Visual End-User Programming in Smart Homes: Complexity and Performance

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

***Abstract*— End-user programming in smart homes addresses tasks that range from very simple to very complex. In this study we investigate how task complexity impacts performance when using two different visual programing representations: form-filling and data-flow programming. We invited 16 participants to create rules to solve smart-home situations of varying complexity and analyzed their completion rates for the two visual programming representations. We identify the following areas of difficulty for programming novices in our smart home scenario: choosing and connecting triggers and their specifications, and using Boolean operators. Both visual representations enabled users to specify complex rules, with advantages in different areas. They indicate that overall task complexity might be less decisive for performance than complexity of triggers and Boolean operators.**

I. Content and Claims

For this poster, we investigate how the complexity of an end-user programming task impacts users' performance with two different visual programming representations in a smart home context. In this context, programs span a wide spectrum from simple to highly complex. To understand the applicability of a visual programming language in a realistic smart-home scenario, it is necessary to address the whole complexity range.

Previous research did not find significant differences for performing simple or complex conjunctive trigger-action programming in different form-filling approaches [1]. Yet, it is unclear how well individuals differentiate between conjunctive and disjunctive items. It remains similarly unresolved whether users are assisted differentially by different visual representations of end-user programming. In order to answer these questions we conducted a laboratory study with 16 participants (8 male, 8 female). Users were tasked with programming smart home rules of varying complexity (defined as number of items, [1]) using two prototypes realizing different programming approaches.

A. Tasks and Prototypes

We used two sets of situations participants were instructed to solve by creating a smart home rule. Situations in either set had the same structure but different items (e.g. *The sun is setting. I need light in the living room.* vs. *It is raining. I need light in the kitchen.*). The situations systematically increased in complexity (see *rule structure* in Table I). Two situations (4 and 6) used disjunctions as well as conjunctions. Each situation was equally often used with each prototype. Prototypes had a form-filling or data-flow visual representation. In the *form-filling approach*, items were chosen via drop-down menus that formed a sentence. In the *data-flow approach* items were drawn from a repository, positioned on a canvas and connected via lines. Details on situation generation and the prototypes can be found in [2].

B. Procedure and Measures

After familiarizing themselves with the prototype, participants created rules for one set of situations with each prototype. Their use of correct items (e.g. *bedroom*, *radio*) and correct connections (e.g. *bedroom* to *radio*) compared to the total number of items and connections necessary for the program yielded their *Item Completion* and *Connection Completion Rates*. Some situations could be solved in one way only, while others had multiple solutions. If participants created a more complex rule, they were not penalized, but their rule had to measure against the higher item and connection count of the chosen solution. If participants used an item in place of one of the same category (e.g. *bedroom* instead of *kitchen*) they were penalized within their item completion rate, but not within their connection completion rate if the connections were otherwise correct. *Item Completion Rate* and *Connection Completion Rate* were used to calculate their *Total Completion Rate*.

C. Results

In order to investigate complementary strengths and weaknesses of the prototypes and the impact of rule complexity on performance of both prototypes, we analyzed *Total Completion Rate*, *Item Completion Rate* and *Connection Completion Rate* per rule type (Table I):

Both representations had similar total completion rate ranges (70-97% and 64-92% in form-filling and data-flow respectively) and a similar range of differences between item and connection completion rates (1-13% and 5-18% respectively). Form-filling had higher total completion rates in all rules except triple-trigger-single-action rules. Mixing conjunctive and disjunctive trigger types (task 4) had a lower completion rate than the longer task structure of task 6 in both visualizations. Connection completion rates were higher than item completion rates in the first three but not in subsequent tasks for form-filling. Using data-flow

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).

programming, item completion rates were consistently higher than connection completion rates.

Item completion ranged from 68% to 93% in form-filling and from 44% to 95% in data-flow respectively (Table II). Using form-filling, participants were less successful choosing proper triggers and trigger specifications than choosing actions and action specifications. Their connection completion rate was also lower for trigger-trigger connections than for other connections using form-filling (Table III). This greater difficulty of providing and connecting multiple triggers in form-filling can also be seen in lower total completion rates for tasks 4 and 5 (as opposed to multiple actions in task 2 and 6).

Triggers were chosen and connected more successfully in data-flow programming, but participants had difficulties implementing disjunctive trigger conditions and connecting multiple elements in that prototype (lower total completion rates for task 4 and 6 than for task 5,

Table). This is mirrored by the especially low item and connection completion rates for operators (Table II, Table III): Operator-operator connections were least often completed, followed by trigger-operator connections (26.3, 43.8%, and 52.8%, per trigger type - person, state or relation respectively) and operator-action connections (55.3%). Participants were generally quite able to specify triggers and actions using locations, states or actions (86% and 71.1% respectively) when using the data-flow prototype.

II. Relevance

Our results indicate that complexity does not impact visual representations in the same way, and that it is not the total number of items that creates this difference: Task 6 was not the most difficult task in either visual representation. Instead, it is tasks with a greater number of triggers (5 and 4 respectively) that proved more difficult to participants. Data-flow item completion rates indicate that using states and events and differentiating between them (e.g. *a window is opened* vs. *a window is open*) may be more difficult than other trigger specifications. The form-filling prototype made this distinction more intuitive due to its verbal nature. In that prototype, identifying necessary conditions and events posed greater difficulty than identifying necessary actions, (indicated by the form-filling's low completion rates for trigger-specification items and trigger-trigger connections).

Boolean operators had the lowest item completion rates in the data-flow approach. This indicates that individuals are least habitual using such elements in an exact logical sense. It is thus an area where they need support most. Low trigger-operator, operator-operator and operator-action connection rates additionally indicate that using them without a supportive structure is not intuitively understandable to programming novices. Using a guided specification scenario or designated *if/then* layers - elements that representations with an open canvas oftentimes lack - would be able to assist individuals in such a visual representation. Likewise, Boolean operators would be more easily useable if embedded in a predefined grid, which could also increase this representation's lower connection completion rates.

TABLE I. RULE STRUCTURES AND TOTAL COMPLETION RATES.

Task	Rule structure [a]	Total completion rate (Item completion, connection completion)	
		Form-Filling	***Data-Flow***
1, 2[b]	T → A	96.45% (94.1%, 98.8%)	91.94% (97.3%, 86.6%)
3	T → A & A	96.90% (96.3%, 97.8%)	80.59% (85.1%, 76.1%)
4	T / T & T → A	76.61% (79.6%, 73.6%)	63.82% (71.6%, 56.0%)
5	T & T & T → A	70.33% (71.8%, 68.9%)	79.72% (88.0%, 71.2%)
6	T / T → A & A & A & A	82.37% (88.7%, 76.0%)	68.04% (77.0%, 59.1%)

a. *T* indicates a trigger, *A* an action, *&* a conjunction, */* a disjunction.

b. The different trigger types in rule 1 and 2 had no meaningful effect (0.2% difference), hence they are reported conjointly.

TABLE II. ITEM COMPLETION RATES PER ITEM TYPE. ITEM TYPES OF DIFFERENT VISUALIZATIONS CONCEPTUALLY OVERLAP AS SHOWN. [a, b]

Form-Filling *Item type*	**Item completion rates**		**Data-Flow** *Item type*
Main Trigger	73.9%	94.7%	Relations
		80.8%	Individuals
Main Action	90.6%	91.2%	Objects [a]
Trigger Specification	67.5%	76.6%	States & Events
Action Specification	92.9%	87.4%	Location [b]
		90%	Actions
Operators	85.7%	44.4%	Operators

a. Overlap with Main Trigger and Main Action: Objects could be used both within a trigger and within an action, while Relations and Individuals could only be used as triggers.

b. Overlap with Trigger and Action Specification: Locations could be used to specify trigger and/or actions, while States and Actions were only applicable as specification of either.

TABLE III. CONNECTION COMPLETION RATES PER CONNECTION TYPE.

Form-Filling		**Data-Flow** [e.]	
Connection type	***Connection completion rate***		***Connection type***
Trigger - Action [a.]	92.7%	81.9%	Trigger – Action [a.]
Trigger - Trigger	60.4%	86.0%	Trigger – Specification [b.]
		41.0%	Trigger – Operator [c.]
Action – Trigger [d.]	81.7%	12.5%	Operator – Operator
		55.3%	Operator – Action [c.]
Action - Action	85.7%	71.1%	Action – Specification [b.]

[a] This type is the only directly comparable type since direct triggers-action connections were legit in both prototypes. [b.] Specifications had to be connected to their main element in data-flow, but were automatically connected in form-filling; they do not have an equivalent there. [c.] Multiple triggers had to be connected to an operator in order to link to a subsequent action element in data-flow, which would be equivalent to trigger-trigger connections in form filling. Yet, trigger-operator connections additionally include other cases (e.g. the operator functioning as attachment for multiple actions). The same holds true for action-action vs. operator-action connections. [d.] Building a program serially in form-filling, an action might be followed by a condition (again followed by another instance of the same action), creating an action-trigger-connection. [e.] Participants would sometimes rearrange items into a correct order in the data-flow prototype, but not reform connections between them. This could have contributed to its low connection completion rates.

References

[1] 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.

[2] M. R. Reisinger, J. Schrammel, and P. Fröhlich, "Visual Languages for Smart Spaces: End-User Programming between Data-Flow and Form-Filling," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'17)*, 2017.