Document downloaded from:

http://hdl.handle.net/10251/65115

This paper must be cited as:

Pons Tomás, P.; Catalá Bolós, A.; Jaén Martínez, FJ. (2015). Customizing smart environments: a tabletop approach. Journal of Ambient Intelligence and Smart Environments. 7(4):511-533. doi:10.3233/AIS-150328.

The final publication is available at

http://dx.doi.org/10.3233/AIS-150328

Copyright IOS Press

Additional Information

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

# Customizing smart environments: a tabletop approach

Patricia Pons[a*], Alejandro Catala[a] and Javier Jaen[a]
[a]*Grupo ISSI, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain*
*E-mail: {ppons, acatala, fjaen}@dsic.upv.es*

**Abstract.** Smart environments are becoming a reality in our society and the number of intelligent devices integrated in these spaces is increasing very rapidly. As the combination of intelligent elements will open a wide range of new opportunities to make our lives easier, final users should be provided with a simplified method of handling complex intelligent features. Specifying behavior in these environments can be difficult for non-experts, so that more efforts should be directed towards easing the customization tasks. This work presents an entirely visual rule editor based on dataflow expressions for interactive tabletops which allows behavior to be specified in smart environments. An experiment was carried out aimed at evaluating the usability of the editor in terms of non-programmers' understanding of the abstractions and concepts involved in the rule model, ease of use of the proposed visual interface and the suitability of the interaction mechanisms implemented in the editing tool. The study revealed that users with no previous programming experience were able to master the proposed rule model and editing tool for specifying behavior in the context of a smart home, even though some minor usability issues were detected.

Keywords: Rules, behavior definition, event-based systems, customization, smart environment, interactive tabletop, visual language

## 1. Introduction

Ambient Intelligence (AmI) [39],[47] has emerged as a computational field that can benefit different facets of our daily lives. New advances in AmI technologies deal with automation, transparency and adaptation [9],[18],[31],[46], as smart environments are increasingly provided with heterogeneous devices that need to be successfully integrated and interconnected [2],[14],[24]. The user's explicit interactions with the environment tend to be avoided in order to achieve ubiquitous solutions [34]. However, due to the great variety of possible situations, configurations and user demands, it is unlikely that developers can come up with systems capable of discovering the user's contextual preferences with a high degree of accuracy in all cases without any input from the user himself. The user's preferences should therefore form the key knowledge to be identified during the initial stages of the configuration. Other computer science areas consider the information provided by the user as a fundamental piece of their development process [43], and the AmI field is beginning to consider similar approaches. Recent studies within the AmI field are interested in determining the users' expectations of the future smart environments [3],[19],[25],[26]. Moreover, AmI systems are beginning to introduce implicit inputs to improve user satisfaction and customization, also known as *implicit human computer interaction* [36].

Considering the integration of explicit user specifications is also important because of the growing complexity of intelligent systems and the difficulty of automatically adapting them to many different contextual situations, so that it will be vital to involve final users in personalizing the behaviors/reactions of these environments. This is a challenging requirement as the personalization must be carried out by the final users of the system in a natural way, even if they have little or no programming knowledge.

[*] Corresponding author. E-mail: ppons@dsic.upv.es

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

Due to their reactive nature, smart environments often rely on event-based solutions. Event-Condition-Action (ECA) rules are usually preferred for these specifications [27],[40], and several studies reveal that ECA rules are easily understood by users with no programming knowledge [17],[22],[32]. However, the existing ECA rule-based approaches for specifying behavior either tend to minimize the expressiveness of the language in order to make it understandable to users or restrict highly expressive language to professional developers or skilled users. In addition, the scenarios in which ECA rules are involved rely on limited domain-specific ontologies, and thus restrict the specification to a few reactive and non-extensible elements, which prevents their use in real ambient intelligence scenarios in which intelligent devices are heterogeneous and constantly changing.

After studying the limitations of the existing rule models, we proposed a highly expressive and domain-independent language to define behavior rules and showed that this language is understandable by users without any prior programming experience [7],[33]. However, in the context of our proposed language, it remains to be seen whether adequate tools can be provided to support the specification of reactive environments. This paper makes manifold contributions in this respect. Firstly, it presents a tangible rule editor based on interactive surfaces offering natural interactions with fingers or physical objects. This editor allows rules to be specified in a wide range of expressiveness levels according to the users' knowledge and skills. Secondly, we provide a user study to validate the suitability of the proposed tangible mechanisms to define reactive behaviors in smart environments. Finally, we identify some of the problems that will need to be addressed in the design of future tangible tabletop-based editors for personalizing smart environments.

## 2. Related work

This section reviews existing studies that provide behavior specification mechanisms to customize ubiquitous systems according to users' preferences and motivates the use of a tangible-based user interface for the proposed customization system.

Dey et al. present *iCAP* [11], an informal pen-based tool for prototyping context-aware systems without coding, using IF-THEN rules as the behavior-specification mechanism in these environments. The rules defined with *iCAP* only allow conjunctions or disjunctions inside the conditional part of the rule – the antecedent. Inside these conditions only relational operators can be used. The action part of the rule, known as the consequent, only allows the conjunction of actions. This way, *iCAP* is intended to be easy enough to be used by end-users and expressive enough for developers who want to prototype simple context-aware applications. For this purpose, the authors highlight as future work that they would like to increase *iCAP*'s expressiveness and allow the edition of more complex rules.

The work in [8] presents a rule-based Service-Oriented Device Architecture (SODA) reactive system in which ECA rules are defined by programmers. This event-driven programming mechanism includes a time frequency modifier operator which allows the optimization of pull/push operations to retrieve data from sensors, thus, achieving important energy savings.

More expressive rules can be found in the work described in [4], in which IF-THEN rule constructions are extended with optional WHEN sentences, allowing the distinction between the event and the rule condition, which is more accurate for an ECA approach. In addition, OR-IF statements can be used to specify alternative behavior. Although this system still has not been evaluated, it is conceived to be used by non-expert smart home dwellers, as it uses a web interface based on drag-and-drop interactions to construct the rules.

García-Herranz et al. [16] created an application-independent programming system based on powerful ECA rules, which allows end-users to program complex behavior in ubiquitous environments. The system relies on a common kernel supporting the textual specification of highly expressive rules. García-Herranz also introduces the concept of timer, thus allowing the specification of rules to be executed before, during or after the specified time. Several user interfaces are implemented to adapt the system to different user backgrounds and situations. One of the proposed interfaces, not aimed at end-users, is based on traditional window-based controls and makes use of drag-and-drop metaphors to construct the rule. An interface more suitable for end-users was developed using tangible drag-and-drop interactions, but has only been tested with a limited ontology.

*Homer* [30] is a highly expressive work based on ECA rules. The conditional part of the rule allows the conjunction and disjunction of conditions and events, as usual, but it also facilitates the specification of timed sequences of events. In this case, events have to be raised in the specified order and within the specified time slot for the rule to be triggered. *Homer* offers two different user interfaces designed for tablets and smartphones, respectively, which allow the textual specification of behavioral rules. Despite the highly expressive language, *Homer* lacks the ability to define rules for groups of devices, and it would be desirable to have more flexibility when defining the actions to be performed, e.g.: being able to set numerical values as lighting options, and not just "*turn off*" and "*turn on*" actions.

*Gallag Strip* [23] is a programming by demonstration context-aware application for mobile phones. In order to define simple IF-THEN rules, the user has to start the "demonstration" mode in the application, where the system listens for sensor events triggered by user actions. When the desired event/action is recognized, the user can add it to the appropriate part of the rule, i.e., the antecedent part if it was an event or the consequent part if it was an action. Time and date restrictions can also be added manually to a rule. However, the programming by demonstration methodology has some disadvantages dealing with the wide range of possibilities of smart environment scenarios. It does not allow the specification of abstract concepts or hypothetical situations such as "*when there is a fire at home*", limiting the system to just the automation of known and reproducible tasks.

Díaz et. al. present a visual rule definition language prototype for smart environments [12]. This visual language will incorporate semantic mechanisms and fuzzy logic in order to define imprecise rules involving related or contextual devices/services. The antecedent of the rule is formed by conditional expressions that instantiate the scenario and the entities involved. The actions to be performed consist of the combination of services invocations using conjunctions, disjunctions, loops and similar control sequences. Although its visual interface is conceived for novice users, there is still no evidence supporting this claim as it has not been implemented yet.

Proprietary solutions as *Cortexa*[1] or *Control4*[2] offer customization systems restricted to household domains and only provide a limited variety of actions. In addition, open source initiatives as *NinjaBlocks*[3] or *Twine*[4] have arisen, but they still offer simple functionality and require the explicit configuration of each sensor added to the network.

[1] Cortexa Automation: http://www.cortexaautomation.com/
[2] Control4: http://www.control4.com/
[3] NinjaBlocks: https://ninjablocks.com/
[4] Twine: http://supermechanical.com/twine/

One of the common goals all these customization systems aim to reach is being usable by end-users, who usually do not have any programming background. Thus, proposals as programming by demonstration tools, visual languages or activity learning systems have become really helpful. However, there are limitations with these kind of customization mechanisms. Firstly, not every situation can be reproduced for the system to record it. Secondly, increasing the expressiveness tends to increase the complexity of the user interface. Finally, behavior recognition systems act as black boxes and their decisions can hardly be explained, which annoys the user. The necessity for a comprehensible and useful behavior editing tool arises. This editing tool should allow an easy customization of the system in the majority of situations in which the previous mentioned systems would cause difficulties to the user.

Aiming at easing the interaction and understanding of the system, new advances in the Human-Computer Interaction field have been considered. Tangible User Interfaces (TUI) have arisen over the last years as an appealing way for learning and interacting with computing systems [5],[20]. There have been several studies on how to objectively evaluate the benefits of TUIs for learning purposes [28],[29]. Also, research and studies on comparing classical UIs to tangible ones have been carried out in several areas with the main goal of facilitating learning mechanisms and the interaction with the system [21],[38],[42]. The comparative studies found in previous related works point out TUIs as an effective mechanism to captivate the users' attention and motivation [41],[45], easing the completion of hard tasks. Motivated by the advantages TUIs can offer, the editing tool presented in this work relies on a tabletop based user interface. In combination with a powerful behavior model and a comprehensible visual language, our customization system could have the highly desired balance between high expressiveness and usability. This paper presents our proposal for a tabletop-based rule editing tool and the usability study conducted. Thanks to this study, potential pitfalls have been detected and reported, which will contribute to improve the design of promising tangible editing tools.

## 3. Rule-based behavior specification with tabletops

In order to combine a comprehensible way of defining behavior with high levels of expressiveness we opted for a rule-based model enriched with dataflows, and a tangible editor for interactive tabletops based on the rule model. Both contributions are aimed at enabling users with little or no previous programming experience to specify reactive behaviors in the context of smart environments.

### *3.1. Rule model*

There already exists a wide range of AmI middleware [1],[13] that deals with the extreme heterogeneity of devices by defining proprietary ontologies. Instead of developing yet another ontology, we therefore adopted a generic and flexible abstraction based on categories/types and elements/entities that is able to integrate any specific ontology without affecting the way in which the behavioral rules are edited.

In this context, reactive environments, such as smart spaces in which the elements respond to users' inputs or events, are here considered as ecosystems. These ecosystems are populated with entities, which are an abstraction of the intelligent elements physically present in the smart environment. As elements of a smart space belong to a category or type of element, each entity in the ecosystem conforms to a specific entity type which determines both its behavior and features. For example, in a smart home the entity types would represent types of devices, such as *Television*, *Radio* or *AirConditioner*. Each entity type defines a set of properties and actions, e.g., an *AirConditioner* entity type would have several actions such as *TurnOn, SetFanSpeed, SetReferenceTemperature*, *TurnOff*, etc. and properties such as *ReferenceTemperature* and *FanSpeed* among others. Each instance of a specific entity type, e.g., entity *DiningRoomAC*, would be able to execute the actions defined by its type. However, the state of each entity of a given type, regarded as the value of its properties, would be different and would change during the evolution of the ecosystem.

As an intelligent space responds to changes in its environment, ecosystems evolve due to the occurrence of events. These occurrences conform to a given event type and are raised by the entities during their life cycle, e.g. every entity conforming to the *TemperatureSensor* entity type could trigger an event *TemperatureChanged* to announce changes in room temperature.

Taking these terms as basic building blocks, a rule is formally defined as an ordered pair R = <P, Q>, where P is the antecedent and Q is the consequent. The antecedent P, defined as P = (E, S, C), is formed by the occurrence of an event conforming to an event type E that has to be raised by a source population S, and a condition C associated to E and S. The consequent Q, defined as Q = (T, O, F, {DP}), is constituted by a target population of entities T, a condition F that filters the entities in T which are affected by the operation O, and a set of data processes {DP} indicating how the operation parameters are established before its execution.

Looking for highly expressive solutions, source and target populations in the rule can be specified in two different ways. Firstly, they can be specific entities in the ecosystem (e.g. *DiningRoomAC)* indicating an individual behavior that applies only to a given entity. Secondly, populations can be specified as a collection of entities of the same type, as several entities of a given type frequently participate in the same way in certain scenarios (e.g. all *AirConditioners* in a room). Conditions C and F are useful for refining these populations as required, instead of defining individual rules for each entity involved.

The semantics for a rule R is as follows: when an entity belonging to S raises an occurrence of an event E, and if the condition C is satisfied, then the rule is instantiated and activated. The operation O will be executed on those entities in T which satisfy the filter condition F. The operation parameters will be established regarding the specification of the data processes {DP}. A common behavioral rule in the context of a smart home is shown in Fig. **1** (a), representing the situation in which someone leaves the house and the alarm is effectively enabled. This rule should be edited without major concerns by novel users. The rule in Fig. **1** (b) exposes an elaborated rule, which uses generic populations and filtering conditions in order to define a single rule for a group of devices, which should act in the same manner under certain conditions. For defining this rule with previous approaches as the ones discussed in the related work section, many single rules to deal with several devices in the ecosystem would be required. Thus, our approach is more efficient and fits to the abstract reasoning that people usually perform when they talk about sets or collections of items. In addition, the hypothetic situation described in Fig. **1** (b) would be hard to define using programming by demonstration tools: David should be aware of Mary's intentions to call in order to start the demonstration mode of the editing tool.

Conditions C and F are *Boolean* expressions that can be represented as data processes, which define a visual representation of an assignment expression. The resulting *Boolean* value of these data processes is used to determine whether a particular condition

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

**WHEN S:** House **THROWS** an EVENT occurrence of type E: PersonLeaving
**THEN**
  **PERFORM**
    True → Alarm.Enabled

(a) Simple rule for activating the alarm when someone leaves the house

**WHEN S:** HousePhone **THROWS** an EVENT occurrence of type E: ReceivingCall
  [IF condition C:
  ReceivingCall.caller = Mary holds]
**THEN**
  **FOR_EACH** entity in T: Speaker
  [so that the filter condition F:
  Speaker.Location = David.Location AND
  Speaker.Volume < ( Speaker.VolumeMax / 2 )]
    **PERFORM**
      Speaker.VolumeMax * 75 / 100 →
      Speaker.Volume

(b) Elaborated rule for incoming calls, which turns up the volume of the speakers surrounding the receiver

Fig. 1. Text based rule example for a smart home.

holds or not. Moreover, the operation O of the rule is also specified by means of data processes. This operation can be either the assignment of a value to a target population's property or the execution of an action of the target population. In case of a property value assignment, only one data process is required whereas if an action is executed several data processes are required, one per parameter.

Data processes represent expressions visually and thus consist of several graphical elements. Firstly, a data process contains a collection of data sources, such as an event's attributes, entity properties or constant numerical values. Secondly, operators are required to transform data sources into new values. Finally, dataflow connections allow the application of operators to source data and their combination in subsequent operations. The final result of the entire expression in the data process may be assigned either to a property, an action parameter or a condition result. An example of the visual representation of an expression is shown in Fig. 2.

In previous work [7] we showed that dataflows provide high expressiveness and could be understood by users with non-programming backgrounds.

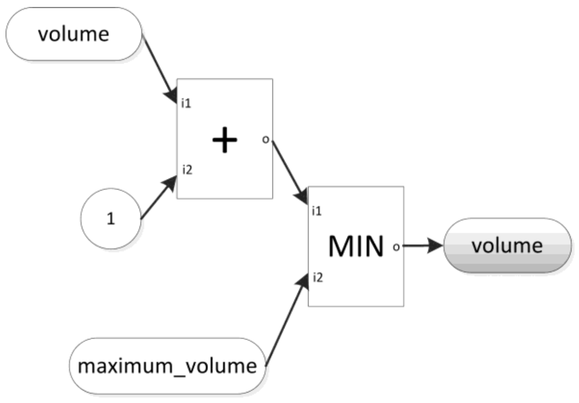

Fig. 2. A visual representation of the expression MIN(volume + 1, maximum_volume) to calculate the new volume of an entity.

### 3.2. *Rule editor*

The rule editor evaluated in this work relies on the rule model enriched with dataflow expressions presented above. As the editing tool relies on this generic and flexible model, it can be connected to any reactive environment that can be mapped to the model. The editor was conceived to work on interactive tabletops, which have touch input and related techniques suitable for editing dataflows. Exploring, selecting and editing by finger touch would make the system simpler for end-users. An important issue is that the concepts of the underlying model should be shown in such a way that users can overcome the inherent complexity of the model in an iterative and incremental process. With this in mind, the edition task was devised as a succession of decisions to be taken within different partial views of the rule available to the users. Once a partial view is shown, the users are supposed to use both finger-based input and specific tangibles to access the required collections and edit the data processes that compose the assignment expressions of the view in hand. Rules are saved throughout the entire editing process and also at the end of the edition, so users can review and reedit former rules to correct mistakes or adapt a rule to their current preferences. Also, experienced users can freely navigate from one rule to another and from view to view inside one of the rules, without following a specific editing order. Thus, both skilled and unskilled users could find the easiest way to configure their environment by adapting the editing tool to their learning curve. Given that a tabletop can serve as potential collaborative mediator, with the aim of

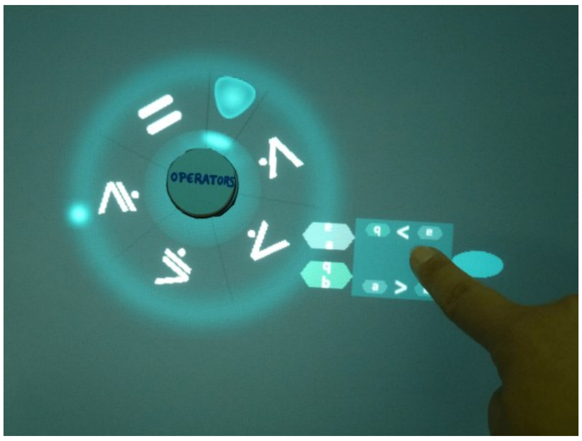

Fig. 3. Selecting an operator by exploring the operators' collection and dragging the operator into the canvas.

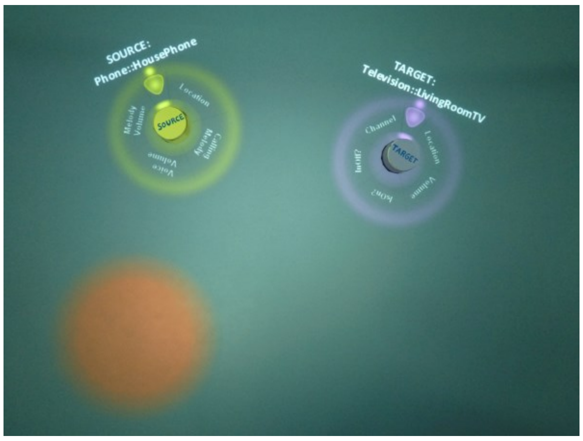

Fig. 4. Initial view of the editor: both source and target populations have been set, while the event remains unselected.

facilitating collaboration in future scenarios, texts can be read from two sides of the tabletop by mirroring them on an axis and using 360º controls. The main elements of the editor and just how the editing process is carried out are explained below. The editor was developed for a Microsoft Surface 1.0 Unit and was implemented using the Microsoft XNA Framework and Microsoft Surface SDK v1.0.

### *3.2.1. Exploring collections*

A generic control for collection manipulation on tabletop displays was used [6] to facilitate the task of exploring collections and selecting rule elements, such as entities, events, operators, and properties.

There are three collections from which to select the most important elements in a rule: the *source population*, the *target population* and the *event*. The elements in these collections are accessed in a structured way, meaning that there can be different collection levels. This leveled structure organizes elements in the collection and filters them according to users' selections.

When selecting source or target populations, users have to navigate through different collection hierarchical levels. For example, if the source population has to be the *DiningRoomAC* entity, users should first choose the source entities' collection, explore it to find the *AirConditioner* entity type and select it. Once the *AirConditioner* entity type is selected, the collection of *AirConditioner* entities is displayed. Finally, the user has to explore this collection and select *DiningRoomAC*. *DiningRoomAC* is then established as the source population. Likewise, users can select an entity type as a source or target population, meaning that all the instances of the selected entity type will be affected by the rule. For example, if the target population has to be all the televisions inside the house, users should choose the target population collection and explore it to select the entity type *Television* as the target. At this moment, all the television entities will be considered as target entities when the rule is triggered, with no need to define individual rules for each television in the house.

Once the event and the source and target populations have been selected, the associated collection will display the selection and it will remain fixed until the end of the editing process unless the user decides to modify the selection. As the users can make a mistake during the selection, and also experienced uses may want to redefine an old rule for a new purpose, modifications of the source, the target or the event of a rule once they have been already selected are allowed.

Two additional collections are used to explore operators and foreign elements, i.e. entities or properties which are not part of the source and target populations. The operators' collection displays a set of operators to be applied to source data. Operators are grouped by category: logical, relational, arithmetical, etc. The operators' collection is shown in Fig. 3, illustrating a user selecting a "*greater than"* operator and dragging it to the editing area. The foreign element collection contains all the entities in the ecosystem, and interaction with this collection is the same as with source and target collections. However, selecting an entity in the foreign element collection will not establish it either as the source or target population. The purpose of this collection is to allow any

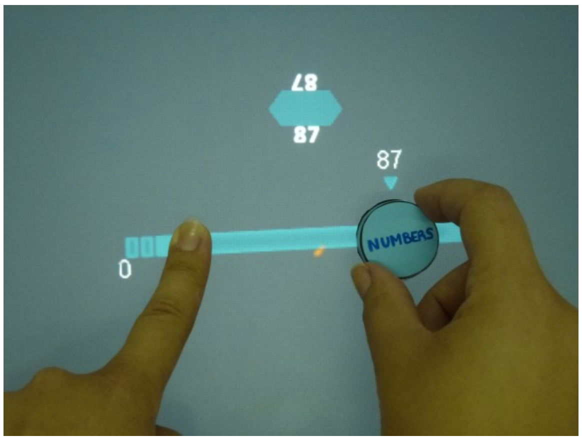

Fig. 5. Numerical setting control.

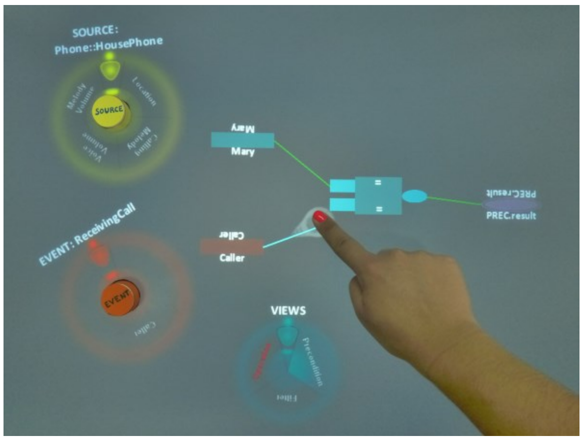

Fig. 6. Dataflow edition between an event attribute and an operator to form a condition data process.

entity or property to be used as a data source in a rule, e.g. comparing the *ReferenceTemperature* property of an entity *OfficeAC* with the new *Temperature* notified by the *OfficeTemperatureSensor*.

All the collections are accessed with tangibles, which can be dragged across the surface to move a collection to a different location. Moreover, users can display, hide or relocate any collection at any time during a rule's editing process.

### 3.2.2. *Views*

For usability reasons, rule editing was divided into four views. The initial view of the editor is given in Fig. 4, and shows the selection of the source, target and event. This first step is meant to introduce the user to the main rule elements and consists of three colored areas in which the user has to put a specific tangible to start the selection and pick the desired items. There are three special tangibles for this purpose, each of them attached to one of the required collections: source population, target population and event. When a user puts a tangible on the tabletop, the corresponding collection is displayed. The tangible can be rotated in order to display more elements in the collection, following the interaction techniques of the collection control. As explained above, each entity is only able to raise the events that have been defined within its type. To facilitate the correct selection of an event, the events collection is updated dynamically and only shows events that the selected source population can trigger.

Once these selections are performed, users can choose between the three remaining views, which deal with the condition, filter and operation processes and can only be specified or edited one at a time. This design decision forces the user to focus on a specific dataflow expression and is expected to reduce editing errors. In any of these three views, users have to edit the corresponding data process described in the formalized rule model of section 3.1: the condition view will include the definition of the conditions applicable to the source population and the event; the filter view consists of editing the conditions that target entities must hold; and the operation view will define the actions to be carried on those target entities fulfilling the filtering conditions defined in the filter view.

For editing the condition, filter and operation views, the user has to proceed in each of them as explained in the following section.

### 3.2.3. *Editing data processes*

Once the user has chosen to edit a view's associated data process, he can drag elements into the editing area and set up dataflows between them. Elements that can be connected with dataflows are generically referred to as *nodes*. Initially, the only visible element in the condition and filter views is the target node that represents the *Boolean* result of the data process, while in the operation view, the user has to select the required property or action of the target population.

Users may drag entity properties from collections to the workspace area, or just select a property from a collection and the property will be added to the data process. The same interaction mechanism can be applied to event attributes and operators. Numerical constants can also be added to a data process, for

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

which a special control has been considered (see Fig. 5). When the corresponding tangible is positioned on the surface, a slider control is displayed and the desired value can be established.

Creating dataflows between the elements in the editing area is simply a matter of drawing lines between the elements to be linked. A dataflow can only be drawn in its natural direction, i.e., from a data producer to a data consumer node, thus, avoiding syntactical mistakes. An example of dataflow edition is shown in Fig. 6.

The users are informed of any mistakes made in the dataflow connections during the editing process and are provided with hints to help solve them. This visual feedback is given in two different ways: some errors are displayed while the user is editing the data process; for example when a new dataflow is created, its color will show whether it has been correctly edited, red indicating an error (e.g. input and output value types do not match). Other mistakes are pointed out at the end of the process, such as cycle detection or disconnected node verification. The later kind of mistakes are pointed out by indicating in red the associated view name in the view collection, and by coloring in red the names of the affected nodes. For any mistake or modification, there exists a specific deletion tangible which can be used to delete both nodes and dataflows, restore collections to their initial state if any selections where done, and delete the source, target or event of a rule by touching the corresponding collection with the deletion tangible. Fig. 7 shows a completely edited data process without any mistakes in the editing process. Users can change from view to view whenever they need, and the modifications on each view will be saved for later review, as well as any mistake information feedback. A rule will become completely edited when the source and target populations are selected, as well as the event, and the condition, filter and operation data processes are correctly edited, i.e., with no mistakes. Once a rule is completed, it can be revisited later for further modifications.

## 4. Experimental evaluation

An exploratory test was carried out to evaluate the usability and adequacy of the developed rule editor. The trial involved a comparison of the process of defining reactive behaviors in the context of a smart home on two categories of subjects with different programming backgrounds. The aim was to find out whether users with no programming skills would have difficulties when specifying the reactive behaviors. The ease of use of the proposed visual interface and the suitability of the interaction mechanisms based on tangibles and multi-touch input were also evaluated, to determine whether the specification process would be negatively affected by issues at the interface interaction level and/or by problems related to the understandability of the underlying conceptual model.

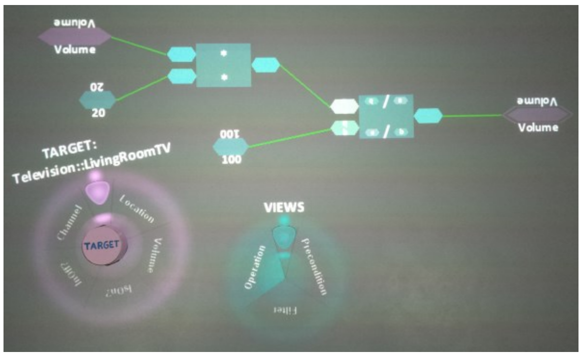

Fig. 7. Complete operation data process conforming the expression LivingRoomTV.Volume * 20 / 100 → LivingRoomTV.Volume.

### 4.1. Participants

Sixteen volunteers (11 male and 5 female) participated in the study. Eight had previous programming experience (P), as they were studying or working in the Computer Science field. The remaining eight subjects had no previous programming experience (NP) and heterogeneous backgrounds. The programmers' ages ranged from 22 to 36 (Mean (M) = 27.75, Standard Deviation (SD) = 5.676), and non-programmers' from 17 to 37 (M = 26, SD = 7.874). All the programmers reported using both personal computers and tactile devices every day. Seven non-programmers reported using personal computers every day and one almost every day. Five non-programmers used tactile devices every day, two almost every day, and one infrequently.

Only one of the participants (a programmer) rarely used interactive tabletops. The rest of the participants had never used an interactive surface before doing this test. All the participants had normal or corrected-to-normal vision and were right-handed, except for one non-programmer who was ambidextrous.

### *4.2. Method*

The participants were asked to use the rule editor proposed in Section 3.2 to complete a set of exercises individually. All the experiment and the exercises it comprises were constructed around a hypothetic smart environment model, which represented a common smart home and its commonplace devices (air conditioners, televisions, lights, etc.). Consequently, participants focused only on the editing assignments. This way, external noise is avoided and the experiment effectively analyzes the usefulness and comprehension of the tool. Previous to the edition with the tabletop-based tool, they were introduced to the basic concepts of reactive rules and behavior definition and performed a set of paper-based exercises to assess their understanding of the concepts. They were then introduced to the editing tool by associating the concepts with the elements in the editor and guided through a definition of a complete example rule to introduce them to the interaction mechanisms used in the tool. They were then asked to complete eight exercises in each of which they had to define a behavior rule using the visual editor. The rule specification of each exercise was given on paper in the form of the visual notation of generic data processes.

Rule difficulty varied among the exercises, and the sequence in which they were carried out was different for each subject, to eliminate order effects. Table 1 shows the difficulty of the rule proposed in each exercise, according to three dimensions: condition difficulty, operation difficulty and appearance of foreign entities (entities different from the source and target populations). A condition or operation was considered of low difficulty if the expression involved was as simple as a single comparison or a trivial assignment with only one operator, e.g., *Volume* – 20. Conversely, conditions or operations with high difficulty were those containing compound expressions, e.g. *Volume* > 20 AND *Volume* < 30. To simplify this classification, the difficulty of editing a data process was considered *high* if there were two or more operators involved. If there were one or more foreign entities *present* in the rule, its difficulty was considered *high* in this regard. Fig. 8 and Fig. 9 show two of the eight exercises presented to the participants using the visual notation they learned during the training.

Table 1

Rule difficulty dimensions.

| Exercise | Condition difficulty | Operation difficulty | Foreign entities |
|---|---|---|---|
| Exercise 1 | Low | Low | Missing |
| Exercise 2 | High | Low | Missing |
| Exercise 3 | Low | Low | Present |
| Exercise 4 | High | Low | Present |
| Exercise 5 | Low | High | Missing |
| Exercise 6 | High | High | Missing |
| Exercise 7 | Low | High | Present |
| Exercise 8 | High | High | Present |

The users' interactions with the rule editor were recorded and then post-processed for the extraction of relevant information for the statistical study. The sessions were also video-recorded for further analysis of the participants' behavior patterns. At the end of the session, the subjects were asked to fill in a ques-

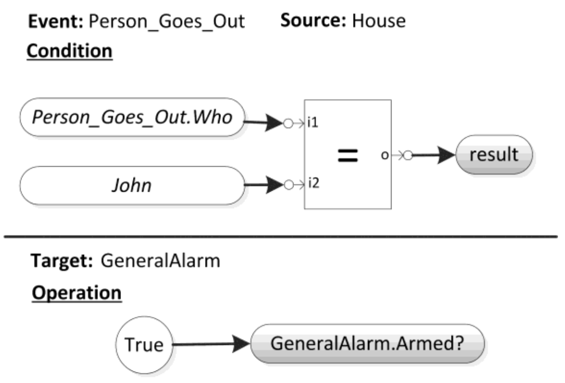

Fig. 8. Rule from exercise 3, representing the activation of the alarm when John leaves the house.

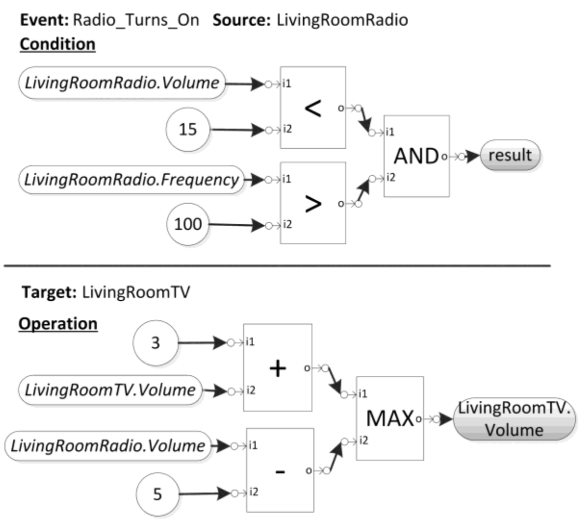

Fig. 9. Rule from exercise 6, representing the configuration of the volume of LivingRoomTV.

tionnaire in order to collate their opinions on usability of the system and their understanding of the interactions.

### 4.3. *Evaluation dimensions*

In order to evaluate the ability of the editing process to enable both programmers and non-programmers to produce behavioral rules, several dimensions of the process were studied:

- *Process efficiency*: the capability of the tool to enable users to effectively achieve their goals. This can be measured by several factors, such as time spent on each exercise, unsatisfactory explorations or incorrect actions which have to be undone.
- *Correctness*: the degree of correctness of the rules obtained after the editing process, according to the paper-based specification given.
- *Process layout management*: the suitability of the layout interface elements in the editing tool for the spatial distribution of elements and the movements that users perform to position them.

### 4.4. *Experimental results*

The following sections give the main results of the experiment concerning the previously defined dimensions for both groups of subjects and an evaluation of any significant differences between the groups. General conclusions on the editing tool's effectiveness in supporting behavior specification are also given.

#### 4.4.1. *Process efficiency*

The difficulty of the editing process, and thus the required level of expertise to carry it out, can be measured by means of several metrics, such as the average time subjects spend on each exercise, the number of fruitless explorations, or the number of times the deletion tangible is used to remove incorrect parts of the expression. The higher these scores are, the harder the editing process is considered. Interaction mechanisms should therefore be improved by offering users more intuitive and flexible editing methods if the scores are particularly high.

The time spent on each exercise and the time spent on each view of the rule editor were recorded, to collect evidence on which parts of the rule required most cognitive effort by the participants. Considering the whole session, i.e., the edition of the 8 assigned exercises, programmers spent an average time of 30 minutes and 28 seconds (SD = 2.81 minutes), while non-programmers spent an average time of 37 minutes and 40 seconds (SD = 5.37 minutes). These results were pleasantly surprising, as the whole session was expected to be completed in 45 minutes average time and both groups required considerably less time. According to the t-test performed, there is statistical significant difference between both groups of participants (t = 3.358, p = 0.005). Nevertheless, this overall performance should not lead to the idea that both groups behaved quite differently, as the cause for the statistical significant difference is due to just two of the exercises, while the other six remain with quite similar times. A deeper analysis of the time spent on each exercise is required, what can be carried out with this experimental design.

Table 2

Descriptive statistics for time (seconds) spent on each exercise grouped by category of subjects.

| Exercise | t | Degrees of freedom (df) | p-value | Category | Mean | Std. Deviation |
|---|---|---|---|---|---|---|
| Exercise 1 | 1.819 | 14 | 0.090 | NP | 113.011 | 23.613 |
| | | | | P | 96.471 | 10.190 |
| Exercise 2 | 1.352 | 14 | 0.198 | NP | 296.895 | 106.101 |
| | | | | P | 221.816 | 115.806 |
| Exercise 3 | 1.868 | 14 | 0.083 | NP | 151.684 | 44.816 |
| | | | | P | 118.680 | 22.101 |
| Exercise 4 | 0.723 | 14 | 0.481 | NP | 169.783 | 35.758 |
| | | | | P | 159.118 | 21.456 |
| Exercise 5 | 0.990 | 10.859 | 0.344 | NP | 376.355 | 176.877 |
| | | | | P | 305.764 | 96.963 |
| Exercise 6 | 0.757 | 14 | 0.462 | NP | 413.874 | 135.834 |
| | | | | P | 373.056 | 69.346 |
| Exercise 7 | 2.241 | 14 | 0.042 | NP | 378.510 | 133.836 |
| | | | | P | 265.731 | 48.469 |
| Exercise 8 | 2.512 | 8.668 | 0.034 | NP | 360.418 | 77.145 |
| | | | | P | 287.866 | 26.825 |

Table 3

Statistical analysis of the number of fruitless explorations for each type of collection.

| Collection | Mann-Whitney U | Z | p-value | Category | Mean | Std. Deviation |
|---|---|---|---|---|---|---|
| Operators Collection | 20.5 | -1.218 | 0.223 | NP | 9.75 | 2.816 |
| | | | | P | 8.00 | 3.817 |
| Foreign Element Collection | 29.5 | -0.272 | 0.786 | NP | 2.63 | 1.061 |
| | | | | P | 2.88 | 2.031 |
| Source Collection | 27.5 | -0.485 | 0.628 | NP | 1.63 | 1.685 |
| | | | | P | 2.00 | 1.773 |
| Target Collection | 21.0 | -1.272 | 0.203 | NP | 0.38 | 0.518 |
| | | | | P | 1.13 | 1.356 |
| Event Collection | 29.0 | -0.463 | 0.643 | NP | 0.25 | 0.707 |
| | | | | P | 0.25 | 0.463 |

In terms of the time spent on each exercise, Table 2 summarizes the average time and standard deviation for each exercise grouped by subject category. Independent t-test were run for each exercise with 95% confidence intervals for the mean difference, and assuming variance equality for exercises 1 to 4, 6 and 7 (df = 14). Levene's test for equality of variances showed this equality could not be assumed in exercises 5 (p = 0.014) and 8 (p = 0.011), so adjustments to the degrees of freedom were made using the Welch-Satterthwaite method in order to properly apply independent t-test for both exercises. The statistical results of the t-tests show that non-programmers average time is always higher for every exercise, although there are significant differences between programmers and non-programmers only in Exercises 7 ($t_{14}$ = 2.241, *p* = 0.042) and 8 ($t_{8.668}$ = 2.512, *p* = 0.034). A detailed statistical analysis of the different rule views revealed that these differences are related to the editing of the operation view of the rule, as the average time spent defining the operation on these rules differs significantly between the two participant categories: *p* = 0.008 for Exercise 7 and *p* = 0.033 for Exercise 8. This result can be explained by the difficulty level of the operation dataflows for these rules and by their specific cascade-like operators layout, such as the operation in Fig. 13 – (b).

Moreover, a repeated measures ANOVA has been performed in order to determine which of the three difficulty factors significantly affected the editing time. The condition difficulty (c), operation difficulty (o) and presence of foreign entities (f) have been considered intra-subject effects with binary values, and the category of the participants (programmers or non-programmers) has been considered as a between-subjects effect. The three intra-subjects main effects were statistically significant (p (c) < 0.001, p (o) < 0.001, p (f) = 0.027), so each of the three considered factors incremented gradually the difficulty of the exercises. Based on Cohen's Partial Eta Squared, it can be seen that the most important effect was the operation difficulty ($\eta_p^2$ = 0.883), followed by the condition difficulty ($\eta_p^2$ = 0. 666) and finally by the foreign entities presence ($\eta_p^2$ = 0.304). However, the interaction of each of the intra-subject factors with the category of participants was not significant in any case. This result means that each intra-subject factor affects similarly to both categories of participants.

After the statistical analysis, it can be seen that both user groups performed almost similarly when the amount of time spent on each exercise is considered. Only statistical differences were found for the two rules of greater difficulty, because, as expected, adaptation to new systems and interactions is usually slower for non-programmers. However, time results for both groups were satisfactory and improved the expectations.

The second metric in the process efficiency dimension is related to the exploration of collections by using graphic controls that provide 360º interaction mechanisms, based on tangibles for visualizing, positioning and exploring the contents of any existing collection during the edition process. The participants' explorations are occasionally unsatisfactory as they are unable to find the desired element. Fruitless explorations are those that do not lead either to the selection of the source, target or event item, or the extraction of a property or operator to the canvas. These explorations force participants either to use the deletion tangible on the collection or to lift and bring back the tangible from/to the surface to put the collection back into its initial state. Table 3 shows that there are no significant differences between programmers and non-programmers with respect to the total number of fruitless explorations for any type of collection, considering the non-parametric Mann-

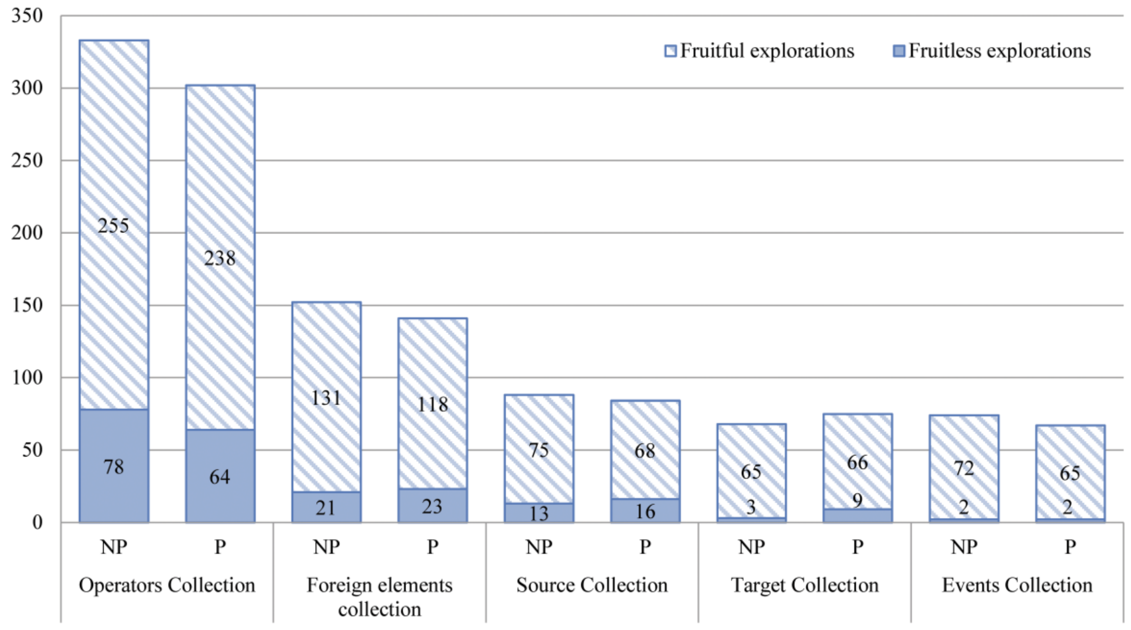

Fig. 10. Total amount of fruitless explorations for each type of collection in all tasks.

Whitney tests performed. Fig. 10 shows the total number of fruitful and fruitless explorations in comparison for each category of users and type of collection. Thus, it can be seen that fruitless explorations are almost inexistent in the source, target or event collections. The operators' collection seems to be the most conflictive one (66.67% and 56.14% of fruitless explorations for non-programmers and programmers, respectively). The operators' collection offers its elements initially grouped by category: logical, relational, arithmetic operators and selectors. Once the user has selected a category, operators belonging to that category are shown. The participants often failed to select the operator's proper category, so that a more intuitive and visual grouping method should be considered in future versions of the editor. Even with these difficulties, participants managed to explore the collection satisfactory in the vast majority of situations, as fruitless explorations with the operators' collection represent the 23.42% of interactions with this collection for non-programmers and 21.19% for programmers.

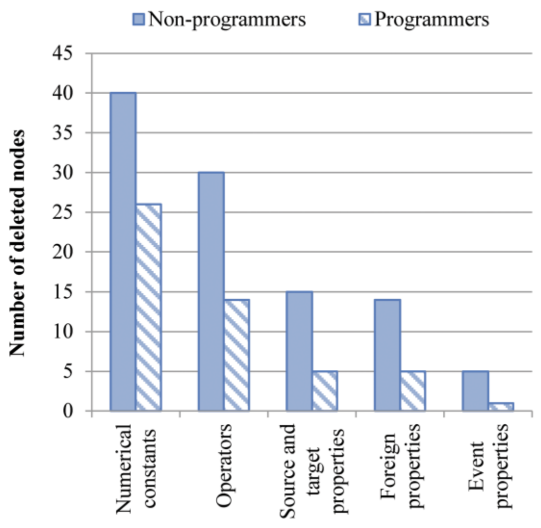

Fig. 11. Number of deleted nodes grouped by type of node.

The third metric considered in the efficiency dimension is the number of elements removed from the editing area, especially the types of nodes and dataflows connections removed during the editing process.

Fig. **11** gives the number of deleted nodes by type and shows that constant values come up as the most problematic type of node for both categories of users (42.58% of all deletions). The control associated with numeric constant insertion was shown to be unsatisfactory for many users. In this respect it can be concluded that, although non-programmers always per-

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

Table 4

Statistical analysis of the number of deleted nodes by type.

| Node type | Mann-Whitney U | Z | p-value | Category | Mean | Std. Deviation |
|---|---|---|---|---|---|---|
| Numerical constants | 23.5 | -0.904 | 0.366 | NP | 5.00 | 3.586 |
| | | | | P | 3.25 | 2.712 |
| Operators | 16.0 | -1.721 | 0.085 | NP | 3.75 | 2.493 |
| | | | | P | 1.75 | 1.282 |
| Source and target properties | 15.0 | -1.864 | 0.062 | NP | 1.88 | 1.356 |
| | | | | P | 0.63 | 0.744 |
| Foreign properties | 21.5 | -1.188 | 0.235 | NP | 1.75 | 2.188 |
| | | | | P | 0.63 | 1.061 |
| Event properties | 23.5 | -1.179 | 0.239 | NP | 0.63 | 1.061 |
| | | | | P | 0.13 | 0.354 |

form considerably more deletions than programmers, there are no significant differences in deletions of any type of node, according to the non-parametric Mann-Whitney tests performed (see Table 4). Additionally, the analysis of the deletions related to dataflow connections reveals that there are no significant differences between programmers (M = 0.45, SD = 1.402, Median (Mdn) = 0.00) and non-programmers (M = 0.34, SD = 0.996, Mdn = 0.00).

### 4.4.2. *Correctness of rules*

The edited rules were examined in order to determine whether non-programmers make significantly more mistakes than programmers. A deeper analysis was performed to determine the most frequently occurring errors and to provide reasonable solutions that would prevent these errors and thus improve the editing process.

Concerning the 8 exercises assigned to each user, non-programmers made a total of 17 mistakes (M = 2.13, SD = 3.271, Mdn = 0.50) in contrast to 7 mistakes by programmers (M = 0.88, SD = 0.641, Mdn = 1.00). Despite the high number of mistakes made by non-programmers, the non-parametric Mann-Whitney test shows there is no significant difference between both user groups (U = 32.00, Z = 0.00, p-value = 1.00), as most non-programmers' mistakes were made by two of the subjects (this will be discussed below). For non-programmers, the distribution of errors among the three parts of the rule consists of 4 errors in the event, source and/or target population, 3 errors in the condition and 10 errors in the operation. The programmers did not make any mistake establishing the event, source and target population, but made 6 errors on the condition and 1 error on the operation. A detailed classification of the errors made while editing data processes leads to 4 different mistake typologies: incorrect operator, incorrect property, incorrect numerical insertions and incorrect dataflow connections. Fig. 12 shows the different types of mistake for each category of users.

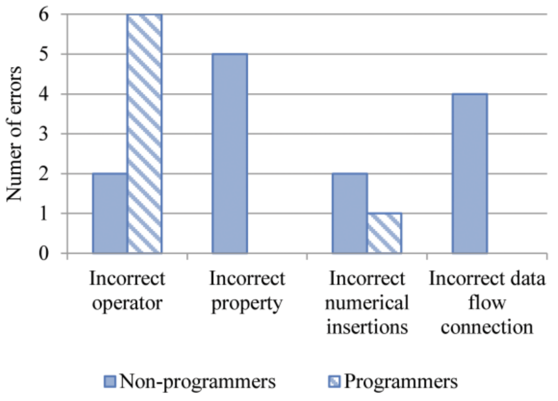

Fig. 12. Number of errors by typology for each category of users.

*Incorrect operator* errors occur when participants use a different operator from the one provided in the rule specification. This was the most common mistake among programmers and was observed to always occur when dealing with the "*greater than*" and "*smaller than*" operators. As can be seen in Fig. 3, the inverse symmetry of these two operators combined with the 360º collection exploration controls leads to confusion, despite the use of graphical aids to help distinguish them, such as a pointer indicating the lower side of the operator. As can be seen in Fig. 12, only 8 operator confusions were made by the totality of participants in the whole experiment. Only 6 programmers confused an operator, each of them only once in the whole session (26 operators in total). Also, there was just one non-programmer who made this mistake twice in the whole experiment.

*Incorrect property* mistakes come up when the user extracts a property other than the one specified in the rule. According to the video recordings of the

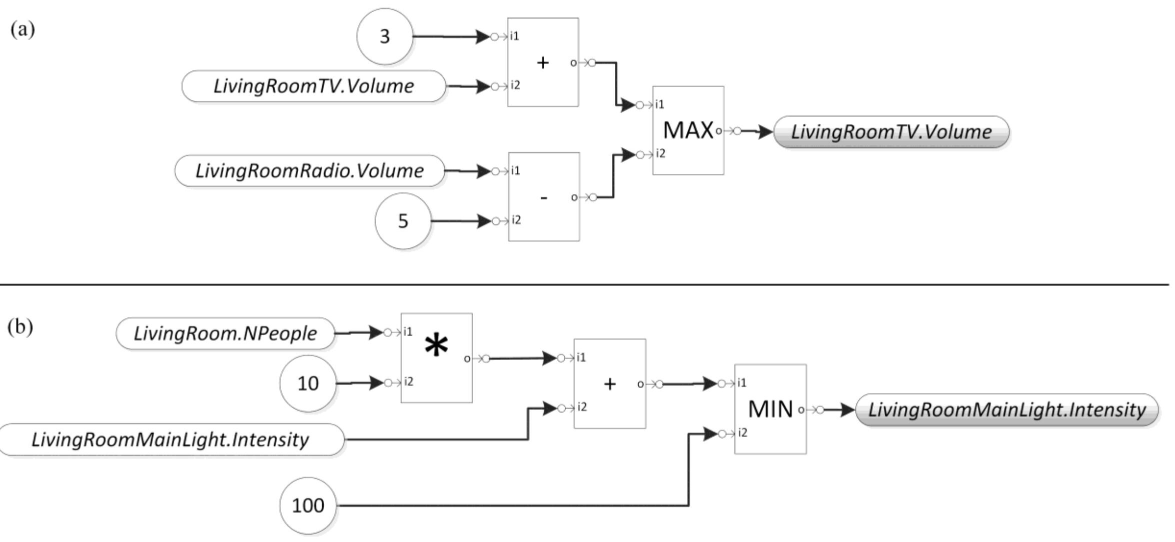

Fig. 13. (a) Tree-like operation. (b) Cascade-like operation.

tests, these mistakes happened under the following circumstances: on two occasions the participant did not look at the correct view of the rule specification on his paper-based solution, and so used the properties shown in the condition instead of those necessary for the operation, as their names were similar. Also, on two occasions the participant chose a property with the same name but belonging to a different population with similar name to the one required (i.e.: *Bedroom1Blinds.Level* and *Bedroom2Blinds.Level*). And on one occasion the error was due to a complete misunderstanding of the operation to be edited. These observations suggest that some kind of visual representation of the elements should be given instead of their names in text. Text-based representation of

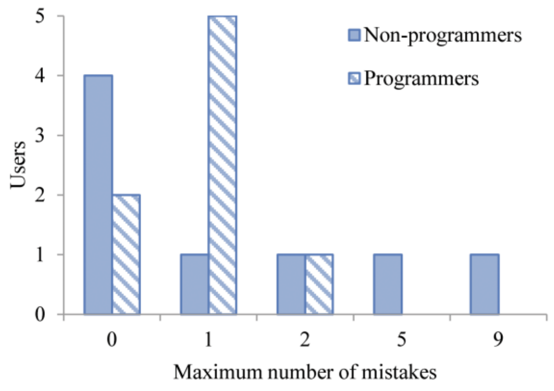

Fig. 14. Maximum number of mistakes made by a single user for each category of participants.

populations seems unsuitable when dealing with 360º controls, as it leads to confusion. Each participant had to extract 16 properties in total, thus, the number of incorrect property mistakes is considered low.

*Incorrect numerical insertions* arise due to the user interface control provided to insert constant numbers. Several participants reported this control as being hard to understand and manipulate (see Section 4.5). When users have to establish several constant numbers in the same data process, they usually join these interactions together. Due to the incorrect use of the tangible widget, they inadvertently modify the values of the previous numerical insertions. A single participant has to perform 16 numerical insertions along the 8 exercises, and considering that programmers only made one mistake, while non-programmers made two, the number of incorrect numerical insertions is far from being critical.

Finally, the *incorrect dataflow connection* error accounts for the properties, constants or operators not connected to the correct input node. This is sometimes due to the similarity of two property names, which causes the participant to swap their connections, or because of an unusual cascade-like series of operators, such as the one in Fig. 13– (b). There are 64 dataflow connections within the 8 exercises each participant edited. Considering that only 2 non-programmers made 2 incorrect dataflow connections each, it has not been considered a problematic issue.

Fig. 14 contains a histogram of the maximum number of mistakes made by a single user in each participant group. In this regard, one programmer made a maximum of two mistakes during the whole session, whereas there were two non-programmers that made 9 and 5 mistakes, respectively. The 9 mistakes can be classified into several categories: mixing up the "*greater than*" and "*smaller than*" operators twice; establishing an incorrect property four times; and selecting an incorrect source or destination population three times. In general, the main source of this participant's errors can be attributed to the editor's legibility issues. The five mistakes made by the other error-prone non-programmer were due to incorrect dataflow connections, incorrect constants and incorrect properties. The unusually high number of mistakes made by this user can be partially explained by the short length of time he devoted to performing the different exercises, as witnessed by the video recordings and log records. In fact, he spent less time on the exercises than several of the programmers, and more than likely did not give enough time to reading and understand the rules to be edited.

### 4.4.3. *Process layout management*

The main aim of this dimension is to evaluate the suitability of the layout of interface elements in the editing area. Considered in this dimension are interactions which do not affect the meaning of the rule but help users feel more at home with the visual interface. Two types of action (collection and node movements) were studied related to the movement and dragging of interface elements on the surface. Analyzing the number of times the participants redistributed both nodes and collections was expected to reveal whether the position of these elements on the surface could give problems to the participants.

Table 5

t-tests comparing the excess node movement ratio for each exercise between programmers and non-programmers.

| Exercise | t | Degrees of freedom (df) | p-value |
|---|---|---|---|
| Exercise 1 | 1.288 | 14 | 0.219 |
| Exercise 2 | 2.196 | 14 | 0.045 |
| Exercise 3 | 1.334 | 14 | 0.203 |
| Exercise 4 | 2.826 | 14 | 0.013 |
| Exercise 5 | -0.516 | 14 | 0.614 |
| Exercise 6 | -0.479 | 14 | 0.639 |
| Exercise 7 | 0.461 | 14 | 0.652 |
| Exercise 8 | 2.016 | 14 | 0.063 |

The number of collection movements during the whole session has been analyzed for both programmers (M = 41.56, SD = 16.459, Mdn = 39.5) and non-programmers (M = 44.42, SD = 20.126, Mdn = 38.50). Mann-Whitney test shows that there are no significant differences between both groups (U = 1945.5, Z = -0.489, p-value = 0.625).

In relation to node movements, a statistical analysis was carried out on the mean *excess node movement ratio*. This ratio was computed from the number of nodes moved by a user divided by the number of nodes that the user extracted into the editing area for each exercise. This measure gives more precise information on how node movements are distributed, since it takes into account the fact that users extracting more nodes than those necessary for the current rule are more likely to perform a higher number of drags. Fig. 15 shows that non-programmers have higher average node dragging ratios than program-

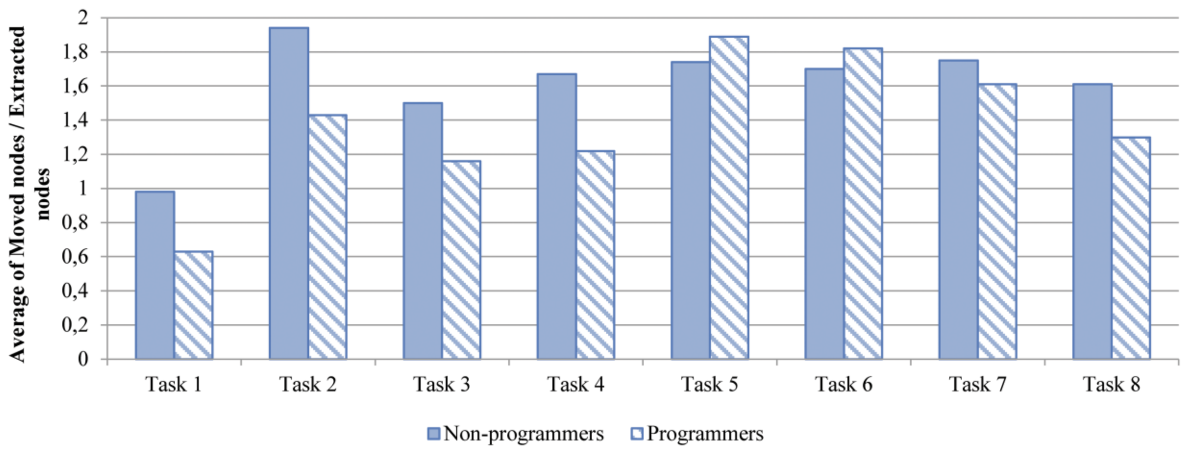

Fig. 15. Average ratio of moved nodes divided by the number of extracted nodes for each exercise.

mers for the majority of the analyzed exercises. However, Table 5 shows no significant differences between non-programmers and programmers, except for Exercises 2 and 4, in which non-programmers moved a significantly higher number of nodes. In both these exercises, two non-programmers performed an extremely high number of node movements. Further analysis of the video recordings showed that this behavior responds to a series of relocations of the elements on the surface in an attempt to correct previous mistakes.

Additionally, the analysis of the session videos also explains the high number of node relocations. Properties, attributes and operators may be added to the editing area by tapping on the corresponding collection element, which are automatically extracted and positioned in the nearest empty area. As more often than not the automatically selected location is at a distance from its final location, the participants are forced to perform a high number of dragging movements to place the nodes at their final destination. To facilitate the editing process, the algorithm for node positioning could be improved in order to find the best place for the current structure of the rule being edited (e.g. empty places near to operators' inputs with compatible types) and not just the empty places around the collection being explored.

A related issue is the spatial distribution of elements inside the canvas. This metric should give visual evidence of the best place in the canvas to place a given type of element, according to the users' preferred positions during the editing process. Redistributing the elements in future versions of the editing tool will make it easier by reducing the number of movements of both nodes and collections.

Fig. 16, Fig. 17, Fig. 18 and Fig. 19 represent the

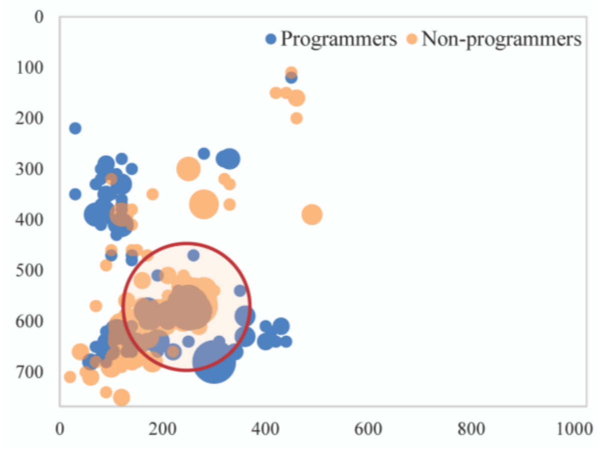

Fig. 16. Final position for events collection.

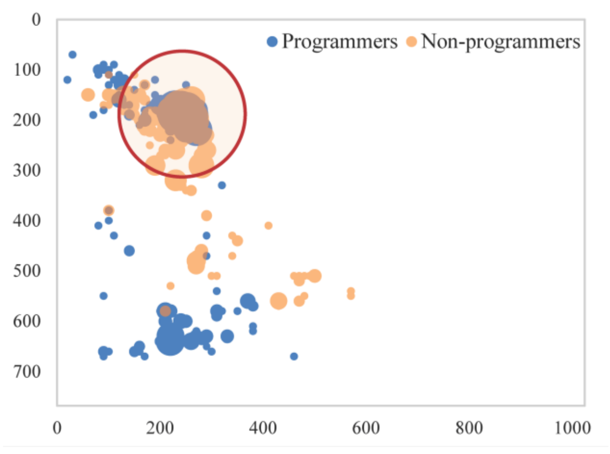

Fig. 17. Final position for source collection.

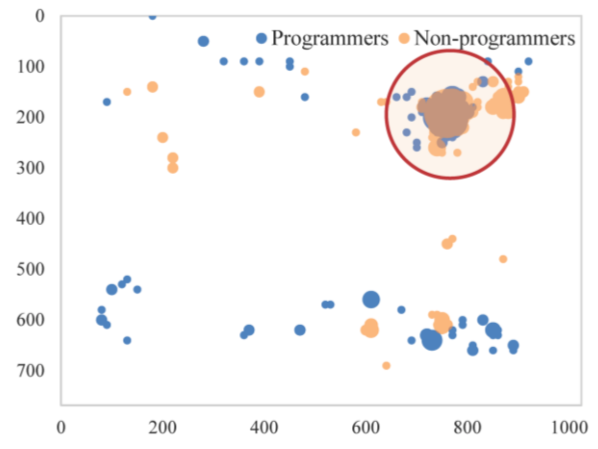

Fig. 18. Final position for target collection.

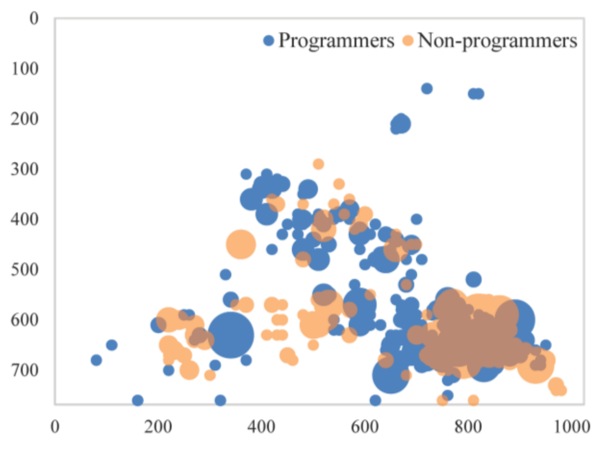

Fig. 19. Final position for operators' collection.

preferred collection positions for both categories of participants for each collection type. The size of the colored bubbles in the mentioned figures indicates the amount of collections positioned in that area: the bigger the bubble is, the more collections have been positioned in that area of the canvas. When defining a rule the initial view forces the user to put the source, target and event collections in a fixed position to start the exploration. These fixed initial positions are shown in Fig. 16, Fig. 17 and Fig. 18 as semitransparent red rounded areas, considering that the user is placed in the bottom side of the picture. It can be seen that both participant groups tend to keep these positions. In addition to the previous collections, there are two exploration collections (operators and foreign elements) which do not have any predefined position, so that the users are free to place them wherever they want. In this case, there is a marked preference by both groups for placing the operators' collection in the lower half of the surface. As can be seen in Fig. 19 and assuming that the user is sitting at the bottom side of the surface, this is near to them so that they can easily manipulate the collection and look for the desired operator with their dominant hand. This is explained by the fact that this is a repetitive interaction during the edition process of dataflows. The position of this control in the workspace agrees with previous studies on the territoriality of controls in tabletop-based user interfaces [35],[37],[44], which indicate that users tend to keep the most frequently used controls close to them, as can be seen in Fig. 19. The figure for the preferred foreign element collection position is not shown as both participant groups did not clearly came up with a specific area in which to display and use this collection. Instead, it was positioned in the empty areas remaining in the canvas, depending on whether the other elements where located.

## 4.5. *Questionnaire results*

The participants were asked to complete questionnaires on ease of use and their personal experience with the editor and its interaction mechanisms. The main goal of the questionnaires was to determine whether the editor was considered an effective tool for behavior specification by each of the user categories. Table 6 shows the eleven questions users had to answer, using a 5-point Likert scale. Fig. 20 shows the median scores for each question by user category.

In general, all the aspects considered were assessed positively, but some guidelines for future im-

Table 6

Questions scored by participants in the questionnaire.

| Id | Question: I consider that… |
|---|---|
| Q1 | Adding properties to the data process being edited is easy. |
| Q2 | Establishing the value for a numeric constant using the numerical setting control is easy. |
| Q3 | Connecting two elements with a dataflow is intuitive and easy. |
| Q4 | Creating dataflows using my fingers is a comfortable task even when having to perform several connections. |
| Q5 | Having tangibles to position and explore the source and target population and the event is useful. |
| Q6 | I would prefer to perform the editing task in collaboration with another user. |
| Q7 | The editor could be useful if I had an interactive tabletop in the context of a smart home. |
| Q8 | This tabletop-based rule editor is a novelty tool. |
| Q9 | The rule editor is easy to use. |
| Q10 | People could easily learn how to use the editor. |
| Q11 | I would not need technical support to manipulate the rule editor. |

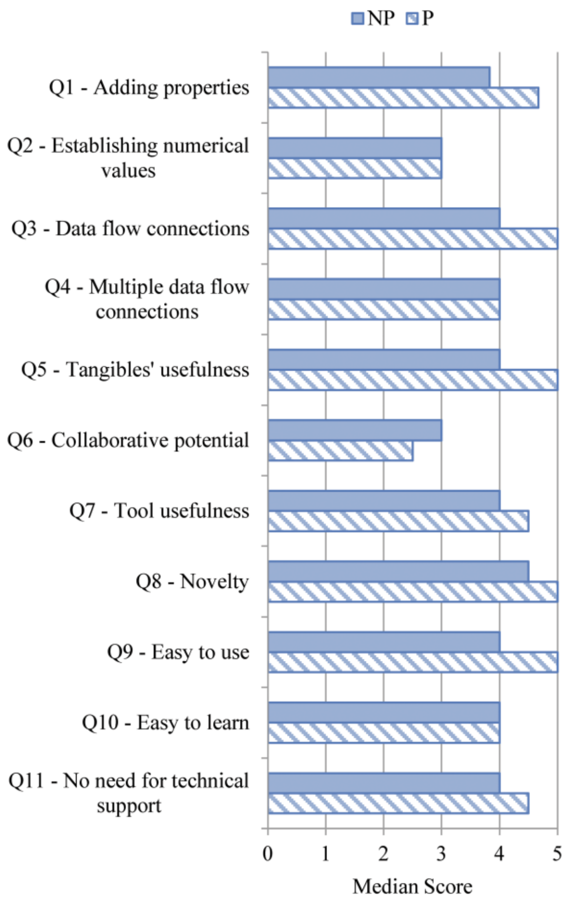

Fig. 20. Median scores by question and participant category.

provements were revealed. For instance, extracting properties into the editing area (Q1) was considered as an easy step by both groups of participants. However, they also reported that establishing the numerical constant using the associated control was not as easy to use as it was designed to be (Q2).

Connecting elements with the fingers in order to create a new dataflow was considered as an intuitive, easy to use interaction mechanism (Q3). If there are a large number of dataflows to connect, this is a useful and easy method of creating expressions (Q4).

Another issue considered was the ability to use tangibles for freely positioning the collections associated with the event and the source or target populations (Q5), which was described as a highly desirable feature. Nevertheless, the editing tool and the controls for exploring collections are not limited to tangible-based manipulation. If future versions of this editor have to be deployed on multitouch platforms with no support for tangible interaction, the collections could be accessed through finger-based interaction and positioning could be done with the fingers.

The participants were also asked whether they preferred editing in collaboration with another user (Q6), as interactive tabletops offer several advantages in this area. This was the only issue in which the programmers showed less interest than the non-programmers, and most preferred editing by themselves. On the other hand, the non-programmers did not show a marked preference for either option.

When asked about the usefulness of the editor within the context of a smart home with an interactive tabletop to specify behavioral rules (Q7), they considered that it would be an effective tool in this regard. The novelty of the editing tool was also highly rated (Q8) by both categories. Both groups also considered that in general the editor was a very easy tool to use (Q9).

The last two questions were designed to evaluate the perceived difficulty of the learning process. They thought people could easily learn how to use the rule editor (Q10) and that little technical support would be needed to be able to specify behavior rules with the proposed editor (Q11).

The participants were also asked to answer several open-ended questions (see Table 7) with the aim of getting their personal opinion and comments. When asked about new connection mechanisms for creating dataflows between elements (OQ1), 8 participants proposed tapping the elements to be connected with the fingers or using tangibles. Another frequent suggestion was to allow bidirectional connection, as in this experiment a dataflow could only be created in the natural data flow direction, from output elements (properties, numerical values) to input elements (operators' inputs, target nodes).

Table 7

Open-ended questions survey.

| Id | Question |
|---|---|
| OQ1 | How would you improve the dataflow connection mechanism? |
| OQ2 | Which parts of the editing process do you think are the clearest/easiest? |
| OQ3 | Which part of the editing process do you think is the least clear/easy? |
| OQ4 | Which part of the editing process do you think is the most tiring? |
| OQ5 | If the source, event and target collections had to be in a fixed position on the surface, which one would you prefer? |
| OQ6 | Comments |

The next three questions dealt with which parts of the editing process the participants perceived as clearest/easiest (OQ2), difficult (OQ3) and tiring (OQ4). The easiest part of the process seems to be selecting the source/target populations and the event (mentioned 8 times), followed by the connection of elements to create dataflows (4 times). The most difficult part was thought to be establishing the numerical values (7 times), followed by exploring the operators' collection to find the desired item (3 times). The answers related to tiring actions did not deal with interaction issues but with the inappropriate height of the interactive tabletop, which forced them to adopt an unnatural posture.

In answer to Q5, the subjects reported a preference for using tangibles to freely position the source, event and target collections. However, when asked to suggest the most appropriate fixed position for these collections (OQ5) many different responses were obtained. All of them would place these collections at the sides of the screen as fixed or deployable collections, but each subject suggested a different configuration: some of them would prefer to have the three collections on the same side and others would keep them on different sides.

Several interesting ideas emerged from the last question, focused on comments and suggestions (OQ6), and thus they will be reported in Section 5.

## 5. Discussion

Several metrics were studied regarding process efficiency, rule correctness and layout management. These studies revealed that both programmers and non-programmers are able to handle the underlying concepts on which the edition tool is based and are able to use the interaction mechanisms provided by the editor to define behavior rules.

Firstly, although the programmers completed the exercises in less time than non-programmers, as could be expected, the average time of both groups only differed when dealing with complex operations. These results agree with the answers obtained from the questionnaires, in which both programmers and non-programmers reported that the editor was easy to use, easy to learn, and they did not believe advanced technical support was required to use it. Consequently, as the non-programmers found it easy to understand the editing task, similar completion times were expected.

Secondly, both groups of participants were observed to behave similarly when managing the elements on the surface. They both tended to move and redistribute nodes several times in a structured way while editing a data process, which suggests that direct touch interactions were used to check whether the expression was complete, instead of simply inspecting what they had done. This seems to indicate that the visual representation of the rule-based behavior is appropriate for helping non-skilled users in such a demanding task and allows them to approach the problem in stages by grouping and reorganizing elements.

The possibility of manipulating collections freely using tangibles is a highly desirable feature in this visual editing process. They have been reported as being a useful mechanism for exploring and positioning collections in order to make full use of the available space. Moreover, positional patterns can be detected from both the video recordings and the final positions of the collections. Most of the participants preferred the same positions for source, event, target and operator collections.

The interaction techniques used to edit dataflows between two elements were considered to be easy to apply, although some usability suggestions were offered in response to the open-ended questions. Some subjects requested the possibility of editing dataflows in both directions, not only from the data producer node to the data consumer node, but also vice versa. Other suggestions included for consideration were using tangibles to connect elements or simply connecting them by tapping.

Some usability problems were also reported and the possible solutions were analyzed. In the prototype, representing elements inside a collection by their text names is inappropriate, as text is hard to read with 360º controls. Several users confused the names when selecting properties or populations or connecting dataflows involving properties with similar names. This problem could be overcome by using visual representations. The types of elements in a smart home are easy to identify by icons, such as an image of a radio or a television. However further solutions will have to be found to differentiate between entities of the same type, e.g. *OfficeRadio* and *BedroomRadio*, perhaps by changing the text orientation or combining visual and textual representations.

A similar problem was found with the *operators* collection when classifying the operators into categories. The textual representation of the operators' category name was not apparent to the participants, and they were forced to explore this collection looking for the required operator inside each category repeatedly. An efficient solution would be to show a miniature of the operators inside a category instead of its name in text. Also, the visual representation of the "*greater than*" and "*smaller than*" operators will have to be changed into a more intuitive one, since it was the source of several mistakes in the editing process; five users reported these operators to be difficult to distinguish. The possibility of collaborative editing has led to the introduction of this type of mirroring mechanisms and 360º controls. However, the problems detected in this regard suggest a reconsideration of the user interface if the tool is going to be used mostly by a single user, thus avoiding 360º visual representations and orienting the elements only towards the user's position.

The control that caused most problems was the numerical value selector, as can be deduced from the questionnaires and due to the high number of deletions in this type of node. This control should be redesigned to improve the insertion of numerical values into the rule. Different solutions were also proposed by the participants, such as simply rotating the tangible to set the number instead of using a slider. Another option would be using "+" and "-" symbols to increase or decrease the numerical value to be established, or possibly the use of a calculator-like control.

## 6. Threats to validity

As in any other empirical study, special attention should be paid to the scope and implications of the results due to limitations and validity threats on the empirical design [48]. The design of the experiment has included some considerations to prevent several validity threats. Regarding the reliability of treatment implementation, all subjects were trained by the same researcher, applying the same treatment and environmental conditions and following a standard guideline to avoid differences during the process. Another consideration was to correctly verify the statistical tests' assumptions and select the best appropriate test for each dimension. In terms of reliability of measures, all the data has been extracted from the log files of the system to assure completeness and correctness of the information. Cronbach's Alpha for the usability questionnaires (except for question Q6, which evaluates the collaborative potential) is 0.742.

The conducted study has been performed on a simulated environment. The obtained results can only be applied to the mentioned experimental conditions. Thus, it has yet to be studied how users would feel from medium to long term using the system in a real environment.

Another issue that has been handled to avoid affecting either the conclusion or the external validity of the experiment is the selection of participants. A very heterogeneous group could lead to a situation in which a small variation of a single participant's performance causes larger differences in the outcomes of the experiment. On the contrary, a very homogeneous group could affect the external validity of the experiment, as it would not allow the generalization from the sample to a broader population. The subjects of the experiment have been selected randomly through mailing lists, but maintaining an adequate equilibrium between homogeneity and heterogeneity of individuals inside each group. On the one hand, programmers have been recruited from different computing backgrounds: software design, multiprocessors networks, videogame development, current bachelor students and PhD students. On the other hand, non-programmers backgrounds also range from diverse areas: laws, business management, physiotherapy, architecture, and current engineering and arts students. For balancing the heterogeneity within groups, the age range of the participants helped to establish a common technological framework: people aged from 17 to 37 are born in the digital era, are familiar to tactile devices and screens and are not afraid of using new technologies. Thus, they are considered the target audience for the developed editing tool, although it would be desirable to study how people from different age ranges make use of the system.

The study lacks of left-handed participants, but the graphical user interface has been designed having in mind any kind of edition: using one or two hands, right or left-handed people, and with the user positioned at any place around the table. Thus, all the controls can be reoriented and repositioned. Therefore, left-handed participants should feel as comfortable as right-handed ones while editing.

Collaboration features have been included in the design of the editing tool but its analysis was not part of the scope of this experiment. It was expected that using a collaborative environment for individual tasks could led to confusions. Nevertheless, the information extracted from the questionnaires and the empirical evaluation of the individual editing exercises has been really helpful. With this information, a refined version of the editing tool could improve the collaboration features of the interface and study the effectiveness of collaborative editing tasks.

A major issue has been the reduced number of participants in the experiment. Even that several replications of the experiments have to be carried out with a higher number of participants, the statistical power of the performed tests according to Cohen's d parameter and the size of the samples is sufficient to extract initial conclusions. Additional experiment replicas would be needed to confirm the results with higher statistical power.

## 7. Conclusions and future work

Customizing future smart environments will require intuitive and highly expressive languages in order to give final users full control of the system. This work presents a rule editing tool for interactive tabletops aimed at specifying behavior in reactive smart environments. The behavior specification in this editor is based on a generic rule model enriched with dataflow expressions, which allows highly expressive rules to be defined in terms of comprehensible representation. An experimental study was conducted to evaluate the suitability of this tool to support behavior definition in intelligent environments, such as a smart home, by users with different programming backgrounds. The results obtained show that, in general, users with no programming experience are able to successfully specify behavior using the proposed visual language and tool, with minimal differences in relation to programmers.

The experiment has served to identify some of the problems that will need to be addressed in the design of future tangible tabletop-based editors for personalizing smart environments. Certain usability problems were detected in the editing tool, such as the inadequate use of text representation, which led to the subjects confusing elements, the ambiguous classification of operators within categories and the defective usability of the numerical setting control. Further work should be done to solve these issues in future versions. New or redesigned interface elements should be provided for the user interface aspects found to cause problems. Other interaction mechanisms, not reported as unsatisfactory, are also being reconsidered. Creating dataflows between nodes with dragging finger movements needs longer manipulation times. In order to reduce this time, a set of improved interaction techniques is being developed and evaluated to be included in future versions of the editor.

The editing tool interface introduces several features which will allow collaborative editing scenarios, e.g., two users in an interactive play room deciding how to configure the color of the light according to the activity being carried out. These features have led to confusion in some cases, and users are not as involved as expected in collaborative edition. Therefore, an additional experiment should be carried out once the interaction problems reported are solved. In this experiment, a collaborative editing rule interface versus an individual one should be presented to users in order to perform the edition individually or in collaboration with other users. This study will reveal whether the neutral feedback obtained in the questionnaires in section 4.5 are just a matter of technology's lack of awareness or not.

Although the editor adequately serves its purpose, non-expert users may have difficulties in their initial attempts to create their own personalized rules. Even experienced users can get into trouble when they attempt to create highly expressive and complex rules from scratch. For this reason users should be provided with assistance during the environmental customization process. An intelligent rule management system is being designed, aimed at allowing non-expert users of smart environments to effectively configure their own behavioral rules, with the help of guidance throughout the process. This assisted edition should provide intelligent mechanisms to anticipate users' interactions, offering suggestions or corrections based on the analysis of previously defined rules inside a global repository by means of heuristics or the detection of rule patterns. This is an ambitious project, as several dimensions have to be explored regarding the appropriate patterns and heuristics to be used, and how suggestions or auto-completions are to be shown on the user interface in order to achieve a non-obtrusive intelligent system.

Other interesting future work related to the evolution of the environment will deal with handling rules in a conflict. In a real smart environment, several users coexist, each one with different preferences. If every user defines his own behavior rules in the same environment, it is likely that conflicting rules will appear. Some work has been done in terms of defining multi-agent systems to manage preferences from several users inhabiting the same environment [15]. Even a single user may inadvertently specify contradictory behavior, so that a mechanism must be provided to automatically detect these conflicts. A study should also be made of designing a comprehensible way of giving feedback to users on any problems detected, to help them to solve the situation. In this regard, interesting studies are emerging focused on *intelligibility*, or the ability to understand how users model the behavior of their environment and how to represent this information to help them to manage the environment when its behavior goes wrong [10].

## Acknowledgements

We would like to thank all the volunteers that participated in the empirical study. Our thanks are also due to the ASIC/Polimedia team for their computer hardware support. This work was partially funded by the Spanish Ministry of Science and Innovation under the National R&D&I Program within the project CreateWorlds (TIN2010-20488). It also received support from a postdoctoral fellowship within the VALi+d Program of the Conselleria d'Educació, Cultura I Esport (Generalitat Valenciana) awarded to Alejandro Catalá (APOSTD/2013/013). The work of Patricia Pons has been supported by the Universitat Politècnica de València under the "Beca de Excelencia" program, and currently by an FPU fellowship from the Spanish Ministry of Education, Culture and Sports (FPU13/03831).

## References

[1] C. Becker, M. Handte, G. Schiele, and K. Rothermel, PCOM - a component system for pervasive computing, in *Proc. of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04),* IEEE Computer Society Washington, DC, USA, 2004, pp. 67–76.

[2] Z. W. Bhatti, N. Z. Naqvi, A. Ramakrishnan, D. Preuveneers and Y. Berbers, Learning distributed deployment and configuration trade-offs for context-aware applications in Intelligent Environments, *Journal of Ambient Intelligence and Smart Environments* **6**(5) (2014), 541–559.

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

[3] D. Bonino and F. Corno, What Would You Ask to Your Home if It Were Intelligent? Exploring User Expectations about Next-Generation Homes, *Journal of Ambient Intelligence and Smart Environments* **3**(2) (2011), 111–126.

[4] D. Bonino, F. Corno, and L. Russis, A User-Friendly Interface for Rules Composition in Intelligent Environments, *Ambient Intelligence - Software and Applications, Advances in Intelligent and Soft Computing,* vol. 92, Springer Berlin Heidelberg, 2011, pp. 213–217.

[5] X. Carandang and J. Campbell, The design of a tangible user interface for a real-time strategy game, in *Proc. of the 34th International Conference on Information Systems (ICIS 2013)*, Association for Information Systems (AIS), 2013, pp. 3781–3790.

[6] A. Catala, F. Garcia, J. Jaen, and J. A. Mocholi, TangiWheel: A Widget for Manipulating Collections on Tabletop Displays Supporting Hybrid Input Modality, *Journal of Computer Science and Technology* **27**(4) (2012), 811–829.

[7] A. Catala, P. Pons, J. Jaen, J. A. Mocholi, and E. Navarro, A meta-model for dataflow-based rules in smart environments: Evaluating user comprehension and performance, *Science of Computer Programming* **78**(10) (2013), 1930–1950.

[8] C. Chen, Y. Xu, K. Li and S. Helal, Reactive Programming Optimizations in Pervasive Computing, in *Proc. of the 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT'10)*, IEEE Computer Society Washington, DC, USA, 2010 , pp. 96–104.

[9] D. J. Cook, J. C. Augusto, and V. R. Jakkula, Ambient Intelligence: Technologies, Applications and Opportunities. *Pervasive and Mobile Computing* **5**(4) (2009), 277–298.

[10] A. K. Dey, Modeling and intelligibility in ambient environments, *Journal of Ambient Intelligence and Smart Environments* **1**(1) (2009), 57–62.

[11] A. K. Dey, T. Sohn, S. Streng and J. Kodama, iCAP: Interactive prototyping of context-aware applications, in *Proc. of Pervasive Computing, Lecture Notes in Computer Science*, vol. 3968, Springer-Verlag Berlin Heidelberg, 2006, pp.254–271.

[12] N. Díaz, J. Lilius, M. Pegalajar and M. Delgado, Rapid prototyping of semantic applications in smart spaces with a visual rule language, in *Proc. of the 2013 ACM conference on Pervasive and ubiquitous Computing adjunct publication*, ACM New York, NY, USA, 2013, pp. 1335–1338.

[13] L. Fuentes, D. Jiménez and M. Pinto, Development of ambient intelligence applications using components and aspects, *Journal of Universal Computer Science* **12**(3) (2006), 236–251.

[14] N. Gámez and L. Fuentes, FamiWare: a family of event-based middleware for ambient intelligence, *Personal Ubiquitous Computing* **15**(4) (2011), 329–339.

[15] M. García-Herranz, X. Alamán, and P. A. Haya, Easing the Smart Home: A rule-based language and multi-agent structure for end user development in Intelligent Environments, *Journal of Ambient Intelligence and Smart Environments* **2**(4) (2010), 437–438.

[16] M. García-Herranz, P. A. Haya, and X. Alamán, Towards a Ubiquitous End-User Programming System for Smart Spaces, *Journal of Universal Computer Science* **16**(12) (2010), 1633–1649.

[17] J. Good, K. Howland, and K. Nicholson, Young People's Descriptions of Computational Rules in Role-Playing Games: An Empirical Study, in *Proc. of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, IEEE, 2010, pp. 67–74.

[18] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux and A. K. Dey, A context-aware service provision system for smart environments based on the user interaction modalities, *Journal of Ambient Intelligence and Smart Environments* **5**(1) (2013), 47–64.

[19] S. Holloway and C. Julien, The case for end-user programming of ubiquitous computing environments, in *Proc. of the FSE/SDP workshop on Future of software engineering research (FoSER'10)*, ACM New York, NY, USA, 2010, pp. 167–172.

[20] M. S. Horn, R. J. Crouser and M. U. Bers, Tangible interaction and learning: the case for a hybrid approach, *Personal and Ubiquitous Computing* **16**(4) (2012), 379–389.

[21] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. K. Jacob, Comparing the use of tangible and graphical programming languages for informal science education, in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*, ACM New York, NY, USA, 2009, pp. 975–984.

[22] C. Kelleher and R. Pausch, Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, *ACM Computing Surveys* **37**(2) (2005), 83–137.

[23] J. Lee, L. Garduño, E. Walker and W. Burleson, A tangible programming tool for creation of context-aware applications, in *Proc. of the 2013 ACM international joint conference on Pervasive and Ubiquitous Computing (UbiComp'13)*, ACM New York, NY, USA, 2013, pp. 391–400.

[24] J. B. Lézoray et al., A design process enabling adaptation in pervasive heterogeneous contexts, *Personal and Ubiquitous Computing* **15**(4) (2011), 353–363.

[25] B. Y. Lim and A. K. Dey, Assessing demand for intelligibility in context-aware applications, in *Proc. of the 11th international conference on Ubiquitous computing (Ubicomp'09)*, ACM New York, NY, USA, 2009, pp. 195–204.

[26] B. Y. Lim and A. K. Dey, Evaluating Intelligibility Usage and Usefulness in a Context-Aware Application, in *Proc. of the 15th International Conference on Human-Computer, Lecture Notes in Computer Science,* vol. 8008, Springer Berlin Heidelberg, 2013, pp. 92–101.

[27] D. López de Ipiña, An ECA Rule-Matching Service for Simpler Development of Reactive Applications, in *Proc. of Middleware 2001,* IEEE Distributed Systems Online **2**(7) (2001).

[28] P. Marshall, Do tangible interfaces enhance learning?, in *Proc. of the 1st international conference on Tangible and embedded interaction (TEI'07)*, ACM New York, NY, USA, 2007, pp. 163–170.

[29] P. Marshall, Y. Rogers and E. Hornecker, Are tangible interfaces really any better than other kinds of interfaces?, in *CHI'07 workshop on Tangible User Interfaces in Context & Theory*, 2007, San Jose, California, USA.

[30] C. Maternaghan and K. J. Turner, A configurable telecare system, in *Proc. of the 4th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'11)*, ACM New York, NY, USA, 2011, pp. 14:1–14:8.

[31] D. A. Norman, *The invisible computer*. MITT Press Cambridge, MA, USA, 1998.

[32] J. F. Pane, B. A. Myers, and C. A. Ratanamahatana, Studying the language and structure in non-programmers' solutions to programming problems, *International Journal of Human-Computer Studies* **54**(2) (2001), 237–264.

[33] P. Pons, A. Catala, J. Jaen, and J. A. Mocholi, DafRule: Un modelo de Reglas Enriquecido mediante Flujos de Datos

*Author version*
Patricia Pons, Alejandro Catalá, Javier Jaén. Customizing Smart Environments: A tabletop approach. Journal of Ambient Intelligence and Smart Environments, vol. 7 (4), pp. 511-533, 2015

para la Definición Visual de Comportamiento Reactivo de Entidades Virtuales, *Actas de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2011)*, 2011, pp. 989–1002.

[34] K. Rasch, An unsupervised recommender system for smart homes, *Journal of Ambient Intelligence and Smart Environments* **6**(1) (2014), 21–37.

[35] K. Ryall, C. Forlines, C. Shen, and M. R. Morris, Exploring the effects of group size and table size on interactions with tabletop shared-display groupware, in *Proc. of the 2004 ACM conference on Computer supported cooperative work*, ACM New York, NY, USA, 2004, pp. 284–293.

[36] A. Schmidt, Implicit human computer interaction through context, *Personal Technologies* **4**(2-3) (2000), 191–199.

[37] S. D. Scott, M. Sheelagh, T. Carpendale, and K. M. Inkpen, Territoriality in collaborative tabletop workspaces, in *Proc. of the 2004 ACM conference on Computer supported cooperative work*, ACM New York, NY, USA, 2004, pp. 294–303.

[38] T. Sapounidis and S. Demetriadis, Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences, *Personal and Ubiquitous Computing* **17**(8) (2013), 1775–1786.

[39] N. Shadbolt, Ambient Intelligence, *IEEE Intelligent Systems* **18**(4) (2003), 2–3.

[40] L. S. Shafti, P. A. Haya, M. García-Herranz and E. Pérez, Inferring ECA-based rules for ambient intelligence using evolutionary feature extraction, *Journal of Ambient Intelligence and Smart Environments* **5**(6) (2013), 563–587.

[41] A. Strawhacker, A. Sullivan and M. U. Bers, TUI, GUI, HUI: is a bimodal interface truly worth the sum of its parts?, in *Proc. of the 12th International Conference on Interaction Design and Children*, ACM New York, NY, USA, 2013, pp. 309–312.

[42] C. Sylla, P. Branco, C. Coutinho and E. Coquet, TUIs vs. GUIs: comparing the learning potential with preschoolers. *Personal and Ubiquitous Computing* **16**(4) (2012), 421–432.

[43] M. A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, J. Jaen and P. González, Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments, *Journal of Information and software Technology* **54**(11) (2012), 1215–1228.

[44] E. Tse, J. Histon, S. D. Scott, and S. Greenberg, Avoiding interference: how people use spatial separation and partitioning in SDG workspaces, in *Proc. of the 2004 ACM conference on Computer supported cooperative work*, ACM New York, NY, USA, 2004, pp. 252–261.

[45] P. Tuddenham, D. Kirk and S. Izadi, Graspables Revisited: Multi-Touch vs. Tangible Input for Tabletop Displays in Acquisition and Manipulation Tasks, in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*, ACM New York, NY, USA, 2010, pp. 2223–2232.

[46] A. Uribarren, J. Parra, R. Iglesias, J. P. Uribe, and D. López de Ipiña, A Middleware Platform for Application Configuration, Adaptation and Interoperability, in *Proc. of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, IEEE Computer Society Washington, DC, USA, 2008, pp. 162–167.

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

[48] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in Software Engineering: an introduction*, first ed. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[49] O. Zuckerman and A. Gal-Oz, To TUI or not to TUI: Evaluating performance and preference in tangible vs. graphical user interfaces, *International Journal of Human-Computer Studies* **71**(7-8) (2013), 803–820.