homeBLOX: Introducing Process-Driven Home Automation

Michael Rietzler
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
michael.rietzler@uni-ulm.de

Florian Schaub
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
florian.schaub@uni-ulm.de

Julia Greim
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
julia.greim@uni-ulm.de

Björn Wiedersheim
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
bjoern.wiedersheim@uni-ulm.de

Marcel Walch
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
marcel.walch@uni-ulm.de

Michael Weber
Institute of Media Informatics
Ulm University
89069 Ulm, Germany
michael.weber@uni-ulm.de

UbiComp'13 Adjunct, September 8–12, 2013, Zurich, Switzerland.

ACM 978-1-4503-2215-7/13/09...$15.00.

http://dx.doi.org/10.1145/2494091.2497321

Abstract
Home automation promises more convenience for residential living. We propose process-driven home automation as an approach to reduce the difficulty of specifying automation tasks without restricting users in terms of customizability and complexity of supported scenarios. Our graph-based user interface abstracts from the complexity of process specification, while created sequences are automatically translated into BPEL code for execution. Our homeBLOX architecture extends a process engine with the capabilities to communicate with heterogeneous smart devices, integrate virtual devices, and support different home automation protocols. We report on initial user tests with our automation interface and demonstrate the customizability and expressiveness of our system based on realized example use cases.

Author Keywords
Home automation; UPnP; BPEL; Arduino; UCD; usability.

ACM Classification Keywords
H.4.m [Information systems applications]: Miscellaneous; H.5.2 [Information interfaces and presentation]: User interfaces.

Introduction

Home automation aims to enhance residential living comfort by interconnecting household devices. However, existing research and commercial platforms tend to be closed environments with specifically tailored devices or protocols. Most systems need to be set up by a specialist or require extensive tinkering by passionate users [11, 14], due to the heterogeneity of available devices and protocols. Current home automation systems face the dilemma of providing either usable applications with somewhat limited individual control or highly flexible but complex controls. We propose process-oriented home automation as an approach to facilitate usable, yet highly customizable home automation. Our homeBLOX system combines a modular architecture for flexible integration of heterogeneous devices and protocols with a graphical abstraction that simplifies configuration of home automation tasks as processes, while supporting complex scenarios that exceed existing rule-based approaches.

Related Work

Several home automation approaches focus on providing seamless interoperability. Busemann et al. [2] propose a flexible connector system for sensor nodes, where modular protocol adapters abstract from heterogeneous sensors. ATRACO [10] provides a unified interface for context from various sources. The GatorTech Smart House [4] encapsulates a home's sensor and actuator functions as services. HomeOS [3] also provides programming interfaces and services to manage and control a variety of devices. These approaches assist the development of versatile home applications, however, end users must hope that their needs are covered by these applications.

Other approaches aim to make smart home configuration usable for end-users. NinjaBlocks [12] provides a Web-based rule engine geared towards less tech-savvy users. Users can create a set of *if-this-then-that* rules that utilize events and actions offered by the user's devices. While this approach allows individualized home automation, use of many devices and automation scenarios likely results in a complex set of hard to maintain and potentially interfering rules. For the FHEM [8] home control server, multiple user interfaces exist (e.g., device lists, remote controls, device-floorplan models etc.), which demonstrate the diverse requirements and complexity of home automation interfaces. Kawsar et al.'s development tool [7] is a tangible user interface that consists of an RFID reader equipped with three buttons and LEDs. Each device and application has a corresponding RFID card. The user can place them on the reader and press a button to perform an action (e.g., install or uninstall, run or close). The tangible user interface has the limitation that the user has only the opportunity to use applications a developer has created. The Jigsaw Editor [5] uses the metaphor of puzzle pieces to shape linear sequences. While it enables end users to control their home automation intuitively, the linear approach is not flexible enough to create complex and context- sensitive home automation scenarios.

Process-driven Home Automation

While rule-based approaches to home automation facilitate straightforward specification of simple rules, more complex automation scenarios potentially result in difficult to maintain rule sets. Pipeline approaches, like Jigsaw Editor [5], lack the flexibility of working with larger numbers of inputs and outputs. Tangible configuration [7] facilitates management of smart devices, but is difficult to utilize for specifying complex automation tasks. In accordance with Kortuem et al. [9] and Kawsar et al. [6], we propose to use processes to model human activities. Processes have the advantage that they can model

Figure 1: Wizard to con gure the connection *Getting up* and *Co ee Maker*: event selection (top), action selection (middle), action con guration (bottom).

temporal dependencies: a single process can describe an automation sequence that combines events and actions of di erent devices at di erent steps. Thus, context information is aggregated over the course of a process rather than having to describe the complete situation in a rule s condition. While processes have the potential to simplify home automation, the user interfaces for con guring process engines, such as intalio bpms,[1] are notoriously complex. To simplify process speci cation, we developed a graph-based process representation in which each node constitutes an action of a device, and every edge a triggering event. Wizards assist users in the creation of automation sequences to reduce di culty of process de nition. Our interfaces and interaction concept were developed in an iterative user-centered design process. User-speci ed processes are translated into BPEL code[2] and executed by a process engine, which interacts with home automation components. Our system further abstracts from the heterogeneity of di erent home automation protocols. Protocol-speci c controllers forward incoming events from connected devices to the internal middleware and trigger actions on them. Light-weight drivers ease con guration when integrating new devices and abstract from speci c protocols supported by a device. Such drivers describe a device s capabilities and ensure consistent representation of devices in the user interface. The homeBLOX architecture consists of a middleware server for device management and process execution and a set of smart devices placed in the user s home. A tablet application enables users to manage automation sequences and interact with the system. We rst discuss the design and functionality of the sequence editor before outlining the middleware components and process execution.

[1] http://www.intalio.com/products/bpms/overview/
[2] BPEL = Business Process Execution Language

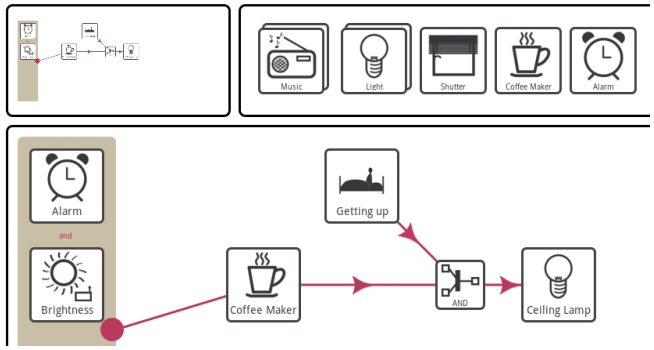

Figure 2: Sequence editor with an example sequence for automating co ee making and lamp control when getting up.

homeBLOX Sequence Editor

An Android-based tablet app provides the homeBLOX user interface to create and manage automation sequences (see Fig. 2). In the sequence editor, the user creates sequences by dragging available devices, represented as block icons (called *blox*), from the top tool bar onto the canvas and drawing connections between them. The canvas has a dedicated start area on the left to place the triggers of a sequence, as user testing revealed initial confusion about how to organize start conditions. When multiple blox are placed in the start area, the user is asked to choose the start semantics (logical *AND* or *OR*). Drawing a connection line between two blox on the canvas prompts a wizard, shown in Figure 1, to con gure the connection. The wizard guides through at most three steps: event selection (ES), action selection (AS), and action con guration (AC). Only relevant steps are shown, e.g. ES is skipped for connections originating from the start area, as triggering events are already de ned there. AC only appears if the selected action has con guration parameters, e.g., the number of cups a co ee maker

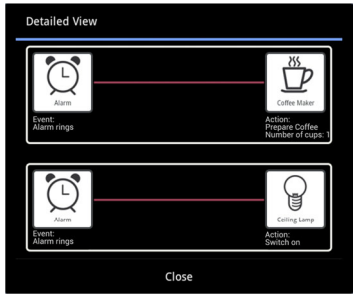

Figure 3: Detailed con guration dialog for the alarm clock and its outgoing connections.

should brew. When a user draws multiple incoming links to the same blox, two execution semantics are possible. Therefore, the system asks the user if the blox action should be executed when *any* of the incoming edges have been activated (*OR*), or only if *all* incoming edges have been activated (*AND*). A blox corresponding to the user s choice is added automatically to the canvas (see Fig. 2). Expert users also have the option to explicitly choose a logic operator from the tool bar and drag it to the canvas. Furthermore, already created sequences may be re-used as subsequences to encapsulate recurring behavior. The process representation in the sequence editor enables users to quickly grasp the purpose and overview of a process. To ensure clear arrangement of even complex processes, we refrained from directly visualizing selected events, actions and action parameters in the visual presentation of sequences. Users can review and edit the con guration of edges by clicking on any device icon on the canvas. Figure 3 shows the respective con guration dialog.

Initial User Testing

The development of the sequence editor was interleaved with user testing sessions for incremental re nement of the interaction concepts. We recruited 4 novices with little to no touch experience and 4 expert touch users as sample groups. None of the participants had any experience with business process modeling. At the beginning of each session, participants were asked to freely explore the interface and given as much time as needed to feel safe navigating it. Afterwards, they were presented with 9 tasks related to sequence creation and 1 sequence recognition task. Participants were asked to think aloud during task completion and the sessions were video recorded. Subsequently, we conducted a semi-structured interview to elaborate on di culties and user needs. User testing revealed, amongst other aspects, that users employ diverse strategies to create sequences. Some participants started by con guring a sequence s trigger events, while others rst dragged all needed blox onto the canvas, or began with the sequence s outcome. As a consequence, our sequence editor does not presuppose either strategy to equally support users regardless of their entry point. Participants also appreciated the redundant implementation of features, e.g., removing blox by dragging them into a displayed garbage can or o the screen. It also became apparent that users have a high demand for assistance and support during sequence creation, which however should not crowd the interface or impede the user s interaction ow. Users found line and icon annotations helpful, however they wanted them to be displayed only after explicit request. Three older participants (1 novice, 2 experts, aged 50+) helped to identify a number of age-dependent requirements. They voiced a need for adjustable font size, zoom capabilities, and help tutorials that let users proceed at their own speed. We plan to address these issues in future iterations to provide an inclusive user experience.

Home Automation System

Our home automation system consists of a server that interacts with heterogeneous smart devices.

homeBLOX Server Architecture

To integrate and control devices and to execute sequences, our server consists of several layers, as shown in Figure 4. The *controller layer* is responsible for communication with smart devices and forwarding their messages to higher layers. The *integration layer*, represented by the *Driver Manager*, manages registered devices and obtains matching device drivers, In addition, a database stores available knowledge about devices, deployed controllers, and the user-speci ed automation sequences.

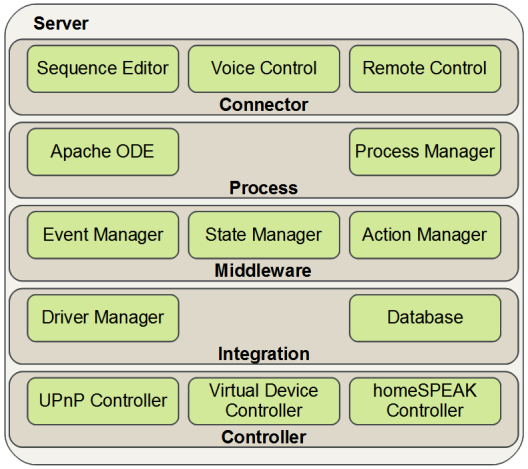

Figure 4: Architecture of the homeBLOX server.

Our *middleware* uses the information gathered by the integration layer to abstract from implementation details and unify different protocols and devices for the higher layers. The heart of the *process layer* is an Apache ODE process engine, which executes automation sequences. The *Process Manager* assists the process engine by starting and automatically restarting completed sequences and translating the user-created, graph-based process representations into executable BPEL code. The *connector layer* provides a control interface, which can support multiple connectors for communication with GUIs or alternative interaction means, e.g., voice control. When a new device is set up in the home, it can connect to the homeBLOX server via WiFi. A controller corresponding to the device's protocol (e.g. UPnP) handles communication with the device on the controller layer. Based on the device's name, the Driver Manager obtains and loads a driver that governs how the device is represented in the sequence editor, e.g., by defining human-readable names for the device's events and actions. Various state variables can be assigned to a defined range of values by using comparison operators. This enables the user to choose a named event instead of adjusting values. It is also possible to define new actions in the drivers. These use the defined operations of a device but with preset values. Our middleware converts these specific action calls into action calls supported by the device, so that the controllers do not have to deal with it. Thus, our driver concept is very different from operating system drivers. Rather than enabling interoperability between devices, which is handled by protocol-specific controllers, our drivers ensure that devices have semantically meaningful representations to enhance user interaction. User-specified automation sequences are sent to the server's GUI connector by the tablet app. They are translated into BPEL code and deployed on our process engine. Since ODE uses only block-structured BPEL, the graph-based representation of the automation sequence has to be converted accordingly. For this purpose, we developed a BPEL template, which already provides meaningful BPEL structures and properties for the home automation domain, and is completed with the user-specified sequence without further need for BPEL-related user configuration. The template allows us to automatically set concepts like partner links, correlations, and other BPEL constructs. The translation process also automatically decides how to handle multiple outgoing connections from one blox. By default an *AND*-split is used, while an *OR*-split is only used if the chosen events' range of values are disjoint.

Figure 5 shows how a sequence is executed in general. Devices send updates of their internal state to a controller (1), which maps values of state variables to specific events defined by the device driver (2). The Event Manager aggregates events and triggers the process engine (3).

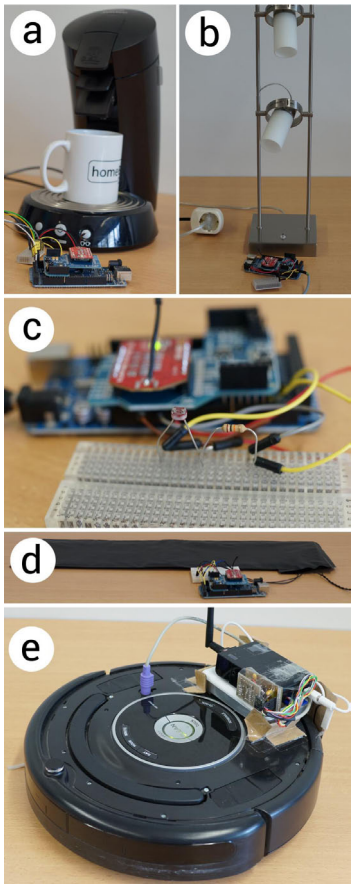

Figure 6: homeBLOX-enabled smart devices: (a) smart coffee maker, (b) smart lamp, (c) light sensor, (d) pressure mat, and (e) smart vacuum cleaning robot.

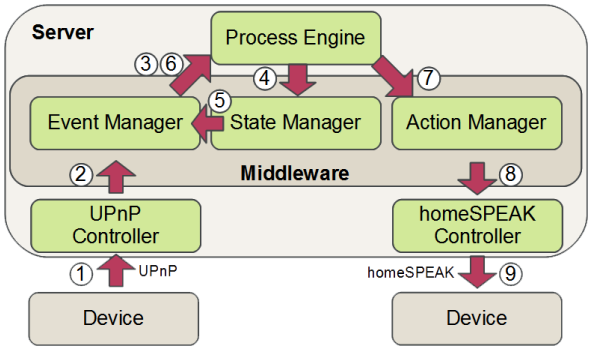

Figure 5: Execution of a home automation sequence.

Some events represent a device state (e.g. *light is on*). To make such states available to the process engine, the State Manager keeps track of the device state (4) and provides the last known values to the Event Manager (5), which can generate respective event notifications (6). The process engine evaluates processes to determine required actions. The Action Manager receives resulting action calls (7) and selects the responsible controller (8), based on the device's supported protocol. The controller then communicates with a specific device (9), which executes the action. The middleware itself is developed as a web service, which offers operations to control the system and receive information about registered devices. Thus, other applications can also benefit from the semantic device descriptions provided by our device drivers.

Smart Devices

Our homeBLOX testbed includes several types of devices, some of which are shown in Figure 6. A *UPnP controller* manages standard UPnP devices, such as an alarm clock or media player. The *virtual device controller* facilitates the use of online and system services, such as weather, time, or calendars. Such services are encapsulated as *virtual devices*. Virtual devices are applications that can be deployed directly on the homeBLOX server and subsequently register with the virtual device controller, which is implemented as a SOAP-based web service. After registration, virtual devices gather information from online APIs or the system and send notifications to their controller like real devices. Virtual devices can also execute actions, such as posting online or sending a tweet.

We further integrated a number of sensors (e.g., pressure mats, light sensors) and turned common household appliances (lamps and a coffee maker), as well as a cleaning robot into smart devices by equipping them with Arduino or Raspberry Pi controllers. These smart devices communicate with the homeBLOX system and can be used in automation sequences. Hereby, the actual device is treated as a sensor and/or actuator, whereas the WiFi-equipped Arduino handls event and action processing. In order to meet the limitations of the Arduino platform, we defined a lightweight communication protocol (*homeSPEAK*) for interaction with the server and added a corresponding controller. When a device is supposed to perform a specific action, e.g., *"make 1 cup of coffee"*, it receives a respective message from the responsible homeBLOX controller. The smart device has to translate the command into hardware control mechanisms. In case of the coffee maker, a physical button click is emulated via electric current. In case of our lamp, a WiFi-equipped power outlet is activated.

Example Use Cases

We evaluated two home automation uses cases with real devices in our testbed to demonstrate the expressiveness of process-driven home automation. The first use case focuses on getting up and is partially inspired by Weiser's *Sal's morning* use case [15], the second use case focuses

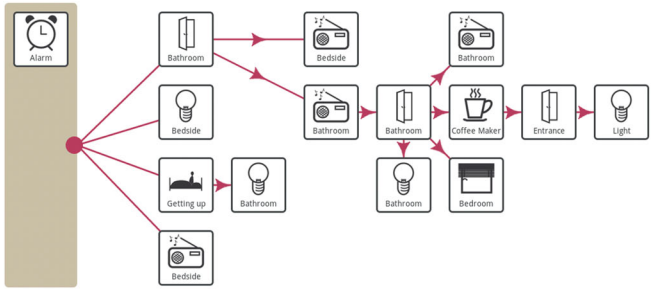

Figure 7: Automation sequence of the getting up use case.

on coming home. Note that each use case can be defined with a single automation sequence in the homeBLOX system, compared to multiple rules in a rule-based system. The respective automation sequences are shown in Figures 7 and 8.

Getting Up
When Birgit's alarm clock rings at 7am (*process trigger*), the bedside lamp and radio are turned on to help her wake up. When Birgit gets up (*pressure mat*), the bathroom is prepared by turning the lights on. When she enters the bathroom (*pressure mat*), the playing music follows her, i.e., the radio in the bathroom is turned on and the one in the bedroom is turned off. When Birgit is done with her morning routine and leaves the bathroom, the lights and radio are turned off. Furthermore, the shutters in the bedroom open and a cup of coffee is brewed in the kitchen, while she gets ready. Leaving the house (*pressure mat*) results in all lights being turned off to save energy.

Coming Home
Birgit programmed her vacuuming robot to clean while she is not at home. When she comes home (*pressure mat*), the robot immediately stops cleaning and returns to its charging station. If it is already dark outside (*light*

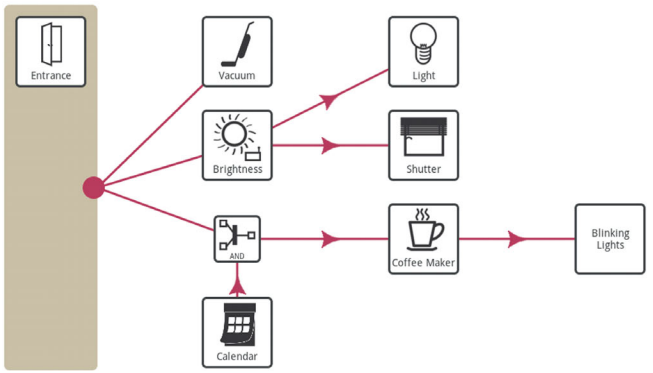

Figure 8: Automation sequence of the coming home use case.

sensor), the lights in her apartment turn on automatically. However, in case it is still light, the shutters in Birgit's apartment close instead. When coming home after work, Birgit prefers a cup of coffee, thus, her coffee maker brews a cup for her when she comes home on workday evenings. Birgit realizes her coffee is ready when the apartment lights blink shortly. This behavior is modeled by using predefined behavior in a subsequence.

Conclusions and Future Work

Process-driven home automation and the homeBLOX architecture facilitate complex home automation scenarios with heterogeneous devices, while providing a user interface that abstracts from underlying complexity without limiting expressiveness. Consistent representation of devices is based on drivers and protocol controllers, which support the clear representation of automation tasks as processes. In contrast to related work, configuration of homeBLOX is process-driven rather than rule-based. User testing in the iterative design process indicates that our sequence editor enables users to effectively create and recognize even complex automation scenarios.

We are conducting further user studies to evaluate the effects of our process-driven approach on home automation usability and refine our sequence editor. In accordance with related work [1, 11, 13], we plan on conducting home visitations to explore real-life use cases and automation processes. In addition, we are working on a community-based approach to build a repository of drivers, controllers, and virtual devices created by users or manufacturers. Further enhancements could be multi-user support and personal user profiles. Resolving conflicting sequences could be done with the help of dedicated room owners or prioritizing users and their sequences in the home. Deviations from activated sequences present further challenges in that they require a compromise between pervasiveness and user control.

References

[1] Brush, A. B., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., and Dixon, C. Home automation in the wild: challenges and opportunities. In *CHI '11*, ACM (2011).

[2] Busemann, C., Gazis, V., Gold, R., Kikiras, P., Kovacevic, A., Leonardi, A., Mirkovic, J., Walther, M., and Ziekow, H. Enabling the usage of sensor networks with service-oriented architectures. In *MidSens '12 workshop*, ACM (2012).

[3] Dixon, C., Mahajan, R., Agarwal, S., eBrush, A., Lee, B., Saroiu, S., and Bahl, P. An Operating System for the Home. In *NSDI '12*, USENIX Assoc. (2012).

[4] Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. The Gator Tech Smart House: A Programmable Pervasive Space. *Computer, IEEE 38*, 3 (2005), 50–60.

[5] Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.-P., Koleva, B., Rodden, T., and Hansson, P. Playing with the bits: User-configuration of ubiquitous domestic environments. In *UbiComp '03*, ACM (2003).

[6] Kawsar, F., Kortuem, G., and Altakrouri, B. Supporting interaction with the internet of things across objects, time and space. In *IOT '10*, IEEE (2010).

[7] Kawsar, F., Nakajima, T., and Fujinami, K. Deploy spontaneously: supporting end-users in building and enhancing a smart home. In *UbiComp '08* (2008).

[8] Koenig, R., Haas, M., and Droegehorn, O. FHEM. http://fhem.de/fhem.html.

[9] Kortuem, G., Kawsar, F., Fitton, D., and Sundramoorthy, V. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE 14*, 1 (2010), 44–51.

[10] Meliones, A., Economou, D., Grammatikakis, I., Kameas, A., and Goumopoulos, C. A context aware connected home platform for pervasive applications. In *SASO '08 workshops*, IEEE (2008).

[11] Mennicken, S., and Huang, E. M. Hacking the natural habitat: an in-the-wild study of smart homes, their development, and the people who live in them. In *Pervasive '12*, Springer (2012).

[12] Ninja Blocks Inc. The Ninja Platform. http://ninjablocks.com/.

[13] Poole, E. S., Chetty, M., Grinter, R. E., and Edwards, W. K. More than meets the eye: transforming the user experience of home network management. In *DIS '08*, ACM (2008).

[14] Takayama, L., Pantofaru, C., Robson, D., Soto, B., and Barry, M. Making technology homey: finding sources of satisfaction and meaning in home automation. In *UbiComp '12*, ACM (2012).

[15] Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (1991), 94–104.