Visual Languages for Smart Spaces: End-User Programming between Data-Flow and Form-Filling

Michaela R. Reisinger, Johann Schrammel, Peter Fröhlich
Center for Technology Experience
Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at

Abstract— **Visual end-user programming interfaces are becoming increasingly important in the context of smart homes. This paper describes the comparative evaluation of two prototypes following different approaches: form-filling and data-flow programming. We assessed rule completion time, success, user experience and rule recollection as well as suitability to different user-groups. Participants were significantly faster and had a higher overall completion rate using form-filling, but remembered significantly more items when being presented with a data-flow visualization. We therefore recommend approaches that imbibe characteristics of both approaches for end-user programming in a smart home context with untrained users, focusing on a guided process flow, explicit logic and content-categorization.**

I. Introduction and Related Work

A. *The Smart Home Context and Its Programming Tasks*

Homes have predictable logistic needs, but feature high complexity and rapid changes. Truly *smart* homes must enable the control of evolving tasks and routines, and assist in the *control of lives* rather than the *control of devices* [1]. Control can be realized via programming, yet configurations have to be applied promptly, frequently, in a privacy-aware manner and at low-cost [1]. This makes residents most suited to programming their home, albeit often lacking necessary skills: They have a deeper and more detailed knowledge about their activities, routines and surroundings than any third person (should have) [2]. System providers can not anticipate all necessary functionalities [3], nor the changes they undergo over time.

Research on smart homes reveal the prevalence of programming tasks (62.6%) compared to other task types [4]. Users usually express tasks as trigger-action rules, which follow an *if-then* approach. Nearly 25% of such tasks include multiple triggers and/or actions [2], [4].

B. *End-user Programming for Smart Homes*

Programming concepts for smart home residents can and should build on end-user programming research. Both research areas address similar challenges. End-user programmers typically have variable amounts of experience and program in order to reach goals other than programming itself [5]. End-user programming must address syntactic, semantic and pragmatic qualities that constitute barriers to novices. Visual programming languages can bridge this gap and perform a mediation between the conceptual and computational model of a task [6]. This can impact a language's expressiveness (the expressive ability as number of relevant expressible concepts, [7], [8]) and complexity (number of available items [4]). End-user programming languages must maximize expressiveness while reducing syntactic and cognitive gaps to be easy to learn and understand ([6], [9]). Smart home interfaces oftentimes either lack the expressiveness to evolve with household needs or leave too many gaps to be considered intuitive (see C).

Smart homes can employ different *visual representations* to facilitate the specification of rule-based "programs". The most common visual representation is *form-filling*, in which triggers and actions are added by dropdown, drag-and-drop or submenus. While some instances of this form are more visual (e.g. [10]–[13]), others are more textual (e.g. [14]–[16]) or combinations of these approaches (e.g. [17]–[20]).

Data-flow programming is less common, but present in interactive flow editors and wiring diagrams (e.g. [19], [21], [22]). This representation usually provides a canvas onto which items are added from a repository. Items can be arranged freely and connected by attaching a wire or pipe to the items' nodes. This wire represents the logical connection or main data-flow.

Block-programming is a significant visual representation in end-user programming (e.g. [23], [24]). It has been transferred to the smart home only in the form of simple jigsaw metaphors ([19], [25]), lacking the expressiveness of actual block-programming languages. Other visual representations include *card sorting* [26] and the use of *pseudo-natural language* [27].

Visual programming languages can be further characterized by their structure, logic, process and their use of repositories and categorizations: Programming surfaces may be *structurally* pre-defined (e.g. form-filling) or open (e.g. canvas in block programming). *Developmental processes* can use a fixed or open starting point and order [7]. The *developmental logic* can use markers explicitly (e.g. *if*, *then*) or implicitly. *Repositories* can be general (i.e. non-changing throughout the programming) or position-based (i.e. multiple repositories in different places). Repositories can be using *categories* or sorted using user input.

C. *Comparing Programming Visualizations*

Studies show that categorization, complexity and a low cut-off for expressiveness affect the performance and perception of different form-filling representations ([7], [28]). These characteristics are thus important to monitor when comparing different visual programming languages. A qualitative assessment of different visual programming languages of limited expressive power, found form-filling to be more efficient, more perspicuous and well-arranged, data-flow to

This research was performed within the Smart City Demo Aspern (SCDA) project (financed by the Climate and Energy Fund of the Austrian Research Promotion Agency) and the SIM4BLOCKS project (funded from the European Union's Horizon 2020 research innovation program under grant agreement No. 695965).

provide a better view of relationships, and jigsaw programming to be more engaging [19]. Comparing card sorting, textual and visual form-filling showed that card sorting was perceived as faster, more organized and innovative, textual form-filling as more efficient, secure and perspicuous, and visual form-filling as complicated but most optically appealing [26].

Difficulties encountered across different representations include navigating programming interfaces (e.g. [7], [26]), differentiating between states and events, and identifying the category of a desired item [7]. Additionally, users struggle with specific developmental processes (e.g. having to define an action immediately after a trigger [7]) and with the complexity even of basic workflows (e.g. [26]). Users also have difficulties understanding control flows (data-flow programming) and differentiating between triggers and actions (jigsaw puzzle prototype) [19]. The use of specific verbal cues (e.g. if, do), a clear distinction between settings and conditions, and the use of a simple structure [7] was noted as a good logical mechanism.

Depending on cognitive style, e.g. the extent to which people prefer analytical thinking or experiential engagement [33], users might have different preferences for interface styles (e.g. [29]). Additionally, individual differences in visual and verbal processing [34] might be important factors for the preference of and performance with different styles (e.g. [30]).

II. Research Questions

For this study, we focused on the two most frequently used representations among home automation systems: form-filling and data-flow programming. We investigated whether differences noted in I.C can be replicated, or are a product of different complexity and unilateral use of categorization. We furthermore examine the role of personal characteristics, namely need for cognition and verbal-visual dimension.

III. Method

We conducted a laboratory study with 16 participants. In Part I, participants were shown programming sequences of typical smart home rules in form-filling or data-flow representation and were asked to reproduce them as correctly as possible after completing Part II. Understanding how well users comprehend and remember programs written by others is crucial, because smart home programming interfaces should be able to accommodate multiple-user households. In Part II, participants used two prototypes to specify programs of varying complexity.

A. *Participants*

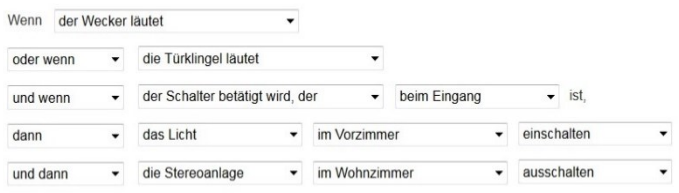
8 male and 8 female individuals (Mean Age = 41.56, SD=15.16) participated in the study. They were regular computer users (two participants using computers multiple times a week, 14 daily) and reported their computer knowledge as average to comprehensive (M=5.2 on a scale from 1 to 7). One person was regularly engaged with computer programming, four persons occasionally did some programming, and 11 persons were never active in this regard. Four participants were using smart home features on a regular basis (heating, air condition, light and electricity). Correspondingly, the reported smart home knowledge ranged from none to comprehensive (M=2.7 on a scale from 1 to 7).

B. *Prototypes*

Both of our prototypes enabled the programming of smart home rules. Each interface held items to define the triggers and actions of a rule. The prototypes were of similar complexity (around 50 items each) and used the same vocabulary for conditions, events, objects and actions. Their repositories were of similar sizes.

Form-filling (Fig. 1) had a pre-defined structure, fixed starting point, open order, explicit logic markers, and position-based repositories. As such, it constrained user input, e.g. that a condition is always and only followed by specifications possible for that condition. Items were stored in drop-down repositories for trigger and action pathways, which included main elements (objects, persons and relations) and specifications (locations, states or actions). While many items were the same for both trigger and action pathways (e.g. *lights*), some were only available in one pathway (e.g. *nobody* only as trigger, *turn off* only as an action). The form-filling prototype had a limit of ten triggers or actions within a program. As this cut-off point was not reached by any participant this difference in expressiveness is not of practical relevance.

Data-flow programming (Fig. 2) consisted of an open canvas and a general, categorized repository. Items were categorized into locations, objects, conditions, actions, individuals, relation items (time, weather, distance) and connecting items. The logic of the experimental prototype was implicit except for Boolean operators. It had an open starting point and open order.



Wenn	der Wecker läutet			
oder wenn	die Türklingel läutet			
und wenn	der Schalter betätigt wird, der	beim Eingang	ist,	
dann	das Licht	im Vorzimmer		einschalten
und dann	die Stereoanlage	im Wohnzimmer		ausschalten

Fig. 1: Example of a program using form-filling. Drop-down menus appear and adjust their content according to the previous selection. Each program starts with if/when (*Wenn*), followed by a main trigger dropdown. Subsequent lines start with a Boolean operator. In this program, the first two trigger conditions have no further specifications, while the third (*der Schalter […]*) specifies a location in an additional drop-down. Both action drop-downs are followed by two option fields on the same line. The full program reads “If | the alarm sounds, | or if | the doorbell rings, | and if | the switch is pressed | at the entrance door | then | [turn] the lights | in the hall | on | and then | [turn] the stereo | in the living room | off ”.

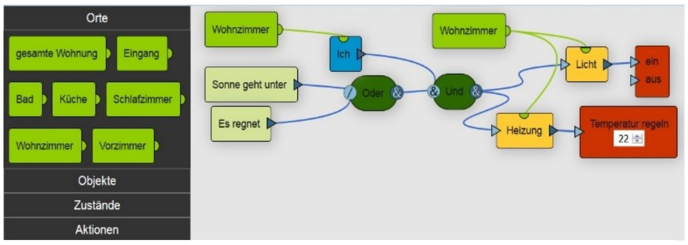


Fig. 2: Example of a program using data-flow programming. The repository (dark grey, left) holds all items categorized into locations (items expanded in picture), objects, conditions, actions, individuals, relations (time, weather, distance) and connecting items. These items can be drag-and-dropped onto the canvas and connected there. The full program includes the following items: The sun is setting (pale green) | or (dark green) | it is raining (pale green) | and (dark green) | I (blue) | living room (bright green, twice) | light (yellow) | heating (yellow) | on/off (red) | set temperature (red).

TABLE I: OVERVIEW OF ALL 12 SITUATIONS (TRANSLATIONS, ORIGINAL IN GERMAN).
REFERENCES DENOTE ESTABLISHED RULES, TRIGGERS OR ACTIONS THAT HAVE BEEN REFRAMED FOR USE IN THIS SET.

No.	Situation Set 1	Situation Set 2
1	The sun is setting and it's getting dark. I need light in the living room. [4]	It is raining. I need light in the kitchen. [4]
2	I want the flat to be nice and warm when I am home. I am comfortable at 22°. [7]	I don't want to heat unnecessarily when I am not home. 18° are enough.
3	It's morning and my alarm clock sounds. I want the bedroom blinds to open. Since I also like to listen to the news, I want the bedroom radio to turn on. [38]	I am pressing the switch in the living room. I want the blinds to be drawn and the TV to be turned on in the living room.
4	If windows or doors are being opened when nobody is home something is wrong. The security system should sound an alarm. [38]	When I am home and it is 8:00 or the alarm is ringing I have to get up. I need light in the bedroom.
5	It is past 22:00 and my bedroom door is closed. All lights are off. I sometimes forget to turn off the TV in the living room. [2], [4]	It is past 21:00 and all windows are closed. The bedroom stereo is on. I sometimes forget to draw the blinds of every room.
6	I am leaving the house, so I press a switch near the entrance. By that, the heating should be lowered to 18° and all lights have to be switched off. I want the security system to be activated after 5 minutes. Since I sometimes forget to press the button, all this should also happen when I am more than 500 m from home. [26]	I am less than 400 meters from home. To prepare for my arrival, the temperature should be increased to 22° and the light in the hall should be turned on. After 5 minutes, WLan should be activated. All this should also happen when I press the switch near the door.

C. *Situations*

In Part II of the study, participants were instructed to solve situations by programming rules (Table I). We used situations instead of explicit tasks (e.g. if A then B) to minimize priming participants towards the more textual prototype. The situations were based on a review of established rules ([2], [4], [7], [19], [26], [31]). There were two sets of situations, which had similar structure and wording, but different items. Situation complexity systematically increased within each taskset (Table I).

D. *Procedure and Measures*

The order of the prototypes was the same for Part I and II but alternated between participants. In Part I, participants started by looking at a picture of a program of each visualization (Fig. 1 and Fig. 2, program alternated between prototypes). Participants were asked to explain what the program did and to inspect the pictures for around a minute. After conducting Part II (see below), they tried to reproduce these programs.

In Part II, participants then were shown how to use the first prototype, and explored it on their own. Once they felt confident, they started with the situations, creating six programs per prototype. After each, participants indicated how confident they were with their solution and how difficult they perceived the task (*Perceived Success and Difficulty*). The prototype recorded the time from the point participants started with a situation until they reported their program as completed (*Task completion time*). After creating the last program per prototype, participants answered the User Experience Questionnaire (UEQ, [32]) and were introduced to the second prototype.

In order to assess the impact of interpersonal differences, we measured cognitive style using the Need for Cognition portion of the German Rational-Experiential Inventory [33]. We additionally used the Visual-Object score of the Object-Spatial Imagery and Verbal Questionnaire [34]. This was succeeded by questions on participant's computer and smart home experience, and demographic data.

To measure task completion, participants' final programs were analyzed according to the use of the proper items (e.g. *lights*, *living room*) and whether individuals connected them in a correct way (e.g. *living room* to *lights*). Their scores were measured against the total number of items and connections necessary for that situation's solution. Participants were not penalized for using a longer or more complicated solution, but their choice of items and connections had to match the higher total of this longer or more elaborate route.

IV. RESULTS

A. *Part 1: Recollection*

Study participants were asked to remember a specific smart home program in each visualization. Comparing the recollection rate (n=15) of the two visual representations, we see that significantly more programming elements were remembered correctly using the data-flow approach than with the form-filling interface (M=1.83, SD =1.91 and M=0.57, SD = 1.08 respectively, t14=-2.92, p=0.038).

B. *Part II: Performative and experiential differences*

The main performance and perception measures were *Task Completion Time*, *Total Task Completion Rate*, *Perceived Difficulty* and *Success*, and *User Experience Questionnaire* scores. MANOVA with these measures as dependent variables and visual representation as independent variable was used for statistical analysis. Table II shows that task completion times were much shorter for form-filling, and that the success rate (both measured and subjectively experienced) was higher for this approach. Qualitative remarks also described form-filling as easier and faster. More than half of the participants felt form-filling to be more suitable for smart homes. Several stated that it had a better process flow, in which they felt more secure due to a pre-defined direction. They also mentioned the restrictive quality and reduced feel of the prototype, greater intelligibility as well as a monotonous, inflexible and old-fashioned feel.

Participants who used data-flow programming in their second turn were considerably faster (M=218.39, SD=140.84) than those who used it in their first (M=400.17, SD=233.25). In comparison, those who used form-filling first were slightly faster (M=100.20, SD=78.38) than those who used it as their second prototype (M=139.27, SD=90.49). There were no meaningful differences in Task Completion Rate, Perceived Success or Perceived Difficulty per turn.

TABLE II: DESCRIPTIVE MEASURES FOR TASK COMPLETION TIME, TASK COMPLETION RATE, PERCEIVED SUCCESS AND PERCEIVED DIFFICULTY.

	Mean (± Standard Deviation)		***MANOVA***
	Form-Filling	***Data-Flow***	
Task Completion Time in seconds	119.74 (±35.69)	306.76 (±144.24)	$F_{1,15} = 34.81$ $p < 0.001$
Task Completion Rate as ratio %	86 (±9)	79 (±12)	$F_{1,15} = 11.99$ $p = 0.003$
Perceived Success 7-Point Likert Scale	6.03 (±1.06)	4.86 (±1.52)	$F_{1,15} = 29.12$ $p < 0.001$
Perceived Difficulty 7-Point Likert S.	2.19 (±1.06)	3.70 (±1.52)	$F_{1,15}=36.18$ $p < 0.001$

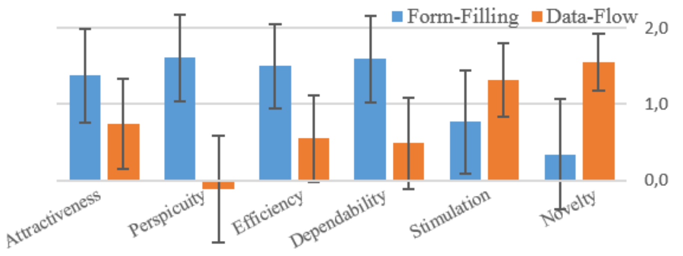


Fig. 3: Mean scores of the User Experience Questionnaire with 95% Confidence Interval. Scores ranged from -3 to 3 on each dimension.

Regarding user experience, perspicuity was by far the most diverging score from the User Experience Questionnaire (Fig. 3). Form-filling scored higher on attractiveness, efficiency and dependability, while data-flow programming scored higher on stimulation and novelty. Prompted which visualization participants would prefer, many observed that hybrid versions would be of more interest to them than either one. They especially mentioned that data-flow programming could be enhanced by some aspects of form-filling (e.g. use fixed spaces for items, pre-defined process to follow). They also remarked that they would like to use form-filling for editing and simple programs but data-flow programming for visually revisiting and revising rules as well as for more complicated programs.

Generally data-flow programming was perceived as more complicated and complex. Some participants observed they would prefer it, if they were more used to it, and that it was more visually appealing. Though it was felt to require more practice and more cognitive effort, it was also noted as more interesting, exciting, creative and playful. Participants noted that "[data-flow programming] is of course superior, even though one is faster using [form-filling]" (P1) and reasoned that their preference or better understanding stemmed from the "logical" use of colors and categories. Several participants remarked that because data-flow programming allowed them to see multiple items at once, it was better arranged and more manageable.

C. *Part II: Influence of personal characteristics*

We used R [35] and lme4 [36] to perform a linear mixed effects analysis of the relationship between task completion time, prototype and personal characteristics. As fixed effects, we entered prototype, Need-for-Cognition and Visual-Object score (without interaction terms) into the model. Intercepts for subjects were included as random effects. Visual inspection of residual plots did not reveal any obvious deviations from homoscedasticity or normality. P-values were obtained by likelihood ratio tests of the full model with the effect in question (i.e. Need-for-Cognition or Visual-Object score) against the model without the regarding effect in question.

The results of our analysis shows that contrary to our expectations Need-for-Cognition does not seem to have any relevant influence on the performance with the two programming approaches ($\chi_2(1)=1.13$, p=0.288). Visual-Object score shows an influence on task performance time ($\chi_2(1)=4.07$, p=0.043), however the direction of this influence is opposite to our anticipation. We expected users with a high Visual-Object score to complete the tasks using the visual data-flow approach faster than people with a lower score.

V. Discussion

Comparing the performance and perception of two end-user programming visualizations in a smart home context, we find that form-filling is seen as more perspicuous and performed superior in our study. However, data-flow approaches did also show important qualities of greater engagement and better recollection. Our findings are largely in line with previous work (e.g. [26]), though recollection has not been focused on so far. As the used prototypes were of similar complexity and used categorizations, our results indicate that expressiveness differentiates the performance of approaches.

Participants noted to feel more secure in their solutions in form-filling, yet this feeling of security might have contributed to quicker decisions on whether a program was complete, and to miss elements their program still lacked. This underlines the necessity of reviewing and debugging facilities ([12], [37]). Regarding complementary strengths and weaknesses of both approaches, form-filling is less time consuming to use but also felt restrictive to some users. Its main strength lies in its guided process flow and explicit logic.

In contrast, our data-flow prototype necessitated too many definitory decisions, not all of which were relevant to users: To increase its performance, we think it would be needed to substantially restrict item and connection choices. The significant choices would need to be visible at a glance, for example by drawing on aspects of other end-user programming languages such as block-programming. Our results suggest that data-flow may be superior in the manner of categorization, namely that its use of color-coding and its repository categories felt more logical to users. This, combined with providing a better overview, recollection and connection visualization could make data-flow programming a promising candidate for debugging and testing facilities.

A more high-end implementation of a data-flow approach might be able to achieve better results than the rather basic implementation of our prototype. Possibilities for improvement here lie in improved methods for interactively 'snapping' connections together, for automatically aligning items to achieve appealing displays, better indication of possible fit between elements, etc. Using such approaches it should be possible to reduce the gap in performance to form-filling while maintaining the advantages of the data-flow approach.

Regarding the influence of personal characteristics, more strongly visually oriented individuals took longer completing tasks using the data-flow prototype - contrary to our expectations. This might be due to more visually oriented individuals taking more time to improve the visual appearance of their programs (e.g. by re-arranging) or being more diligent about item placement or connection aesthetics.

In the smart home context, control can be expressed as the establishment of boundaries [1], which has to benefit users time-wise [2]. Task complexity will increase with the number of controllable elements. Regarding context and complexity, we therefore recommend a hybrid approach that incorporates a guided process flow, explicit logic both in verbal cues (e.g. if, then) and color-coding and content-categorized repository for use in more expressive or complex smart-home scenarios.

REFERENCES

[1] S. Davidoff, M. K. Lee, C. Yiu, J. Zimmerman, and A. K. Dey, "Principles of smart home control," in *UbiComp 2006*, 2006, vol. 4206, pp. 19–34.

[2] A. K. Dey, T. Sohn, S. Streng, and J. Kodama, "iCAP: Interactive Prototyping of Context-Aware Applications," in *4th International Conference, PERVASIVE 2006*, 2006, vol. 3968 LNCS, pp. 254–271.

[3] J. Coutaz, E. Fontaine, N. Mandran, and A. Demeure, "DisQo: A user needs analysis method for smart home," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*, 2010, pp. 615–618.

[4] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," *Proc. 32nd Annu. ACM Conf. Hum. factors Comput. Syst. - CHI '14*, pp. 803–812, 2014.

[5] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–44, 2011.

[6] A. Repenning and T. Sumner, "Agentsheets : A Medium for Creating Visual Languages," *Computer (Long. Beach. Calif).*, vol. 28, no. March, pp. 17–25, 1995.

[7] G. Lucci and F. Paternò, "Understanding End-User Development of Context-Dependent Applications in Smartphones," in *5th IFIP WG 13.2 International Conference of Human-Centered Software Engineering - HCSE 2014*, 2014, vol. 8742, pp. 182–198.

[8] G. Lucci and F. Paternò, "Analysing How Users Prefer to Model Contextual Event-Action Behaviours in Their Smartphones," in *5th International Symposium on End-User Development, IS-EUD 2015*, 2015, vol. 9083, pp. 186–191.

[9] I. Perez and A. Sturm, "DEV4ME: Can End-Users Program?," in *European Conference on Information Systems (ECIS)*, 2014, pp. 1–9.

[10] IFTTT, "If This Then That," 2010. [Online]. Available: www.ifttt.com. [Accessed: 18-Aug-2015].

[11] Kärntner Elektrizitäts-AG, "Smarthome Austria," 2015. [Online]. Available: https://www.smarthome-austria.at/. [Accessed: 18-Sep-2015].

[12] J. Woo and Y. Lim, "User Experience in Do-It-Yourself-Style Smart Homes," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*, 2015, pp. 779–790.

[13] RheinEnergie AG, "RheinEnergie - SmartHome," 2015. [Online]. Available: http://www.rheinenergie.com/de/privatkundenportal/smarthome_1/smarthome_shop/index.php. [Accessed: 18-Sep-2015].

[14] L. Barkhuus and A. Vallgårda, "Smart Home in Your Pocket," in *Adjunct Proceedings of UbiComp 2003*, 2003, pp. 165–166.

[15] EQ-3 AG, "HomeMatic, Max!," 2015. [Online]. Available: http://www.eq-3.de/. [Accessed: 18-Sep-2015].

[16] Telekom Deutschland GmbH, "Telekom Smart Home," 2015. [Online]. Available: https://www.smarthome.de/. [Accessed: 18-Sep-2015].

[17] Belkin, "WEMO." [Online]. Available: http://www.belkin.com/de/PRODUKTE/home-automation/c/wemo-home-automation/. [Accessed: 11-Apr-2017].

[18] Archos, "ARCHOS Smart Home," 2015. [Online]. Available: http://www.archos.com/at/products/objects/chome/ash/index.html. [Accessed: 18-Sep-2015].

[19] Y. Dahl and R.-M. Svendsen, "End-User Composition Interfaces for Smart Environments: A Preliminary Study of Usability Factors," in *1st International Conference on Design, User Experience and Usability, DUXU 2011, Held as Part of HCI International 2011*, 2011, vol. 6770, pp. 118–127.

[20] Energie Baden-Württemberg AG, "EnBW Smart Home," 2015. [Online]. Available: https://www.enbw.com/privatkunden/tarife-und-produkte/smart-home/index.html. [Accessed: 18-Sep-2015].

[21] Athom, "Homey. Everything at home connected." [Online]. Available: https://www.athom.com/en/. [Accessed: 11-Apr-2017].

[22] Loxone, "Loxone Home Automation." [Online]. Available: https://www.loxone.com/dede/. [Accessed: 11-Apr-2017].

[23] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. a Y. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for All.," *Commun. ACM*, vol. 52, pp. 60–67, 2009.

[24] R. V Roque, "OpenBlocks: an extendable framework for graphical block programming systems," Massachusetts Institute of Technology, 2007.

[25] J. Humble, A. Crabtree, T. Hemmings, K. P. Akesson, B. Koleva, T. Rodden, and P. Hansson, "' Playing with the Bits' User-Configuration of Ubiquitous Domestic Environments," in *5th International Conference of Ubiquitous Computing: UbiComp 2003*, 2003, vol. 2864, pp. 256–263.

[26] G. Leitner, A. J. Fercher, and C. Lassen, "End Users Programming Smart Homes – A Case Study on Scenario Programming," in *3rdInternational Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data, HCI-KDD 2013, Held at SouthCHI 2013*, 2013, vol. 7947, pp. 217–236.

[27] J. Coutaz, A. Demeure, S. Caffiau, and J. L. Crowley, "Early Lessons from the Development of SPOK, an End-user Development Environment for Smart Homes," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 2014, pp. 895–902.

[28] F. Cabitza, D. Fogli, R. Lanzilotti, and A. Piccinno, "End-User Development in Ambient Intelligence : a User Study," in *CHItaly 2015 Proceedings of the 11th Biannual Conference on Italian SIGCHI*, 2015, pp. 146–153.

[29] R. Mancy and N. Reid, "Aspects of cognitive style and programming," *16th Work. Psychol. Program. Interes. Gr.*, no. April, pp. 1–9, 2004.

[30] J. Lee, "The effects of visual metaphor and cognitive style for mental modeling in a hypermedia-based environment," *Interact. Comput.*, vol. 19, no. 5–6, pp. 614–629, Dec. 2007.

[31] C. Perera, S. Aghaee, and A. Blackwell, "Natural Notation for the Domestic Internet of Things," in *5th International Symposium on End-User Development, IS-EUD 2015*, 2015, vol. 9083, pp. 25–41.

[32] B. Laugwitz, T. Held, and M. Schrepp, "Construction and Evaluation of a User Experience Questionnaire," *Usability Prof. Assoc.*, pp. 63–76, 2008.

[33] J. Keller, G. Bohner, and H.-P. Erb, "Intuitive und heuristische Urteilsbildung – verschiedene Prozesse ? Präsentation einer deutschen Fassung des " Rational-Experiential Inventory " sowie neuer Selbstberichtskalen zur Heuristiknutzung," *Zeitschrift für Sozialpsychologie*, vol. 31, no. 2, pp. 87–101, 2000.

[34] O. Blazhenkova and M. Kozhevnikov, "The new object-spatial-verbal cognitive style model: Theory and measurement," *Appl. Cogn. Psychol.*, vol. 23, no. 5, pp. 638–663, Jul. 2009.

[35] R Core Team, "R: A language and environment for statistical computing." R Foundation for Statistical Computing, Vienna, Austria, 2012.

[36] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting Linear Mixed-Effects Models Using lme4," *J. Stat. Softw.*, vol. 67, no. 1, 2015.

[37] A. Demeure, S. Caffiau, E. Elias, and C. Roux, "Building and Using Home Automation Systems: A Field Study," in *End-User Development*, vol. 9083, P. Díaz, V. Pipek, C. Ardito, C. Jensen, I. Aedo, and A. Boden, Eds. Cham: Springer International Publishing, 2015, pp. 125–140.

[38] M. García-herranz, P. Haya, and X. Alamán, "Towards a Ubiquitous End – User Programming System for Smart Spaces," *J. Comput. Sci.*, vol. 16, no. 12, pp. 1633–1649, 2010.