Huang, E. M., & Abowd, G. D. (2004). CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. In *Proceedings of the International Conference on Ubiquitous Computing* (pp. 143-160). Berlin, Germany: Springer.
26. Ur, B., McManus, E., Pak, M. Y. H., & Littman, M. L. (2014). Practical trigger-action programming in the smart home. In *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 803-812). New York, NY: ACM.
27. Ur, B., Pak, M. Y. H., Brawner, S., Lee, J., Mennicken, S., Picard, N., Schulze, D. & Littman, M. L. (2016). Trigger-action programming in the wild: An analysis of 200,000 IFTTT recipes. In *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 3227-3231). New York, NY: ACM.
28. Woo, J. B., & Lim, Y. K. (2015). User experience in do-it-yourself-style smart homes. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 779-790). New York, NY: ACM.