



Department of Psychology and Cognitive Science
Doctoral School in Cognitive Science
XXXVII cycle

LANGUAGE AND MENTAL MODELS IN TRIGGER-ACTION PROGRAMMING

PhD Candidate: **Margherita Andrao**

Advisor:
Massimo Zancanaro

June 2025

Abstract

Allowing individuals to customize and manage the functions of their smart devices is essential in various contexts, ranging from automating daily routines in smart home environments to customizing educational paths in school classes. End-User Development (EUD) through Trigger-Action Programming represents a promising approach to enabling users – regardless of prior programming experience – to achieve their goals defining automation. However, for non-programmers to effectively manage complex scenarios involving multiple variables and events, EUD solutions must support the assumption of appropriate reasoning strategies and mental models. Prior research suggests that specific linguistic cues can facilitate the representation of trigger-action dynamics, thereby enhancing users' ability to define the behavior of their devices. This thesis investigates the role of linguistic elements in guiding non-programmers in the definition of trigger-action rules for programming their smart devices, supporting users in assuming effective mental models and reasoning strategies. Through three studies on language, this work highlights design guidelines for developing systems that better support non-programmers in defining trigger-action rules. Moreover, two systems based on Trigger-Action Programming leveraging two different compositional approaches – form-based and LLM-based – are explored to examine users' interaction with the system and the impact of language and mental models in real-world applications.

Keywords

End-User Development, Trigger-Action Programming, Mental Models, Language

Contents

1	Introduction	1
1.1	Context	2
1.2	Research Objectives and Contributions	4
1.2.1	Objective 1: Language and Mental Models in Trigger-Action Programming	4
1.2.2	Objective 2: Studying Interaction with Trigger-Action Based Systems	7
1.3	Outline of the thesis	11
2	Multiple Perspectives on Mental Models	13
2.1	History and Theories of Mental Models	14
2.1.1	The Theory of Mental Models in Cognitive Psychology	15
2.1.2	Mental Models in Human-Computer Interaction	18
2.2	Different Mental Models: Terminological Disambiguation	20
2.2.1	Terminological Ambiguity	20
2.2.2	Unifying and Differentiating	20
3	End-User Development through Trigger-Action Programming	23
3.1	End-User Development: Definition and Field Overview	23
3.1.1	Related Concepts Definition	24
3.1.2	Paradigms and Metaphors Proposed	25
3.2	Trigger-Action Programming	27
3.2.1	Simple Trigger-Action Rules	28
3.2.2	Complex Trigger-Action Rules	29
4	Factors Influencing Mental Models Assumption in End-User Development Tasks	33

4.1	Temporal Aspects: Distinguishing Events and States	34
4.1.1	Internally Mapping Event-State Distinction in the System	35
4.1.2	Exploiting Linguistic Cues to Inform Event-State Distinction	36
4.2	The Order of Rule's Elements	40
4.2.1	Psycholinguistic Theories: Iconicity and Figure-Ground	40
4.2.2	Trigger-Action or Action-Trigger?	42
4.3	Abstract or Concrete Content	45
5	Studying Language and Mental Models in Trigger-Action Programming	47
5.1	Study 1: A Qualitative Study on Trigger Interpretation in Trigger-Action Rules	48
5.1.1	Participants	49
5.1.2	Materials	49
5.1.3	Task and Procedure	51
5.1.4	Data analysis	53
5.1.5	Results	53
5.1.6	Discussion	64
5.2	Study 2: Investigating Which Linguistic Clues Effectively Guide Non-Programmers	65
5.2.1	Participants	66
5.2.2	Materials	66
5.2.3	Task and Online Procedure	67
5.2.4	Results	70
5.2.5	Discussion	76
5.3	Study 3: Exploring Natural Linguistic Patterns in Trigger-Action Rules	78
5.3.1	Participants	79
5.3.2	Materials	80
5.3.3	Procedure and Task	83
5.3.4	Pilot Study A: Appropriateness of Instructions	83
5.3.5	Pilot Study B: Refinement of Materials and Procedure	85
5.3.6	Data analysis	88
5.3.7	Results	90
5.3.8	Discussion	93

6	Studying Interaction with End-User Development Systems	97
6.1	Investigating Linguistic Clues in Form-Based Compositional Visual Interfaces	98
6.1.1	Context: End-User Development for Educational Technology . .	99
6.1.2	Research Questions	99
6.1.3	The System	100
6.1.4	Study 4: Exploring Teachers Mental Models in Composing and Debugging Trigger-Action Rules	107
6.1.5	Study 5: Studying Teachers Mental Models on Trigger-Action Programming	115
6.1.6	Results	121
6.1.7	Discussion	129
6.2	Exploring How Large Language Model-Based Systems Addresses Ambiguities in Language	131
6.2.1	Context: Home Automation Systems and Accessibility	132
6.2.2	Research Questions	135
6.2.3	Study 6: Exploring How LLMs Handle Linguistic Ambiguities .	135
6.2.4	Study 7: Evaluating LLMs' Ability to Interpret Ambiguous Instructions from Individuals with Intellectual Disabilities	143
7	Toward New Interaction Approaches	173
7.1	RuleCraft: an End-User Development Hub for Education	173
7.1.1	The System Architecture	175
7.2	LLM-Based Iterative Disambiguation Process: A Pilot Study	177
7.2.1	Participants	178
7.2.2	Materials	179
7.2.3	Task and Procedure	180
7.2.4	Data Analysis	182
7.2.5	Results	182
7.2.6	Discussion	185
8	General Discussions	187
8.1	Retrospective on Objectives	187
8.1.1	Addressing RQ1: Language for Distinguishing States and Events	187
8.1.2	Addressing RQ2: Conjunctions and Verbal Structures	188
8.1.3	Addressing RQ3: Order of Elements	188

8.1.4	Addressing RQ4: Mental Models during Interaction	189
8.1.5	Addressing RQ5: Abstract and Concrete Descriptions	190
8.1.6	Addressing RQ6: Natural Language Ambiguities	190
8.1.7	Addressing RQ7: Intent Recognition and Disambiguating Questions	191
8.2	Implications for the Design of End-User Development Tools	191
8.2.1	Mapping Linguistic Cues in Systems' Ontology	191
8.2.2	Form-Based Compositional Visual Interface to Guide Users in Composing Trigger-Action Rules	194
8.2.3	LLM-based System for Handling Ambiguities in Trigger-Action Rules	197
9	Conclusions	201
9.1	Limitations	203
9.2	Future Directions	203
9.2.1	Linguistic Cues and Mental Models	203
9.2.2	User Interaction with TAP-Based Systems	204
	Bibliography	205
	Appendices	231

List of Tables

4.1	This table summarizes the conjunctions used in research prototypes ("research") and commercial tools ("comm.") to introduce states and events in the trigger (in bold are those using different conjunctions to support their distinction).	37
4.2	This table summarizes the order of elements used in TAP-based research prototypes ("research") and commercial tools ("comm.").	44
5.1	Translations in English and Italian of verbal structures and subordinating conjunctions considered in the study.	50
5.2	List of the 24 scenario descriptions paired for the triggering occurrence.	51
5.3	In the table are visible some examples of language primitives (original Italian version and English translation) describing actions and triggers used in our study.	52
5.4	The table reports the absolute and percentage frequencies of time participants choose the five conjunctions in event-based and state-based scenarios.	56
5.5	The table shows the absolute frequencies and percentages of times the participants jointly chose each conjunction with each verbal form.	57
5.6	Frequencies of trigger-first and action-first order preference divided for observation on state- ($n = 144$) and event-based scenarios ($n = 144$).	62
5.7	In the table is visible an example of a state-scenario and an event-scenario paired for the verb used in the trigger part (" <i>to rain</i> ") and the related trigger-action rules (original Italian version and English translation).	68
5.8	The table reports the ANOVA output showing the effect of <i>occurrence</i> , <i>verb</i> , and their <i>interaction</i> on the choice for the specific conjunction.	71

5.9	Post hoc comparisons using the <i>Bonferroni</i> correction. The table presents mean differences (M_{diff}) between conditions, standard errors (SE), and adjusted p -values.	71
5.10	Frequencies for conjunction selection and chi-square (χ^2) values for each comparison in experimental trials (generic vs. specific verb).	74
5.11	Frequencies for conjunction selection and chi-square (χ^2) values for each comparison in control trials (generic vs. specific verb).	75
5.12	The table presents mean RTs for choosing specific vs. generic conjunction (in trials with less than three choices), where n represents the participants' number out of the total sample ($N = 82$). The t -test are reported (<i>Fisher's F-test</i> confirmed no significant difference in variance between the two samples).	75
5.13	The table presents the percentage of accurate answers in event and state trials, divided by conjunction, and the output of two one-way ANOVAs showing the effect of conjunction on the accuracy of identifying the occurrence.	76
5.14	Post hoc comparisons using the <i>Bonferroni</i> correction. The table presents mean differences (M_{diff}) between conditions, standard errors (SE), and adjusted p -values.	76
5.15	The table reports the balanced distribution of state and event topics for each video and a brief description of the elements characterizing it. The duration (in seconds) is also presented for each video.	82
5.16	The table shows the demographic details of our 16 participants (age, gender, education, experience with home automation systems).	86
5.17	The table shows some examples of annotated instructions. We have chosen some instructions related to scenario s2, whose rule prototype is " <i>night</i> [state], <i>rain</i> [event], <i>close window</i> [action]". In the table, "S" refers to the state, "E" to the event, "A" to the action, "conj." to conjunction, "descr." to description, "comm.d" to command.	90
5.18	This table presents the frequency and percentage of rules ($N = 2050$) written in a trigger-first or action-first order, separately for simple and complex rules. It also includes the results of χ^2 tests comparing the orders' distribution in each category.	92
5.19	The table presents in more detail how state-event and action were positioned or omitted in the simple and complex rules.	92

5.20	The table presents the frequency and percentage with which elements were positioned within the complex rules ($n = 1215$). In addition to the action-first and trigger-first order, we classified as "other" those that were trigger-first but with the action placed in the second position. . . .	93
5.21	Chi-square test (χ^2) results for simple and complex rules ("r.") distribution across participants with different levels of expertise in programming. The table includes observed frequencies, χ^2 values, and post-hoc comparisons with <i>Bonferroni</i> correction ($\alpha = 0.02$).	93
5.22	Chi-square test (χ^2) results for trigger-first vs. action-first order distribution across participants with different levels of expertise in programming. The table includes observed frequencies, χ^2 values, and post-hoc comparisons with <i>Bonferroni</i> correction ($\alpha = 0.02$).	95
6.1	The table reports the demographic information of the participants, including their years of experience as teachers (referred to as "teaching") and their use of digital devices in their work as teachers (referred to as "digital devices").	109
6.2	The table lists the nine rules of the correct set of rules to make the "sort numbers in ascending order" game work on <i>SMARTER</i> and the two incorrect rules.	110
6.3	In the table are described the three strategies we highlighted from the qualitative analysis of participants' answers to the debugging task and the two approaches adopted in the error-finding one.	112
6.4	The table shows error-finding task results for each participant.	113
6.5	The table reports the composition task results, including the number of rules created, the presence of errors in the rule-set, and the focus of their reasoning.	114
6.6	Description of metrics used for evaluating rules. For each participant, the aggregate scores of the created rules were calculated (i.e., average rule correctness and rule-set completeness).	120
6.7	Examples of participants' quotes where the "command" metaphor is observable while using concrete primitives and abstract ones.	122
6.8	Examples of participants' quotes where the "react" metaphor is observable while using concrete primitives and abstract ones.	123
6.9	Examples of participants' quotes where the "instruct" metaphor is observable while using concrete primitives and abstract ones.	125

6.10	Metaphors frequency across participants. " <i>As primary</i> " shows how many participants showed the metaphor as their primary reference, and " <i>overall</i> " shows how many participants displayed such metaphor, even when it did not represent their primary reference.	125
6.11	The number of participants who adopted each metaphor per task and the initial primitives used.	127
6.12	The table reports data on participants' demographics, their diagnosis, speaking and typing skills, and familiarity with home automation systems.	145
6.13	The scenarios illustrated in the videos, with the respective themes. . . .	148
6.14	The table presents examples of intent descriptions provided by judges (<i>J1, J2, J3</i>) for participant-generated rules (" <i>rule</i> "). The " <i>comp.</i> " column indicates whether the rule was complete (" <i>yes</i> ") or incomplete (" <i>no</i> "). The " <i>intent</i> " column specifies the interpretation difficulty level. Two scenarios (" <i>sc.</i> ") were selected to illustrate perfect-intent-match (" <i>pim</i> ") with rule prototype: <i>s4</i> (" <i>night, window opens, red light</i> "); <i>s1</i> (" <i>night, person enters bathroom, light</i> "). In bold are rules/description with a <i>pim</i> . . .	153
6.15	The table lists the three prompts used to query ChatGPT. The context table refers to the list of apartment characteristics shown in Table 6.16. .	154
6.16	The table shows the context provided to ChatGPT, including room elements, installed sensors, and smart devices.	155
6.17	The table shows the results of the instruction analysis performed by three researchers.	157
6.18	The table shows the results of the intent analysis of the 56 rules, divided between complete and incomplete. In the lower part of the table we reported the data regarding <i>perfect-intent-match</i> cases divided by level of difficulty of intent interpretation reporting the ratio of judges who had a <i>perfect-intent-match</i> over the total.	158
6.19	The table shows the frequencies with which the codes ($N = 56$) were evaluated for formal and practical correctness and user intent recognition. Specifically, we reported how many times ChatGPT recognized intent, how many times it had a perfect-intent-match (<i>pim</i>) with the original video, and how many of these times it outperformed the human judges. We reported the total counts and divided them for completeness and the level of difficulty in interpreting the rule.	160

6.20	The table reports frequencies for practical and formal correctness of the codes ($N = 56$) when the intent was (or was not) recognized by ChatGPT (respectively, $n_{yes} = 27$ and $n_{no} = 29$), along with χ^2 test results are also reported.	161
6.21	The table shows the averages (M), standard deviations (SD), minimum (min), and maximum (max) of ambiguities (" <i>ambiguity_n</i> ") and questions (" <i>question_n</i> ") per rule resulted from ChatGPT interactions. The results of <i>Mann-Whitney U tests</i> – performed to test for differences between the number of items produced by ChatGPT and the recognition of rule intent – are also reported.	162
6.22	The table reports descriptive statistics and non-parametric tests performed on ambiguities detected and questions generated by ChatGPT considering rules' completeness and clarity of intent.	167
7.1	Examples of game element-based language primitives for events, states, and actions	175
7.2	Participant demographics and diagnostic information, including intellectual disability ("ID") level and verbal production ("verbal") capacity.	178
7.3	The frequency table shows the 39 questions generated distribution among participants and interaction rounds.	182
8.1	Summary of the most adequate linguistic expression to identify triggers that represent events or states.	193
1	The table represents the frequencies of the preferred order divided for scenarios. Some safety-related scenarios – highlighted in bold – made our participants explicitly state and motivate their preference for the reverse order.	232
2	In the table are listed the trigger-action rules (original Italian version and English translation) linked to the 24 scenarios used in our study.	234
3	The table presents RTs for selecting specific vs. generic conjunctions in trials with fewer than three choices (n indicates the number of participants from a total of 82). <i>T</i> -test results for each trial type are reported ("*" marked instances where a <i>Welch's t</i> -test was used due to significant variance differences identified by <i>Fisher's F</i> -test).	242

4	The table presents the frequencies with which rules were written in action-first and trigger-first order for each scenario ("sc." refers to scenario and "n" represents the number of rules analyzed per scenario). Additionally, the table includes the chi-square (χ^2) values for each scenario, comparing the number of action-first vs. trigger-first rules.	243
5	The table reports MQTT messages to interact with the <i>SMARTER</i> OS. In black are commands to send to the <i>SMARTER</i> device, while in blue are the ones sent by the <i>SMARTER</i> OS to the MQTT broker.	248
6	Actions, events, and states implemented in the concrete version of the <i>SENSATION</i> platform.	249
7	Actions, events, and states implemented in the abstract version of the <i>SENSATION</i> platform.	250
8	The table presents the 86 raw codes extracted from the content analysis of teachers' verbal reports and their organization into sub-themes and themes.	251
9	The table shows the results of the normality (<i>Shapiro-Wilk</i>) and homoscedasticity (<i>Brown-Forsythe</i> and <i>Levene</i>) tests performed on the ambiguities detected ("ambiguity_n") and questions generated ("question_n") considering intent recognition ("intent_yes", "intent_no"), for completeness ("complete", "incomplete"), and clarity ("easy", "moderate", "difficult") incomplete rules.	256
10	Ambiguities detected and level of clarity. Post hoc comparisons using the <i>Bonferroni</i> correction. The table presents mean differences between conditions, standard errors (SE), and adjusted p-values.	257
11	User-generated instructions before and after rounds ("r") of interaction with the interface.	258
12	Summary of UEQ-S scores (divided by scales and the global score) collected from the caregivers ($N = 4$).	259
13	Descriptive statistics of UEQ-S score per item (it.). Interpretation guidelines (Schrepp et al., 2017): values between -0.80 and -0.80 represent a neutral evaluation of the corresponding scale, values > 0.8 represent a positive evaluation, and values < -0.8 represent a negative evaluation. The range of the scales was between -3 ("negative pole") and $+3$ ("positive pole").	259

List of Figures

5.1	Two screenshots from <i>Qualtrics</i> platform showing the two tasks performed by the participants. (A) The top part of the figure shows an example of the composition task for the scenario " <i>Having a coffee ready in the morning</i> ". (B) The bottom part of the figure represents the order preference task for the same scenario (light blue boxes contain English translations, and they are not part of the interface).	54
5.2	The image represents the two tasks of the experimental procedure. The left panel represents two trials of the <i>Conjunction Selection Task</i> : one concerning a state-scenario comparing " <i>while</i> " and " <i>if</i> " with a generic verb and one an event-scenario comparing " <i>when</i> " and " <i>if</i> " with a specific verb. The right panel represents two trials of the <i>Scenario Classification Task</i> : one concerning a state-scenario questioning the temporality of " <i>while</i> " and one an event-scenario questioning the " <i>if</i> ".	70
5.3	Comparison of specific conjunction choices for the type of occurrence (state, event) and type of verb (generic, specific).	72
5.4	Number of choices (absolute frequency) for each conjunction for generic (left side) and specific verb form (right side). Panel (A) shows the comparison of generic-specific conjunctions, while panel (B) shows the control comparison (specific-specific conjunctions).	73
5.5	The bar plots represent the average percentage of times participants accurately identified the type of occurrences in the <i>Scenario Classification Task</i>	77
5.6	The figure represents, through a heatmap, the Italian regions of origin of the individuals who participated in our study ($N = 174$).	80

5.7	The figure presents four frames from one of the videos used in the study. The frame (1) represents the event (" <i>the dog goes out</i> ") with the false state (" <i>it is daytime</i> "), which results in no action in frame (2). The frame (3) depicts the same event when the state is true (" <i>it is night</i> "), leading to the execution of the action (" <i>turn on the dog kennel light</i> ").	81
5.8	The bar plots shown in the figure represent the absolute frequency with which the state-event-action elements were presented in the complex rules ($n = 1215$) written by our participants.	94
6.1	The figure on the left side represents <i>SMARTER</i> 2.0 device, while the one on the right shows some mathematical tiles available for creating games on the device.	101
6.2	Deployment Diagram for the architecture of <i>SMARTER</i> 2.0 (cf. Andrao et al., 2023b).	102
6.3	The figure represents <i>SMARTER</i> design (upper images) and implementation (lower images).	104
6.4	The figure presents a screenshot of the compositional visual interface of <i>SENSATION</i>	106
6.5	A collage of the drawings the children gave us after using <i>SMARTER</i> 2.0 at <i>Muse Tech MUSE-TechParty</i> (Trento, Italy, 2023).	108
6.6	The figure shows how the same rule can be created with the abstract (A) or concrete (B) version of the platform.	117
6.7	The figure represents the study setting: the participant is in front of the table with the web-based platform and the tangible device. Behind the table, the video camera captured the interaction with the tools and recorded the verbal interaction. On the left, the participant is represented during the interaction with the platform to create the trigger-action rules, while on the right, during the testing rules on the device.	119
6.8	On the left, the participants' correctness percentages are divided by conditions and task order (abstracts vs. concrete first). On the right, the participants' completeness percentages are divided by conditions and task order (abstracts vs. concrete first).	128
6.9	The image provides a graphical representation of the process, illustrating the transition from the 199 instructions collected in the pilot study (subsection 5.3.5) to the outputs generated by ChatGPT-4 using the two prompts.	137

6.10	Distribution of solved ambiguities by theme (above) and identification code (below).	142
6.11	Example frames extracted for the <i>s1</i> scenario " <i>a person enters the bathroom during the night and the light turns on</i> ". The top-left frame (1) represents the event " <i>a person enters</i> " while the state " <i>during the night</i> " is false (i.e., " <i>during the day</i> "); the top-right frame (2) represents that the action is not performed (i.e., nothing changes); the bottom-left frame (3) represents the event " <i>a person enters</i> " while the state " <i>during the night</i> " is true; the bottom-right frame (4) represents the execution of the action " <i>the light turns on</i> ".	146
6.12	The figure represents a participant writing a rule after observing a video scenario during the study. On the right, an example of a generated rule.	150
6.13	The figure visually summarizes the results of each study phase.	156
7.1	Three screenshots of the <i>SmartGame</i> web interface and a scenario in which <i>SmartGame</i> is connected to two <i>SMARTERs</i> modules are shown in the figure.	174
7.2	In the figure, two screens of <i>RuleCraft</i> are presented. On the left is visible the lateral menu while on the right the trigger-action rule creation page.	175
7.3	<i>RuleCraft's</i> Architecture and Flow.	176
7.4	Interaction flow of a round of interaction with the interface.	179
8.1	The figure graphically represents the distinction and the relation between states ("the door is open" and "the door is closed") and events ("the door opens" and "the door closes").	192
8.2	The figure represents how linguistic clues can be exploited and mapped into the system ontology.	195
1	The figure represents a <i>Miroboard</i> screen where researchers worked to connect, reorganize, and interpret the codes that emerged from the thematic analysis.	231
2	The figure represents the first version of <i>SMARTER</i> designed to support the use of "Cousinarie roads" in mathematics learning for elementary school children.	244
3	The figure illustrates the 6-sided box and the detachable top surface. . .	245
4	The figure represents the detachable top surface of the <i>SMARTER</i> box. .	246

5	The left panel of the figure shows the dimensions of the tiles. The panel on the right shows how the final tiles are composed: connecting a 3 mm-thick drilled rectangle (a) with a 5 mm-thick one (b).	246
6	The figure shows the screen of the <i>Boxes.py</i> box from which we took our cue for the design of the <i>SMARTER</i> Box.	247

Chapter 1

Introduction

Technologies pervade everyone's daily life, from the work sphere to the personal one. Rapid and new technology advancement comes with new systems, updates, technical features, and programming languages. Sometimes, it can be so fast that individuals can get overwhelmed and feel frustrated facing all these changes. In this ever-changing and rapidly evolving world of technology, to avoid asking people to adapt every time to new and more advanced systems, it becomes essential to provide them with techniques and solutions to change their systems and feel even in minimal control of those digital devices.

Empowering people and providing them with solutions and strategies to create, modify, and extend their technology, supporting both the expression of creativity and the attribution of individual meanings to artifacts (Markopoulos et al., 2017), is one of the goals of End-User Development (EUD; Lieberman et al., 2006).

To fully embrace this people-centric vision, it is fundamental to face the challenge of investigating people during their interactions with systems. The moment in which an individual interacts with a system, the person creates mental, transient, and dynamic "images" of this system and its functions (Johnson-Laird, 1983; Norman, 1986). In other words, people develop mental models from their perception of the world while interacting. Individuals' past experiences and personal characteristics influence the interaction and, consequently, the mental models assumed during the interaction with something or someone. Studying mental models and how some may be more functional in guiding people in effective interaction rather than others is a difficult, as well as an interesting mission.

Language is a uniquely human tool for communication and knowledge sharing, enabling the transmission of information and experiences across individuals. It shapes

conveyed information, influencing the creation of individuals' mental models (Zwaan, 2008). Consequently, word choice and syntax affect how information is perceived and determine how effectively concepts are represented in the human mind.

In the context of EUD, the language used in systems, manuals, guidelines, tutorials, and instructional materials provided by designers and researchers directly impacts the mental models users develop when interacting with technology. This manuscript explores the relationship between language and mental models in EUD, examining how linguistic choices can enhance user comprehension and interaction.

1.1 Context

In the last decades, EUD has gained increasing attention as technology is widely used and employed in everyday life and across different domains also by non-programmer users (e.g., home automation, healthcare, education; Barricelli et al., 2019) for working tasks or for managing their home routines. The growing employment of the Internet of Things (IoT; Atzori et al., 2010) – as in home automation systems – requires solutions for empowering non-programmers to manage the functioning of multiple and interconnected elements, potentially making the programming task more complex. Indeed, one of the most studied domains in EUD is the home automation systems (Barricelli et al., 2019; Paternò and Santoro, 2019), where numerous solutions have been proposed as commercial tools (e.g., Rahmati et al., 2017) or as research prototypes (e.g., Ghiani et al., 2017).

EUD proposed several paradigms to achieve its goal, applied to multiple domains (see Barricelli et al., 2019; Paternò and Santoro, 2019 for detailed literature reviews on EUD approaches and domains).

In particular, Trigger-Action Programming (TAP) is an EUD popular approach that enables users to create rules through the specification of an action to be performed upon the occurrence of an event (e.g., Ur et al., 2014). It is widely recognized that TAP may provide an accessible and intuitive way of programming automation because it conveys the fundamental idea in computational logic that certain actions must be taken in particular situations (Dey et al., 2006). Usually, TAP is presented through the *rule metaphor*, allowing users to write or compose a sentence for connecting the trigger to the action. Yet, this rule-based solution has some limitations:

- **Users Expressiveness:** the typical "*If* < single trigger> *Then* <action>" form, visible in popular platforms such as *If This Than That* (IFTTT; Rahmati et al., 2017),

constrains users expressiveness and creativity (Ur et al., 2014) specifically when it comes to defining complex scenarios (e.g., the ones involving IoT technology, developmental, and educational factors). To overcome this limit, numerous authors (e.g., Ardito et al., 2018) proposed solutions involving multiple trigger conjunctions (i.e., enabling users to define the trigger part with one event alongside multiple conditions; Ur et al., 2014).

- **Users' Mental Models:** non-programmer users make numerous mistakes in composing trigger-action rules for handling complex scenarios. It has been hypothesized that user's incomplete and incorrect mental representation of how the system works (Brackenbury et al., 2019; Huang and Cakmak, 2015) – i.e., mental models (Fischer, 1991; Johnson-Laird, 2010; Norman, 2013) – affects interactions with the system and user's performance. Particularly in the context of multiple trigger conjunctions, temporal features of events (e.g., their duration) influence comprehension, interpretation, and the creation of effective and accurate mental models (Brackenbury et al., 2019; Huang and Cakmak, 2015). Dividing instantaneous events (simply *events*) from protracted ones (*states*) – at the end-user interface level and in the internal representation of the system – seems to support users in dealing with trigger-action rules and in assuming effective mental models (Huang and Cakmak, 2015).
- **Language used:** as users often confuse events and states, it has been hypothesized that the "*If* <trigger> *Then* <action>" form may not clearly support the distinction between these two types of occurrences (Huang and Cakmak, 2015). Specifically, the use of the same "generic" conjunction "*if*" for both states and events may hinder users from mentally representing the scenario. The language used implicitly supports the distinction between temporal aspects of states and events (Casati et al., 2000; De Vega et al., 2007; Pianesi and Varzi, 2000).
- **Abstract-concrete content:** non-programmer users' difficulties in interacting with EUD systems may also emerge from the presence of abstract elements in the interfaces (Barricelli et al., 2023; Blackwell, 2002). Many issues and bugs in EUD may be due to the significant gap between the mental models of users and the programming languages used (Hoc and Nguyen-Xuan, 1990), along with users' tendency to apply language knowledge to programming tasks (Bonar and Soloway, 1985).

In conclusion, we suggest that studying linguistics aspects and mental models in depth could shed light on new opportunities in EUD supporting non-programmers in

personalizing and controlling their technological devices.

1.2 Research Objectives and Contributions

From a broader perspective, this thesis aims to study and implement some linguistic aspects to support and guide non-programmer users in assuming effective mental models and reasoning strategies during the interaction with EUD event-driven systems, such as TAP environments. We mainly studied and adopted solutions based on TAP, as it is a popular and intuitive approach used by non-programmer users (Ur et al., 2014; Ur et al., 2016). However, non-programmers often encounter challenges while composing and interpreting trigger-action rules (Brackenbury et al., 2019; Huang and Cakmak, 2015).

This research addresses some of these challenges by exploring how specific linguistic features of the Italian language can facilitate the distinction between events and states, thereby promoting/shaping more effective reasoning strategies and mental representations. Indeed, the language used in TAP – such as verbal primitives, subordinating conjunctions, and syntactic order – can potentially guide users’ mental model assumption and reasoning processes (Clark et al., 2016; Good and Howland, 2017). The research project is structured around two primary objectives:

1. Investigating how different linguistic elements influence the composition, comprehension, and interpretation of trigger-action rules, facilitating the assumption of appropriate mental models (see subsection 1.2.1).
2. Studying users’ interactions with TAP-based systems that exploit linguistic features to understand the potential and challenges of different approaches (see subsection 1.2.2).

The practical goal is to define guidelines for designing and implementing EUD platforms that leverage linguistic cues suitable for non-programmers to support the assumption of more effective mental models.

1.2.1 Objective 1: Language and Mental Models in Trigger-Action Programming

The first objective investigates the role of specific linguistic aspects in the distinct and more effective mental representation of states and events in TAP. Specifically, we were interested in exploring:

- a. Whether some conjunctions ("*as soon as*", "*when*", "*as long as*", "*while*", and "*if*") and verbal structures (generic such as the present tense, state-specific such as the progressive periphrases, or event-specific such as the verbal structure composed of "*it starts to*" followed by the infinitive form) are more fitted for describing states or events;
- b. Whether these linguistic features effectively aid in distinguishing between states and events.
- c. How the order of clauses (containing state, event, or action) influences non-programmers' composition and comprehension of trigger-action rules and their mental models' assumption.

Research Questions

To address this objective, the following research questions were formulated:

- **RQ1:** Do specific language features, such as conjunctions, verbal structures, and clause order, facilitate the creation of effective and distinct mental models for states and events in non-programmer users during the composition and interpretation of trigger-action rules?
- **RQ2:** Which specific subordinating conjunctions and verbal structures are naturally associated with the definition of states and events?
- **RQ3:** Is there a naturally preferred sequence (e.g., action-state-event) that aids non-programmers in composing and mentally representing trigger-action rules?

Methodology

To answer these research questions, three studies were conducted:

- **Study 1: A Qualitative Study on Trigger Interpretation in Trigger-Action Rules.** A qualitative and exploratory study to investigate mental models and the role of specific linguistic aspects - conjunctions, verbal structures, and clause order - in trigger-action rules composition and interpretation (**RQ1; RQ3**).

- Andrao, M., Treccani, B., & Zancanaro, M. (2023). Language and Temporal Aspects: A Qualitative Study on Trigger Interpretation in Trigger-Action Rules. In *End-User Development. IS-EUD 2023. Lecture Notes in Computer Science*, vol. 13917,

pp.84–103. Springer, Cham. https://doi.org/10.1007/978-3-031-34433-6_6

- **Study 2: Investigating Which Linguistic Clues Effectively Guide Non- Programmers.** An online experiment with non-programmers for assessing linguistics preferences for temporal conjunctions and verbal structures to define states and events in trigger-action rules (**RQ2**). This study investigated:
 - I. Which temporal conjunctions and verb forms are most frequently chosen by non-programmers to describe scenarios involving states or events;
 - II. Which conjunctions accurately support the identification of states and events.
- Andrao, M., Treccani, B., & Zancanaro, M. (*In submission*). Making Trigger-Action Rules More Comprehensible: Investigating Which Linguistic Clues Effectively Guide Non-Programmers. *International Journal of Human-Computer Studies*.
- **Study 3: Exploring Natural Linguistic Patterns in Trigger-Action Rules.** An on-line study on linguistic descriptions of rules for defining complex home automation scenarios (involving a combination of one state and one event in the trigger part) by non-programmers and experts (**RQ3**). The study aimed to examine:
 - I. The sequence most commonly used to represent the state-event-action.
 - II. Differences in linguistic patterns between expert and non-expert groups.

Main Findings

The studies revealed that some conjunctions, verbal structures, and specific combinations of these elements better support users in mentally representing and identifying temporal differences between states and events. Specifically:

- Temporal conjunctions such as "*as soon as*" for events and "*while*" or "*as long as*" for states facilitated more accurate mental representations of states and events.
- Conversely, "*if*" and "*when*", although naturally preferred by participants, were found to ambiguously describe both states and events, particularly when combined with the present tense.

- Non-programmers generally preferred *trigger-first* followed by the action order when composing trigger-action rules. For complex rules – where the trigger involves both an event and a state – the naturally preferred sequence is typically *event-state-action*.

Although these findings are specific to the Italian language, they offer valuable implications for designing EUD systems that effectively support non-programmer users in defining trigger-action rules. Specifically, the results contribute to two EUD TAP-based approaches:

- **Form-Based Compositional Visual Interfaces:** The findings suggest implementing linguistic constraints to minimize ambiguous triggers and enhance user comprehension. By guiding users with clearer language structures, these interfaces can foster the assumption of effective mental models and improve rule accuracy.
- **Large Language Model (LLM)-Based Systems:** When users can freely express rules in natural language, the words and terms used are highly variable, increasing ambiguity. These results can contribute to developing integration mechanisms that analyze and regulate the use of conjunctions and verb forms in user-generated rules. This approach would allow the system to detect temporal ambiguities and provide real-time feedback, improving the accuracy of the rules and returning the final rule with unambiguous linguistic aspects.

1.2.2 Objective 2: Studying Interaction with Trigger-Action Based Systems

The second objective aims to study users' interactions with TAP-based systems that exploit language to some extent to guide non-programmer users. Specifically, we studied two approaches: form-based compositional visual interfaces and LLM-based systems¹.

Research Questions

We investigated how a **form-based compositional visual interface** based on specific linguistic aspects (more abstract or concrete descriptions) can guide non-programmer users in defining trigger-action rules fostering or hindering the assumption of effective mental models and reasoning strategies. Specifically, we addressed two research questions:

¹The studies included in Objective 1 and Objective 2 were conducted in parallel and not sequentially. This is why some of the results of Objective 1 were not integrated or investigated in Objective 2 studies.

- **RQ4.** What are the characteristics of users’ mental models when interacting with a “constrained” TAP-based EUD system?
- **RQ5.** Do the linguistic features (abstract vs. concrete descriptions) implemented in a compositional TAP-based interface influence the assumption of different reasoning strategies leading to more effective mental models during EUD tasks?

Besides, we also investigated how **LLMs-based systems** interpret and address natural language written trigger-action rules supporting negotiation and feedback to understand the potentialities and challenges of this approach. We focused on two research questions:

- **RQ6.** Which language ambiguities are identified and addressed by an LLM while interpreting trigger-action rules written in natural language?
- **RQ7.** Which disambiguation questions can an LLM system generate to align with user intent?

Methodology

To answer these research questions, we focused on specific use cases in two application fields:

- **Educational Technology:** Empowering teachers in the customization of their educational tools is essential for enabling them to regain their main role in the definition of teaching-learning paths and exploit educational technology potential (Wiedenbeck and Engebretson, 2004). For this reason, we focused on the application of TAP-based **compositional visual interface** systems in the educational domain to support teachers in the use of technology during their work.
- **Accessibility & Home Automation:** Home automation systems can support individuals with intellectual disabilities in their daily routines (Bacchin et al., 2021). Employing **LLM-based systems**, giving users more freedom to express home automation rules, could be a promising approach for empowering individuals with intellectual disabilities to define home automation. However, ambiguities in the intent expression can grow as language-based communication can be difficult for some individuals with intellectual disabilities (Coppens-Hofman et al., 2017), posing new challenges.

Overall, we conducted four studies:

- **Study 4: Exploring Teachers Mental Models in Composing and Debugging Trigger-Action Rules.** A pilot study to explore teachers' mental models and reasoning strategies assumption during trigger-action rules composition, debugging, and interpretation with a form-based compositional interface (**RQ4**).
 - Andrao, M., Desolda, G., Greco, F., Manfredi, R., Treccani, B., & Zancanaro, M. (2022). End-user programming and math teachers: an initial study. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces* (pp. 1-3). <https://doi.org/10.1145/3531073.3534493>
 - Manfredi, R., Andrao, M., Greco, F., Desolda, G., Treccani, B., & Zancanaro, M. (2022). Toward a better understanding of end-user debugging strategies: A pilot study. In *CEUR WORKSHOP PROCEEDINGS* (Vol. 3172, pp. 31-35). CEUR-WS. <https://ceur-ws.org/Vol-3172/short6.pdf>
- **Study 5: Studying Teachers Mental Models on Trigger-Action Programming.** A qualitative investigation on how different linguistic primitives in the visual compositional interface (abstract vs. concrete) affect the reasoning strategy assumed by teachers, their mental models, and their performance in composition and debugging trigger-action rules (**RQ4**; **RQ5**).
 - Andrao, M., Gini, F., Greco, F., Cappelletti, A., Desolda, G., Treccani, B., & Zancanaro, M. (In press). "React", "Command", or "Instruct"? Teachers Mental Models on End-User Development. In *CHI Conference on Human Factors in Computing Systems* (CHI '25), April 26–May 01, 2025, Yokohama, Japan. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3706598.3713234>
 - Andrao, M., Desolda, G., Greco, F., Manfredi, R., Treccani, B., & Zancanaro, M. (2022). SMARTER: an IoT learning game to teach math. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces* (pp. 1-3). <https://doi.org/10.1145/3531073.3534467>
 - Andrao, M., Cappelletti, A., Desolda, G., Greco, F., Treccani, B., & Zancanaro, M. (2023). SMARTER Than Before: Empowering Teachers to Program a Modular IoT Educational Device. In *IS-EUD Workshops*. <https://ceur-ws.org/Vol-3408/short-s0-01.pdf>
- **Study 6: Exploring How LLMs Handle Linguistic Ambiguities.** An initial inves-

tigation on ChtGPT's ability to interpret and process ambiguous natural language trigger-action rules for home automation scenarios (**RQ6**). Specifically, we investigated:

- I. If the ambiguousness of rules affects the correctness of code generation by ChatGPT;
- II. The type of ambiguities ChatGPT identifies and whether/how they are addressed in the code.

- Andrao, M., Morra, D., Paccosi, T., Matera, M., Treccani, B., & Zancanaro, M. (2024). "This Sounds Unclear": Evaluating ChatGPT capability in translating end-user prompts into ready-to-deploy Python Code. In *Proceedings of the 2024 International Conference on Advanced Visual Interfaces* (pp. 1-4). <https://doi.org/10.1145/3656650.3656693>

- **Study 7: Evaluating LLMs' Ability to Interpret Ambiguous Instructions from Individuals with Intellectual Disabilities.** An investigation, similar to the one of the previous study, collecting home automation instructions from individuals with intellectual disability (Phase 1) and then prompting ChatGPT (Phase 2) to investigate how the system interprets intents and generates disambiguation questions (**RQ6; RQ7**). Specifically, we aimed at exploring:

- I. If the code generated from the ambiguous prompt were correct;
- II. ChatGPT capabilities in interpreting users' intents from ambiguous prompts;
- III. Clarification questions generate ChatGPT to solve ambiguities of the natural language rule.

- Morra, D., Andrao, M., Matera, M., Mores, M., Treccani, B., & Zancanaro M. (*In submission*). LLMs for IoT Programming: A Study Involving Individuals with Intellectual Disabilities. *ACM Transactions on Computer-Human Interaction (TOCHI)*.

Main Findings

From the studies on **form-based compositional visual interfaces**, we found:

- Distinct aspects of mental models influenced individuals' interaction strategies, with some mental representations helping and others hindering effective trigger-action rule creation and testing.

- Abstract descriptions initially posed challenges but encouraged exploration of mental representations often leading to the assumption of suitable and effective mental models for trigger-action programming. Concrete descriptions were easier but reinforced rigid reasoning, sometimes distant from trigger-action logic.

From the studies on **LLMs-based system**, we observed that in the model interpretation of trigger-action rules:

- There were recurring themes of ambiguities detected by ChatGPT, with most not addressed in the code and some arbitrarily handled by the model, often without user involvement.
- The generated clarification question mirrored the identified ambiguities. Most questions sought to refine states and action details. However, a significant number of the generated questions were not inherent to resolve ambiguities or align the generated code with user intent.

1.3 Outline of the thesis

This thesis is divided into two main parts. The first focuses on the Theoretical Background. Specifically, Chapter 2 provides an overview of theories on mental models, Chapter 3 provides an overview of the EUD field, solutions, and tools, while Chapter 4 focuses on factors that can influence mental model assumption in EUD tasks, specifically on language.

The second part of the manuscript describes the Thesis Contribution, specifically, it contains the three studies we carried out on the role of language in the assumption of effective mental representations (Chapter 5) and the four studies investigating the interaction with TAP-based systems (Chapter 6), specifically with a form-based compositional visual interface and an LLM-based system. Chapter 7 contains some preliminary works of research and development based on the results we obtained. Finally, Chapter 8 contains the general discussions, the elaboration of the results we obtained, and the contribution of this thesis, while Chapter 9 the conclusions. The chapters are organized as follows:

- **Chapter 2** provides a general overview of the various theories of mental representations and mental models, highlighting similarities and differences. The Chapter concludes with a description of our conceptualization of mental models, providing guidelines for the terminology used in this manuscript.

- **Chapter 3** provides a general overview of EUD with a specific focus on TAP-based solutions for creating simple or complex trigger-action rules.
- **Chapter 4** discusses three language-related factors that influence mental models assumptions in dealing with trigger-action rules: the linguistic clues used to describe states and events, the order of elements in trigger-action rule, and the level of abstractness of verbal primitives.
- **Chapter 5** describes three studies on language and mental models. Specifically, in the studies, by addressing **RQ1**, **RQ2**, and **RQ3**, we investigate qualitatively and quantitatively the role of conjunctions, verbs, and order of elements in handling trigger-action rules.
- **Chapter 6** describes four studies on interaction with TAP-based systems. Specifically, two investigated users interaction with form-based compositional interfaces involving teachers (addressing **RQ4** and **RQ5**). The other two studies investigated LLMs-based systems, and their application in the home accessibility domain (addressing **RQ6** and **RQ7**).
- **Chapter 7** describes our current directions and specifically a form-based compositional system that we developed and a first pilot study involving individuals with intellectual disability with an LLM-based system prototype.
- **Chapter 8** presents an analysis of the main findings related to each research question we addressed, along with discussions on their implications on the design of EUD TAP-based tools.
- **Chapter 9** concludes this dissertation by summarizing its contributions, discussing its limitations, and outlining possible future research directions.

Chapter 2

Multiple Perspectives on Mental Models

"We rely on mental models of the world. Perception yields a mental model, linguistic comprehension yields a mental model, and thinking and reasoning are the internal manipulations of mental models" (Johnson-Laird, 2004).

Mental representations have attracted the attention of several research fields, such as Psychology (Johnson-Laird, 1989), Human-Computer Interaction (Norman, 1986), Artificial intelligence (De Kleer, 1990), and Linguistics (Van Dijk et al., 1983).

From the literature, it is visible that authors used several expressions to describe the humans' mental representations¹, such as "mental models" (Johnson-Laird, 1983; Norman, 1983), "mental simulations" (Young, 1983), "conceptual models" (Norman, 1986), "situation models" (Fischer, 1991), "small-scale models" (Craik, 1944), or "user's model" (Norman, 1986). Sometimes, these terms overlapped in describing the same phenomenon (e.g., "mental models" and "small-scale models"; Craik, 1944; Johnson-Laird, 1983). Other times, the same term was used to define slightly different constructs. For example, the term "mental models" was used to describe a particular type of mental representation (e.g., the specific and iconic mental representations in working memory that a person forms of a task/situation; Cañas et al., 2001; Johnson-Laird, 1983; Young, 1983). On the other hand, the same term "mental models" was also used to define a mixture of representations involved in a situation (e.g., mental representations of the system, including its functioning, its physical features, the task the user can

¹See Payne, 2007 for further elaborations/discussions on the terminology used.

imagine performing with, analogies with other systems; Norman, 1986; Young, 1983).

In this Chapter, we will strive to structure and differentiate these terms, providing an overview of the history and theories of mental models presented in the literature from different research areas (section 2.1). Our goal is to guide the reader through formal theories of mental models to our conception of mental models, which seeks to unify and differentiate at the same time the insights developed by authors from the research field of Cognitive Psychology and Human-Computer Interaction (section 2.2).

2.1 History and Theories of Mental Models

The concept of “mental models” is generally attributed to psychologist Kenneth Craik², who first introduced this idea in his book *The Nature of Explanation* (Craik, 1944). In the fifth Chapter, the author refers to creating “small-scale models” in the reasoning process. He hypothesizes that humans translate events in the external world into internal models (symbolic mental imitations of reality), manipulate these models to test alternatives and make conclusions, and then retranslate the models into the external world, influencing human behaviors.

From that time on, the study of mental models spanned multiple areas of Psychology (Card, 1983; Johnson-Laird, 1989; Rouse and Morris, 1986), each of which delved into the same phenomenon from different perspectives and research questions. For example, during the same period as Craik, American psychologists also independently reached similar conclusions, using different terms. The American behaviorist Tolman (1948) stated that rats could create “cognitive maps” by investigating their behavior and performance in mazes. Tolman (1948) observed that when a rat learned a specific path to reach a reward, and that path was blocked, the rat would choose a direct route to get to the reward, suggesting that rats learned something similar to a mental map of the environment. Subsequently, cognitivist psychologists focused on understanding the functioning of the “black box” and visual and verbal mental representations (echoing the investigations made in the late 19th century by mentalist psychologists, such as Binet, 1894). Cognitivism investigated the mental rotation of objects (Shepard and Metzler, 1988) and short-term memory stores (visual, verbal, and working memory; Baddeley and Hitch, 1974).

The complexity of the topic and the fragmentation of the research on it contribute

²In the literature, it is possible to identify several historical precursors of mental models, with authors like physicist Ludwig Boltzmann and mathematician-philosopher Charles Sanders Peirce mentioning and elaborating on the idea of human experience mediated by internal representations in the late 19th century. These precursors are well-described by Johnson-Laird (2004).

to the lack of a unified and universally recognized theory (Johnson-Laird, 2004; Payne, 2007). Evidence of this is the terminological confusion surrounding “mental models” that can be noted in two publications in the field of Psychology, both released in 1983. One of the publications, edited by Gentner and Stevens (1983), is a collection of authors (including Norman, 1983) who addressed the same research questions concerning the process of perception and understanding of the external world from rather heterogeneous perspectives (i.e., focusing on different domains of knowledge). The second publication, authored by Johnson-Laird (1983), tackles mental models as a powerful means to study text comprehension, understanding, and inference. While Gentner and Stevens (1983) writing addresses mental models as a macro-topic of cognitive science studies, the book by Johnson-Laird (1983) interprets them as a specific type of mental representation.

In the following subsections, we will describe these different theories on mental models (Johnson-Laird, 1983; Norman, 1983; Young, 1983), focusing specifically on the theory developed by Johnson-Laird (1983), typically studied in Cognitive Psychology (subsection 2.1.1) and on Norman’s conceptualizations, widely adopted in Human-Computer Interaction (subsection 2.1.2).

2.1.1 The Theory of Mental Models in Cognitive Psychology

Mental models are described in the homonymous Theory by Johnson-Laird (1983) as a specific kind of mental representations that iconically represent the *analogs* of instances of the world in working memory (Johnson-Laird, 2010).

The author highlighted the existence of at least three main types of mental representations (Johnson-Laird, 1983; Johnson-Laird, 1989): (i) *propositional representations*, which are strings of symbols that correspond to propositions in natural language; (ii) *mental models*, which are structural analogs of the world; and (iii) *images*, which are the perceptual correlates of the real-world object represented in the minds of individuals, resulting from perception or imagination. The main difference³ among these three mental representations is related to their content. Similarly to images, mental models are **specific**: for example, a person cannot assume a mental model of a general/abstract fruit but only of a specific one, such as a particular red apple. Although their specificity, mental models can represent a **general class** of entities. For example, while listening to the question “Would you like some fruit?”, an individual may create a mental model of

³See Johnson-Laird (1983) and Johnson-Laird (2010) for details on the differences between these three types of mental representations.

a specific red apple, which in this case represents the broader class of "fruits" (including apples, bananas, grapes, oranges, etc.). Although similar, the images and mental models are different, as observed by the study of Knauff et al. (2003) with functional magnetic resonance imaging (fMRI), which detected different neural activations for the images and mental models. At the behavioral level, the authors saw that images slowed the reasoning process (Knauff et al., 2003). This result may be counterintuitive, but it can be explained by the fact that images report details unnecessary for reasoning (Johnson-Laird, 2010).

Grounding on "small-scale models" described by Craik (1944), the Mental Model Theory emerged from Johnson-Laird (1983) attempt to understand human reasoning and specifically understanding and inferences (Johnson-Laird, 1989). The author stated that understanding consists of having functional mental models of the phenomenon in our mind. However, mental models are incomplete and simpler than the corresponding real-world phenomenon. Johnson-Laird (1983) reported an example of mental models of a television. A television can be represented in a person's mind as a box reproducing dynamic images and sounds. Even if this mental model is a simplified and incomplete representation of the television's functioning, it is sufficient for an individual to use it effectively. However, this mental model is unsuitable if the individual aims to repair or create a television. Those who need to fix a television will have more accurate mental models of its functioning, different and less accurate from the one the creator of televisions will assume⁴.

Mental Models Theory⁵ makes three principal assumptions (Johnson-Laird, 2010):

1. Each mental model represents **common aspects** of a set of distinct possibilities. For example, to mentally represent the general category of fruit, the mind builds multiple specific mental models, each illustrating a set of common aspects. An individual might imagine an entire red apple with smooth skin representing edible fruits. Also, this person might have a mental model of an apple cut in half that exemplifies the soft fruit interior and the presence of seeds. These models represent some of the common aspects of fruit (e.g., apples but also pears, cherries, plums, etc.). Thus, instead of recalling all possible fruits, the mind relies on a limited set of specific models conveying fruits' common aspects to make inferences and categorizations.

⁴These differences were also discussed by Fischer (1991) that compared the "*how-to-use-it knowledge*" with the "*how-it-works knowledge*" underlining that the level of accuracy of a mental model is related to the person's goal.

⁵Sometimes, in literature, it is referred to as "model theory" for short (see Johnson-Laird, 2010).

2. Mental models are **iconic**, their structure resembling the one of what they represent as closely as possible. For example, the mental model of a red apple will tend to resemble a real apple rather than something else (e.g., a red ball). This model will maintain spatial and visual relationships similar to the actual structure of the apple, including the stem at the top and the skin with imperfections surrounding the flesh, directly reflecting the physical organization of the represented object.
3. Mental models originating from descriptions represent what is **true** to the detriment of what is false, avoiding working memory overload. For example, a person who reads "the apple is above the table" will create a mental model of the apple above a table. Differently, the person will not form the mental representations of all other possible and alternative states, such as the apple under the table, the apple next to the table, or the apple inside a container, because they are irrelevant according to the given description.

Reasoning often relies on assuming multiple mental models, a concept easily illustrated when making an inference (Johnson-Laird, 1983; Johnson-Laird, 2010). Consider the following premises:

- A. *All the fruit in the basket is edible.*
- B. *An apple is in the basket.*

To infer that "the apple is edible", at least three mental models are engaged: (i) a model representing the overall scene (the basket of fruit), (ii) a model for the specific item (the apple); (iii) a model for the relationship (fruit = edible).

However, as the number of models required for reasoning increases, the process becomes more complex, leading to longer processing times and a higher likelihood of errors (Johnson-Laird, 2010).

In Johnson-Laird's theory of mental models (Johnson-Laird, 1983; Johnson-Laird, 1989; Johnson-Laird, 2010), these are described as dynamic and transient mental representations formed in working memory during interaction with the world. Indeed, the author (Johnson-Laird, 1983) described some constraints on the creation of mental models related to the characteristics of human working memory.

Mental Models Constrains

Johnson-Laird (1989) highlights three constraints that generally delineate the limits of mental model construction dictated by the cognitive features of human beings.

1. **Principle of computability.** According to this first constraint, mental models must be computable. This implies that the construction of a mental model cannot overlook human cognitive limits, particularly working memory limits.
2. **Principle of finitism.** The second principle is based on the assumption that the brain is a finite organism, so a mental model must be finite in its dimensions and cannot directly represent an infinite domain.
3. **Principle of constructivism.** The third principle asserts that a mental model depends on the structural relationships between basic elements, allowing the representation of a potentially infinite number of situations, events, or systems by drawing from a finite base of elements and mechanisms that act on them.

2.1.2 Mental Models in Human-Computer Interaction

As noted by Payne (2007), in Human-Computer Interaction, the term "mental models" describes different aspects of users' knowledge and understanding of systems. Usually, in this context, the expression "mental models" refers to all mental representations that a user forms of a system guiding the understanding and interactions with this given artifact (Norman, 2013; Payne, 2007).

According to Norman (1983), there are several elements and mental models involved in interactions between humans and computers that the author organized into four different elements⁶ (Norman, 1983; Norman, 1986):

- A target *System*: the actual artifact involved in the interaction. Norman (1986) emphasized the role of the *System Image*, which is the physical structure and features of the system that the user interacts with, including instructional materials and documents that the user can consult.
- The *Designer's Conceptual Model* is an accurate and complete representation of how the system works as the one held by designers and developers involved in the system implementation. According to Norman (1986), this conceptual model should include not only mental models of the system but also of the user and the performable tasks (Fischer, 1991).
- *User's Conceptual Model* is a collection of mental models representing the system and its functioning that influence the user interaction with the system. These men-

⁶To facilitate readers' understanding, considering that in the Chapter multiple theories mentioned employ similar or identical names to those used by Norman (1983), we decided to deviate slightly in the terminology used, following the one suggested by the author in a later work (Norman, 1986).

tal representations are based on the user's knowledge of the world, the information extracted from current and past interactions with the system, and how it is described to them (Zancanaro et al., 2022). All these elements compete to develop a representation of the system structure, mechanisms, and operations forming mental models of how the artifact works and is designed. As defined by Norman (1983), these mental models of the system are "*messy, sloppy, incomplete, and indistinct*" and dynamic in evolution, as new experiences and interactions reshape the mental representations of the system. To enable the user to interact with the system effectively, such mental models do not need to be a complete and faithful representation of the artifact (differently from the *Designer's Conceptual Model* held by developers and designers of the said artifact), yet, it must provide an effective representation of its core mechanism.

- The scientist's *Conceptualizations of the User's Conceptual Models*⁷ are meta-mental models derived from the study and observation of the user's mental models during their interactions. These are usually limited conceptualizations, that can't be complete and fully faithful to the actual mental model that the user has. We can imagine it as partial information that researchers and psychologists are trying to organize. This is really difficult as the instruments such as interviews, observation, and other tools from Experimental Psychology are informative but incomplete (Norman, 1983).

In Norman's theory (Norman, 1983; Norman, 1986; Norman, 2013), when considering the user's conceptual model, he referred to several mental models, the ones the user is creating in his working memory, transient and specific of an instance of interaction, and also the ones stored in long term memory. Young (1983), in the previously mentioned book edited by Gentner and Stevens (1983), described these different mental models involved during the system design, the user interaction, and the study of users. The author explicitly adds one element compared to Norman (1983), which is the *Task* the user is involved in during the interaction, which influences and reshapes mental models of the system (Cañas et al., 2001). The *Task* element and the mental models of the specific instance of the interaction are what Johnson-Laird (1983) referred to as "mental models".

⁷Norman (1983) referred to these meta-models also as *conceptual mental models*. However, we decided not to adopt this label as other authors we are referring to in this manuscript employed the name "conceptual model" to address other constructs (e.g., Cañas et al., 2001).

2.2 Different Mental Models: Terminological Disambiguation

From the above paragraphs, and as previously reported by several authors (Cañas et al., 2001; Payne, 2007), a terminological discrepancy emerges when discussing mental models. This terminological issue often leads to ambiguity in research work, especially in multidisciplinary fields, and we believe it is necessary to create clear and contextualized terminology that adequately supports the understanding of this manuscript.

2.2.1 Terminological Ambiguity

We have already pointed out that the term "mental model" has been used by Norman (1983) and Johnson-Laird (1983) with slightly different meanings. In addition, different terms for the same constructs are found in Norman's writings (e.g., Norman, 1983; Norman, 1986) in an attempt by the author to remove the ambiguity of overlapping labels. Norman (1983) refers to mental models as users' mental representations of the system. In a later work (Norman, 1986), the author suggests the use of the term "conceptual" to differentiate mental models related to a system from other meanings associated with this term (e.g., transient and specific mental models formed in working memory defined by Johnson-Laird, 2010). Also, the designer's mental models in Norman (1983) are labeled "Conceptual Models" and "Design Model" in Norman (1986).

Norman (1986) also referred to the fact that the term "User Model" can be ambiguous as it is used to refer to different concepts, such as: (i) multiple mental models that a person forms about how a specific system works; (ii) "generalized" conceptualization of mental models of the "typical" user that designers and developers refer to when designing the system; and (iii) the model that an intelligent system constructs of a specific user (Fischer, 2001).

2.2.2 Unifying and Differentiating

Grounding on the described theories on mental models (subsection 2.1.1 and subsection 2.1.2), we will try to give our terminological definition of mental models. In this manuscript and, specifically hereafter, we will adopt the following terminology:

- **"Mental models"** are a kind of mental representation, and this term will be used as a generic term for referring to all the below constructs.
- **"Conceptual mental models"** (in short, conceptual models) are the set of multiple mental models involved in the representation of a system, created during the

interaction in working memory, stored in long-term memory, and recalled during new interaction with a system. "Conceptual mental models" are the ones described by Norman (1983) involving "Designer's Conceptual mental model", "User's Conceptual mental model", and "Meta-Conceptual mental model".

- **"Working mental models"** (in short, working models) are the specific temporary and dynamic representations created in working memory (i.e., formed and used only during the period of interaction) through the combination of information contained in long-term memory and features extracted from the environment (Cañas et al., 2001). "Working mental models" are those described by the Mental Model Theory of Johnson-Laird (1983).

These different mental models come into play during an interaction with a given artifact. We will provide an example using the terminology proposed in the following subsection.

Different Mental Models in Action: an Example

While performing a task with a particular artifact, users form temporary, dynamic working mental models of their ongoing interaction with it. These *working models* enable them to predict how the artifact will behave in the circumstances described in the task and are maintained in working memory for the duration of the task (Johnson-Laird, 2010). Such representations are constructed by combining incoming perceptual information with information retrieved from long-term memory, including the conceptual mental model of the artifact.

For example, when composing a trigger-action rule for a given scenario, users typically represent the specific circumstances in which the triggering conditions occur and how the system would respond (e.g., the triggering of the action event). They may also consider situations where the triggering conditions do not happen, representing both the absence of these conditions and their outcomes. To compose a rule that achieves the intended goal, users must represent and maintain all possible occurrences and their outcomes in working memory.

The nature and effectiveness of these temporary representations are influenced by several cognitive factors, including the conceptual mental model of the artifact, the way tasks and scenarios are presented to users, and cognitive constraints (see subsection 2.1.1) such as the limits of working memory (Johnson-Laird, 1983; Johnson-Laird, 2010). By examining these temporary, dynamic representations, it may be possible to

infer, at least partially, the users' *conceptual mental model* of the artifact (Norman, 1986).

Working mental models can be established and integrated with conceptual mental models, resulting from multiple interactions and reinforced in time. Understanding users' conceptual mental models is critical to comprehend the interaction between users and the artifact (Cañas et al., 2001; Fischer, 1991; Fischer, 2001; Fischer, 2017; Johnson-Laird, 2010; Norman, 1986). Moreover, incorrect conceptual mental models can lead to errors and biases during interactions (e.g., Yarosh and Zave, 2017). The creation of inaccurate or incomplete conceptual mental models has a strong impact on human-computer interactions because these models are used as interpretive keys for understanding the external world, and once formed, despite being dynamic representations, they are difficult to modify. As widely discussed in Cognitive Psychology literature, changing the information stored in long-term memory can be challenging, as humans tend to re-confirm their beliefs and representation of the world, interpreting different and unexpected outcomes as an exception (Johnson-Laird, 1983). Moray (1987) describes as *cognitive lockup* the situation in which a person anchors himself to his conceptual mental model, ignoring new information he receives. For the same reason, people tend to rationalize unexpected results instead of updating their conceptual models. This phenomenon, linked to *heuristics* and *biases* in human cognitive processes, is also named "creative comprehension" by Lewis and Mack (1982).

Chapter 3

End-User Development through Trigger-Action Programming

In this Chapter, we provide an overview of studies and theories on EUD. As presented in Chapter 1, EUD focuses on studying, designing, and developing solutions that enable users – regardless of their programming expertise – to define how their *digital artifacts*¹ function (Lieberman et al., 2006). In particular, this Chapter focuses and analyses TAP-based solutions (section 3.2).

3.1 End-User Development: Definition and Field Overview

In recent years, digital and smart devices have become increasingly prevalent in our daily environments, unlocking new possibilities for automation across various domains and assisting people in their everyday tasks. However, for automation to be truly effective, it must remain under the user’s control (Lieberman et al., 2006).

Usually, software developers lack expertise in the system application domain, preventing the direct fulfillment of the users’ needs and context requirements. Placing the user at the center of the design phase represents a solution to face this issue (Fischer et al., 2004; Norman, 1986). In addition, as the systems development process can be slow, creating flexible systems becomes fundamental so that users can adapt the system’s functioning during the use phase according to changes in the domain and their needs (Fischer and Giaccardi, 2006).

Unfortunately, non-programmers might struggle to understand and mentally rep-

¹*Digital artifact* refers to both software and hardware artifacts, often interconnected (Barricelli et al., 2019). In this work, it will be used as a general term for describing different pieces of technology.

resent how automated systems operate and how to modify their functioning to fit their needs and express their creativity (e.g., Brackenbury et al., 2019; Huang and Cakmak, 2015).

EUD researchers are addressing the challenge of empowering individuals to understand and employ these technologies effectively (see Paternò and Santoro, 2019, and Barricelli et al., 2019, for a field overview). EUD is **defined** as a wide set of methods and strategies that allow non-programmers to create, modify, or extend a *digital artifact* (Lieberman et al., 2006). The **objective** of EUD is to support users to develop and adapt systems from a perspective that matches their skills. Therefore, it focuses on the systems' *flexibility* and *modifiability* as well as on defining domain-specific development environments (Lieberman et al., 2006).

As pointed out by Fischer (2017), it is essential to involve users in the entire system lifecycle (i.e., in the design, development, use, and re-design during the use of a digital artifact). This is particularly important considering the fast evolution of the socio-cultural environments that may trigger the necessity of extending *digital artifacts* according to the evolving users' tasks, needs, and habits. In other words, EUD strives to place users at the center of the system devising process from the initial design phase to the re-design as new requirements arise (Fischer, 1991).

Indeed, EUD embodies the key concepts of *participatory design*² (Simonsen and Robertson, 2013), emphasizing the centrality attributed to the user in terms of involvement, collaboration, and reciprocal learning with designers (Fischer, 1991).

In contrast to *participatory design*, EUD further extends the active involvement of users by allowing them to redesign and modify parts of their *digital artifacts* during the use phase (Fischer, 1991). To make this possible, methods and tools of *meta-design* (Fischer et al., 2004) can be employed by providing "open environments" that users can change in the design, continuously intersected with their actual use, without requiring people to change the features of their tasks, work, or their priorities.

3.1.1 Related Concepts Definition

In exploring EUD, overlapping nuances with two research fields, **End-User Programming** (EUP; Burnett and Scaffidi, 2013) and **End-User Software Engineering** (EUSE; Ko et al., 2011), are often evident and cannot be overlooked in this overview. Indeed, all EUD, EUP, and EUSE study methods and techniques for empowering end users to

²As described in the book of Simonsen and Robertson (2013), *participatory design* is an approach that originated in Scandinavia in the late 1960s and early 1970s, stemming from social participatory movements where the active role of people is centralized.

modify and create *digital artifacts*. Barricelli et al. (2019) detailed their similarities and differences in a mapping study, while Paternò and Santoro (2019) provided a historical overview of these research fields. Find below a summary of EUP and EUSE definitions and their links with the EUD field.

- **EUP** consists of a set of techniques that enable users to create their own programs (Ko et al., 2011) through programming languages classified as no-code (i.e., where writing code is not necessary) and domain-specific (Myers et al., 2006). EUP is historically “older” than EUD (Nardi, 1993), but it is more limited as the EUD field spans the whole software development process. Nonetheless, EUP is considered the most explored subset of EUD.
- **EUSE** focuses on performing systematic tasks throughout the system’s lifecycle, using techniques widely employed in software engineering to control the quality of the code created by end users³ (Barricelli et al., 2019; Ko et al., 2011).

Although EUD, EUP, and EUSE can be classified as separate, the solutions and tools created by researchers often include facets and nuances that can fit into more than one of them (Barricelli et al., 2019; Paternò and Santoro, 2019). This is another reason why it was decided to include these definitions within this Chapter.

3.1.2 Paradigms and Metaphors Proposed

Several approaches and systems have been proposed to achieve and pursue the goal of EUD of making individuals able to create and modify the functionality of *digital artifacts* (Lieberman et al., 2006).

EUD solutions, also named as “techniques” by Barricelli et al. (2019), can be ground on a mix of more or just one **paradigm**, also referred to as “programming style” by Paternò and Santoro (2019), representing concretely how the EUD environment allows users the creation or modification of a system. Each paradigm can express its developmental constructs through one or multiple **metaphors** (Paternò and Santoro, 2019). Metaphors aim to bridge the gap between users and programming concepts by expressing them in a graspable way for non-programmers. Following, some examples of paradigms commonly used in the field and described in the literature are summarized (for more details, see Barricelli et al., 2019, and Paternò and Santoro, 2019):

³This thesis won’t focus on the EUSE field as it involves writing proper code. Its perspective, different from EUD and EUP, is not on how users can create new functionalities of systems, which is, instead, one of the main motivations of this work.

- **Trigger-Action Programming (TAP)** is a popular event-driven paradigm that enables the definition of relationships between certain types of events and actions within a context. Often, it relies on the *rule metaphor* (e.g., Ghiani et al., 2017), allowing individuals to create trigger-action rules to support the customization of *digital artifacts* behaviors (Barricelli et al., 2019). Even if typically it is expressed in the form of a rule, some solutions proposed a mixed approach (e.g., Mattioli and Paternò, 2020). In section 3.2, TAP-based solutions will be further analyzed.
- **Visual Programming** mainly consisted of applying visual and iconic languages through which users can perform EUD tasks (Paternò and Santoro, 2019). This widely used paradigm is commonly expressed by employing different metaphors, such as *data flow* (e.g., Pot et al., 2009; *Node-RED*), *block-based* or *component-based* programming (e.g., Mørch et al., 2019; Resnick et al., 2009). In distinct ways, these metaphors represent the constructs as 2D or 3D objects, that commonly users can directly manipulate to perform their EUD activities, as visible in famous applications such as *Scratch* (Resnick et al., 2009) – a popular programming environment mainly targeting children. From the mapping study by Barricelli et al. (2019), *component-based* approaches are the most diffused in EUD.
- **Natural Language Programming**, by contrast, employs a subset of constructs expressed in natural language to program *digital artifacts*⁴. This type of paradigm was presented with a “sticky note” interaction metaphor (Perera et al., 2015). Although the idea of programming using natural language was first proposed several decades ago (Sammet, 1966), it has taken years of continuous advances in natural language processing (NLP) to arrive at recent developments in LLMs that have significantly improved the feasibility of this approach for supporting non-programmers. Recently, advances in LLM-based systems have increased the popularity of Natural Language Programming (Tetteroo, 2023), often integrating it with other approaches, such as trigger-action programming (e.g., Gallo et al., 2021).

In this manuscript and, specifically, in this Chapter, we will delve into the details of TAP-based solutions, focusing on rule composition as an interaction metaphor. As motivated in Chapter 1, we devoted to the study of TAP as it is one of the paradigms more used in the EUD field (Barricelli et al., 2019; Paternò and Santoro, 2019). Moreover, previous work repeatedly reported its suitability for supporting intuitively non-

⁴Natural Language Programming is more often associated with EUP than EUD, as described by Barricelli et al. (2019).

programmers in EUD tasks (Dey et al., 2006; Ur et al., 2016), further explained in section 3.2.

3.2 Trigger-Action Programming

TAP-based solutions support the creation of trigger-action rules. In this context, the trigger part refers to specific situations or occurrences detected by sensors or services that, once happened, prompt the execution of the action part (i.e., changing the state of a device or activating a service). The connection between trigger and action in a rule can be expressed through various **composition approach**. These approaches represent different ways of organizing and representing TAP elements and concepts internally in the system and in the presentation to users. This organization influences how individuals interact with systems while handling trigger-action rules affecting their working mental models (Huang and Cakmak, 2015; Zancanaro et al., 2022). Several composition approaches have been proposed in the literature (for an overview, see Andrao et al., 2023a), and some of those are listed and described below:

- **Form-based approach** exploits visual patterns that conceptually bundle functions. Usually, it employs the fill-in of fields combined with Wizard-based solutions⁵. One important advantage of this approach is that it guides the user through a precisely defined path and is thus typically considered easy to use (Dahl and Svendsen, 2011). Such an approach has been implemented into commercial platforms, such as *IFTTT* and *Zapier*, as well as in various research prototypes (e.g., Desolda et al., 2017; Desolda et al., 2021a; Gennari et al., 2023).
- **Conversational approach** enables users to express rules in their natural language, sometimes through LLMs implementation, for example, linking users' natural language input with possible trigger-action rules (e.g., *RuleBot* by Gallo and Paterno, 2022, and *HeyTAP* by Corno et al., 2020). The bots can provide feedback and request additional information to complete the automation, guiding and supporting users if necessary. The conversational approach may be more appropriate than other ones, at least, for users with a poor aptitude towards technologies (Valtolina et al., 2020) and for creating structured rules for handling more complex scenarios (Gallo and Paterno, 2022). Vice versa, form-based and Wizard-like solutions may be more effective for composing simple rules (Gallo and Paterno, 2022).

⁵Wizard-based approach, as described in Barricelli et al. (2019), aims to divide the activities into sequential steps to guide the user in each part of the activity.

- **Block-based approach** consists of creating a rule by linking blocks of various sizes, shapes, and colors by dragging-dropping and manipulating them. EUD tools based on this approach are less constraining than form-based ones, leaving more freedom for users to express their creativity (Gallo et al., 2021). This composition approach combines different paradigms and metaphors, merging the TAP-rule-based solution with the Visual-block-based one. For example, a rule editor block-based was suggested by Mattioli and Paternò (2020), where the trigger features were distinguished and highlighted through a box with a corresponding icon and description.

Regardless of the compositional approach, trigger-action rules can have different levels of complexity (proportional to the number of elements involved in rule definition). Both simple and complex rules have positive and negative aspects that will be discussed in the following subsections 3.2.1 and 3.2.2.

3.2.1 Simple Trigger-Action Rules

TAP is a relatively simple programming model involving the definition of a trigger associated with an action, which is automatically executed upon the occurrence of the triggering event. The most common structure in which trigger-action rules are presented is "*If* <trigger> *Then* <action>", as visible in famous platforms (e.g., *IFTTT*).

Intuitiveness of the *If-Then* Rule Structure

From a usability point of view, the TAP paradigm, specifically if presented with a form-based approach and combined with a Wizard-like one, has the advantage of easily guiding non-programmers in the definition of simple automation (Gallo et al., 2021).

Several studies (Dey et al., 2006; Pane et al., 2001; Ur et al., 2014) reported that the trigger-action rule structure and the TAP logic are intuitive for people independently of their level of expertise in programming. For example, in a study with 20 non-programmers, Dey et al. (2006) collected 371 home automation descriptions in natural language and found that users in most cases (95%) naturally expressed the home automation behavior as a trigger-action rule. The authors observed that every participant naturally and interchangeably used "*if*" or "*when*" to introduce the trigger part.

These results are similar to the ones previously discussed by Pane et al. (2001) in their investigation of how non-programmers naturally express programming solutions. Pane et al. (2001) observed that both adults and children expressed statements using an event-based or rule style, beginning with words like "*if*" or "*when*".

Expressiveness of the *If-Then* Rule Structure

Ur et al. (2014) highlighted that the TAP structure covers most of the desired behaviors for home automation systems supporting people's expressiveness. In an online survey with 318 participants, the authors collected 1,590 desired behaviors for home automation systems, to find that, in approximately 63% of the cases, the descriptions could be easily translated into trigger-action rules. However, they also noted that 22% of these TAP-like descriptions couldn't be translated and implemented in the typical "*If* <single trigger> *Then* <single action>" form. Instead, these cases would require the **conjunction of multiple triggers and actions** to support individuals' expressiveness.

These findings are in line with and supported by other results, such as the ones reported by Dey et al. (2006), previously mentioned, which observed that approximately 23% of the home automation rules written by non-programmers involved Boolean logic ("*or*"; "*and*").

3.2.2 Complex Trigger-Action Rules

Several studies (Bellucci et al., 2019; Brich et al., 2017; Dey et al., 2006; Ghiani et al., 2017; Ur et al., 2014) underlined the need to allow users to create rules with multiple triggers and actions, enabling them a richer expression of their needs and behaviors. Solutions that use multiple trigger conjunctions⁶ in trigger-action rules enable users to define more elements in the trigger part (Ur et al., 2014), for example in the form "*If* <trigger 1> *AND* <trigger 2> ... *AND* <trigger N> *Then* <action>".

Accordingly, several authors studied and proposed TAP-based solutions that allow the conjunction of multiple triggers (Ardito et al., 2018; Bellucci et al., 2019; Desolda et al., 2017; Fogli et al., 2017; Ghiani et al., 2017; Salovaara et al., 2021) and, specifically, the conjunction of one event alongside multiple conditions (Ur et al., 2014).

Creating Complex Trigger-Action Rules

Different studies showed that users successfully write programs with multiple triggers regardless of prior programming experience.

Ur et al. (2014) employed two different prototypes of an IFTTT-like interface in a between-subject study with a total of 266 participants, comparing the performance of the two groups while creating simple rules ("*If* <single trigger> *Then* <single action>") or more expressive rules by composing complex triggers. They observed that users

⁶This possibility of combining multiple triggers or automating multiple actions resembles the Event-Condition-Action (ECA) paradigm used in many programming languages (Casati et al., 2000; Pane et al., 2001; Paton, 1999).

could quickly learn how to use multiple trigger conjunctions independently from their level of programming expertise, as further observed and confirmed by other studies (Bellucci et al., 2019; Ghiani et al., 2017).

Desolda et al. (2017) developed a model that incorporates logic operators (e.g., “and” and “or”), enabling users to specify multiple events along with temporal and spatial constraints related to the rule elements based on the 5W model. This model requires the user to answer five questions: 1) *Who did it?* 2) *What happened?* 3) *When did it take place?* 4) *Where did it take place?* 5) *Why did it happen?* Following an elicitation study with 15 participants, the authors created three user interfaces based on this model. They then conducted a within-subjects study, comparing the performance and satisfaction of participants – 23 with programming expertise and 17 without – when using *IFTTT* or these three user interfaces that support multiple trigger conjunctions. The authors (Desolda et al., 2017) concluded that non-programmers can successfully create complex rules that aggregate multiple triggers.

Understanding Complex Trigger-Action Rules

Dealing with complex and elaborated conditions could be challenging for non-programmer users (Brich et al., 2017), complicating the rules’ understandability (Ghiani et al., 2017; Huang and Cakmak, 2015), leading to errors and misinterpretation – specifically in the trigger composition/interpretation (Brackenbury et al., 2019; Huang and Cakmak, 2015). In particular, it may be problematic for non-programmers when rules are unclearly stated or competing outputs are involved (Huang and Cakmak, 2015). According to some results (Brackenbury et al., 2019; Huang and Cakmak, 2015; Zancanaro et al., 2022), one of the possible causes of ambiguities and errors in rules composition, interpretation, and debugging could be incomplete or incorrect **mental models** (Johnson-Laird, 2010; Norman, 1986), detailed in Chapter 2.

In this context of multiple trigger conjunctions, the **temporal features** of an occurrence (e.g., its duration or its temporal boundaries) that prompts the rule influence the understanding, interpretation, and creation of effective and accurate mental models (Brackenbury et al., 2019; Huang and Cakmak, 2015). In Chapter 4, we will elaborate on these temporal features and the solutions implemented and proposed in the literature to face this issue.

In conclusion, the simple structure “*If* <single trigger> *Then* <single action>” of TAP is natural and intuitive for individuals, supporting to some extent their expressiveness. However, this structure tends to limit people’s possibilities in creating au-

tomation. Its stiffness/rigidity can be experienced as too constrained by users' expressiveness and creativity, therefore it may not be suitable for dealing with complex scenarios (Bellucci et al., 2019; Paternò and Santoro, 2019; Ur et al., 2014).

With this TAP structure, it is impossible to create a rule based on more than one trigger, greatly limiting the possible applications of this approach. For example, the configuration of home automation environments with numerous active automation involving multiple IoT devices, or the control of time sequences may be difficult tasks to handle with simple trigger-action rules (Corno et al., 2019a). Another example of complex scenarios can be retrieved in the educational context where *digital artifacts* may be configured to have different behaviors considering numerous individual factors.

In order to overcome this limitation, several authors (e.g. Ardito et al., 2018; Bellucci et al., 2019; Ghiani et al., 2017) have suggested solutions allowing conjunction of multiple triggers. However, the higher complexity of multiple trigger-conjunctions seems to challenge non-programmers leading to difficulties or bugs, likely arising from the assumption of incorrect mental models of how the system and the trigger-action rule work (Brackenbury et al., 2019; Huang and Cakmak, 2015).

Chapter 4

Factors Influencing Mental Models Assumption in End-User Development Tasks

The processes that guide the formation and construction of mental models (both working models and conceptual ones) are impossible to list exhaustively, just as it would be impossible to list all the products of human imagination. Research in Cognitive Science (Collins and Gentner, 1987; De Kleer and Brown, 1983; Johnson-Laird, 1983) and Human-Computer Interaction (Fischer, 1991; Huang and Cakmak, 2015; Norman, 1983; Norman, 2013; Zancanaro et al., 2022) explored several aspects that can influence the creation and assumption of mental models of users.

According to Norman (Norman, 1983; Norman, 2013), effective mental models of a system should be implicitly induced (i.e., without explicit instructions) by the system design. However, information and instructions provided to users may be crucial as the systems' complexity increases. Studies investigating the effect of different descriptions on participants' performance suggest that more detailed (albeit more complex) information about system functioning allows users to develop better conceptual models. This results in more effective, as well as more satisfactory, interactions with their digital device and more accurate outcome predictions (e.g., Gallitto et al., 2020; Kulesza et al., 2012; Ngo et al., 2020; Zancanaro et al., 2022).

As the language used is one of the prompts that shape our understanding and mental representations of the world (Casati et al., 2000; De Vega et al., 2007; Pianesi and Varzi, 2000), we focused and elaborated on the different linguistic-semantic cues em-

ployed in TAP-based EUD solutions.

In this Chapter, we described some of the linguistic elements that can influence mental models assumptions in TAP-related task: the language used to distinguish events and states (section 4.1), the order of the elements displayed in the rule (section 4.2), and the level of abstractness of the language used (section 4.3).

4.1 Temporal Aspects: Distinguishing Events and States

As first pointed out by Huang and Cakmak (2015), even if enabling non-programmer users to create complex and expressive rules with multiple triggers is feasible and desirable, it seems to introduce issues, errors, and difficulties related to how users mentally represent the scenario. The authors stated that temporal features of occurrences that compose the trigger of rules can be ambiguous and misinterpreted by non-programmers, impacting their understanding, interpretation, and the creation of effective and accurate mental models. For instance, in the rule *"If someone is in the kitchen and it is dark outside, Then switch on the light"* the trigger part can be interpreted in many ways, influencing the prediction of the outcome. One could imagine the light will be switched on: (i) at the exact moment in which someone enters the kitchen and outside it is dark, or (ii) at the exact moment in which the brightness drops below a certain level and someone is in the kitchen. The authors highlighted that it is possible to distinguish at least two types of triggers based on their temporal features:

- **Events:** instantaneous occurrences characterized by temporal punctuality (e.g., the precise moment at which *"someone enters the kitchen"*);
- **States:** prolonged occurrences characterized by the absence of precise temporal boundaries (e.g., the entire time that *"someone is in the kitchen"*) and often referred to in the literature as *conditions*¹(Ghiani et al., 2017; Ur et al., 2014) as these may be TRUE or FALSE, and based on this value the action will or will not be triggered.

In their study, Huang and Cakmak (2015) observed that users often confused events and states and remarked that the lack of distinction between different trigger types may create ambiguities (e.g., interpreting when exactly triggers will occur) and undesirable outcomes. To overcome this problem and support users in assuming effective mental models of how the system and the trigger-action rule work, the researchers

¹In the literature some authors referred to events and conditions (e.g., Ghiani et al., 2017) and others to events and states (e.g., Huang and Cakmak, 2015). Hereafter, we will use "events", "states", and "occurrences" when referring to both of them.

suggested dividing states and events in the internal representation of the system and at the interface level, employing distinct conjunctions and verbs for stressing the temporal differences, clearly distinguishing and grouping them by using matching labels, colors, or shapes.

In the same year, Barricelli and Valtolina (2017) highlighted the importance of temporal aspects in their research. They presented a system for defining rules with expressive triggers in the context of *eWellness*. The system allowed users to create rules composed of multiple elements and define the temporality of occurrences (e.g., enabling users to define whether one element overlapped, occurred during, began, or ended earlier/later than another).

A few years later, Brackenbury et al. (2019) also emphasized the importance of the temporal aspects of occurrences in rules interpretation. In their online study with 153 participants, the authors studied the outcome interpretation of correct vs. buggy pre-written rules considering different temporal paradigms (event-event, state-state, and event-state). The authors used an IFTTT-like visualization of rules, introducing states with "*while*" and events with "*if*" to underline their differences. They found that temporal features are one of the main sources of errors in TAP (named "*flipped trigger*"). To reduce potential ambiguities in the creation and understanding of trigger-action rules, the authors stressed the importance of clearly defining the temporal aspects of triggers.

Other authors in the EUD field (Bellucci et al., 2019; Desolda et al., 2021b; Gallo et al., 2024; Ghiani et al., 2017; Manca et al., 2019; Zancanaro et al., 2022) took into account the temporal considerations described above (e.g., in Barricelli and Valtolina, 2017; Brackenbury et al., 2019; Huang and Cakmak, 2015), providing further support on the importance of temporal aspects in TAP. We will discuss these related works in the sections below.

4.1.1 Internally Mapping Event-State Distinction in the System

Certain authors have implemented the distinction between events and states in the internal representation of the system (Bellucci et al., 2019; Fogli et al., 2017). For instance, the EUD system for home automation described by Bellucci et al. (2019) differentiated states and events internally but did not reflect this distinction in the user platform. In the interface, rules were presented in the format "*If <trigger> AND/OR <trigger>... Then <action>*". However, in the discussion, the authors emphasized the necessity for a clearer distinction between states and events at the user interface level.

Fogli et al. (2017) created an iOS application, called *ImAtHome*, to customize home automation systems by creating unwitted complex rules. The rule composition approach was designed to guide users during the definition of events and states while simplifying the process by hiding the "If <trigger> AND <trigger> ... Then <action>" structure to unwitting the complexity of multiple trigger conjunctions. The authors concluded that non-programmers can handle the creation of rules with multiple trigger conjunctions and that hiding the *if-then* structure can be a feasible solution to avoid users' misinterpretations. However, the study lacks a comparison of the user interface with one that explicitly uses the *if-then* structure.

4.1.2 Exploiting Linguistic Cues to Inform Event-State Distinction

Other research has developed EUD systems that make this event-state distinction visible in the user interface through the use of linguistic and visual cues, aiming to help users assume effective and distinct mental models of states and events (Desolda et al., 2021a; Ghiani et al., 2017; Zancanaro et al., 2022). Similarly, some commercial tools employed different conjunctions for introducing states and events (e.g., *SmartRules*, *SharpTools*).

Indeed, as users often confuse events and states (Huang and Cakmak, 2015), it has been hypothesized that the "If <trigger> Then <action>" form may not clearly support the distinction between these two types of occurrences. Specifically, the use of the same "generic" conjunction "If" for both states and events may hinder users from assuming temporally accurate working models of the scenario. As the language used can foster the distinction between states and events (Casati et al., 2000; De Vega et al., 2007; Pianesi and Varzi, 2000), in the following subsections we will focus and elaborate on the different linguistic-semantic cues employed in TAP-based EUD solution.

From the literature, we identified three combinations of different **conjunctions** to support the distinction of states and events: *if-when* (e.g., García-Herranz et al., 2010; Ghiani et al., 2017), *while-if* (e.g., Brackenbury et al., 2019), and *while-when* (e.g., Desolda et al., 2017; Zancanaro et al., 2022).

However, these are few compared to the linguistic cues retrievable in the language used (e.g., the conjunction "*as soon as*" for events, "*as long as*" for states, and the use of specific verbal structure). Moreover, we observed that very few studies have considered different **verb forms** to elicit temporal differences between states and events (Ghiani et al., 2017; Manca et al., 2019). See Table 4.1 for a summary of the linguistic cues proposed in literature and/or employed in TAP-based interfaces.

Table 4.1: This table summarizes the conjunctions used in research prototypes (“research”) and commercial tools (“comm.”) to introduce states and events in the trigger (in **bold** are those using different conjunctions to support their distinction).

pattern	state description		event description		examples	
	conj.	verb	conj.	verb	research	comm.
<i>if-then</i>	if		if		Bellucci et al., 2019	<i>Atooma</i>
<i>when-then</i>	when		when		Desolda et al., 2017	<i>Assistant</i>
<i>if-when</i>	if	<i>is</i>	when	<i>becomes</i>	Ghiani et al., 2017	
	if		when		García-Herranz et al., 2010	<i>SharpTools</i>
<i>while-if</i>	while		if		Brackenbury et al., 2019	<i>SmartRules</i>
<i>while-when</i>	while		when		Desolda et al., 2021a	

The If-When Rule Structure

A common pattern seems to be *if-when* which was employed by both research prototypes (Gallo and Paterno, 2022; Gallo et al., 2024; García-Herranz et al., 2010; Ghiani et al., 2017; Manca et al., 2019) and commercial tools (e.g., *SharpTools*). Some studies have seen that “if” and “when” are naturally used by non-programmers for problem-solving and “coding” in natural language, as in the work of Pane et al. (2001) and Dey et al. (2006). Using these conjunctions for TAP-based systems may reduce the gap between how non-programmers express themselves and the languages used for EUD solutions.

Ghiani et al. (2017) presented a rule editor that allows multiple trigger conjunctions distinguishing between events and states (named *conditions*) at the interface level and in the internal representation of the system. Users composed the rules combining different occurrences in the form “If / When <trigger> AND/OR/NOT <trigger>... Do <action>”. After selecting temporally neutral occurrences, participants could choose between the keywords “becomes” and “is”: the first for describing events and the second states. An interactive natural language description of the automation was provided, displaying (i) “if” to introduce a state and (ii) “when” to introduce an event. However, interviews with 18 participants who had no programming experience revealed that the keywords “when”, “if”, “becomes”, and “is” were not intuitive or helpful in distinguishing between states and events. Indeed, some participants stated that the language used felt unnatural and overly structured. The study emphasized the need to explore more “natural” approaches to better support end users in defining and understanding rules.

In a following work, the same research group (Manca et al., 2019) modified the interface by enriching it with visual cues emphasizing the difference between states and events, along with a system to support debugging of trigger-action rules. As in the previous version, events and states were associated with specific keywords (conjunctions and verbs). In the old interface, each rule displayed just one conjunction – the one introducing the first element composed. Conversely, in the new version, the rule could contain more than one conjunction (e.g., “If <state ‘is’> AND When <event ‘becomes’> Do <actions>”). In doing so, the differences between states and events in the written rule rely not only on the different verbs but also on the specific conjunctions. From the study with 20 non-developers, the authors (Manca et al., 2019) pinpointed that these visual and linguistic cues somewhat improve rule understanding. However, the paper discussion highlights the need to investigate solutions to better support non-programmers in composing, interpreting, and debugging trigger-action rules.

Recently, some LLMs-based systems adopted the *if-when* paradigm to guide the creation of natural language trigger-action rules (e.g., Gallo and Paterno, 2022; Gallo et al., 2024). Gallo et al. (2024) presented a conversational agent for trigger-action rules, named *RuleBot*, that aims to support users employing a language that differentiates events and states. The system had an “Intent Classifier” trained to identify the triggers from the natural language rule provided by users based on the presence of “if”, “while”, and “when”. Then, the agent reconstructs the rules with a fixed template using the *if-when* structure to return them to the user. In a first study (Gallo and Paterno, 2022), 10 participants were asked to create rules with the system, and the results showed that half of the rules collected with *RuleBot* (20 out of 40) contained an error in the choice between events and states. In another study (Gallo et al., 2024) with 16 participants, the authors recognized that some lexical ambiguities in the user’s natural language text led to an inversion between the event and the state in the rule translated by the system.

Nonetheless, there are contrasting results in employing the *if-when* paradigm. Some authors found that these two conjunctions are used interchangeably by individuals (Dey et al., 2006; García-Herranz et al., 2010), making not intuitive the distinction between states and events grounded on the conjunctions used. García-Herranz et al. (2010) observed that natural language does not make event-state identification easy for non-programmers. As a result of their study with 30 Spanish participants, the authors highlighted that the Spanish words for “when” and “if” (respectively, “cuando” and “si”) are semantically close (as they are in English) and may lead to misinterpreta-

tions and typos among individuals, in particular non-expert programmers.

The While-If Rule Structure

Another pattern occasionally used in the literature is the *while-if*. Brackenbury et al. (2019) employed "*while*" to introduce states and "*if*" to introduce events. Additionally, the commercial platform *SmartRules* adopted the *while-if* paradigm to enable users to create rules for their home automation systems in the form "*If <event(s)> While <state(s)> Then Do <action(s)>*". This combination of conjunctions was also proposed by Huang and Cakmak (2015) in the conclusions of their work on mental models and temporal aspects, as a solution to stress the difference between states and events.

The conjunction "*while*" intrinsically and implicitly contains information about the temporal duration of the occurrence and its simultaneity with another event (Broccias et al., 2022). As a matter of fact, a study by De Vega et al. (2007) found that the "*while*" was more acceptable to introduce a long occurrence (i.e., state) rather than the "*when*" that seems to be suitable for both short and long occurrences (e.g., respectively, "*sending an email*" and "*writing an email*"). The authors observed that the outcomes were more robust in the German language than in Spanish, even if participants for both samples showed a similar pattern.

The While-When Rule Structure

Related work studied and implemented the *while-when* combination of conjunctions to differentiate states and events. It was observed that this approach successfully supports users in the definition of complex trigger-action rules (Desolda et al., 2017; Gallitto et al., 2020; Zancanaro et al., 2022). Zancanaro and colleagues conducted a pilot study with 12 non-programmers (Gallitto et al., 2020), later expanded to include 44 participants in a mixed-methods study (Zancanaro et al., 2022), to compare the *if-then* structure against the *while-when* one. From their findings, the authors argued that using different temporal conjunctions could support users in performing EUD tasks and help them assume effective mental models of states and events. However, in the qualitative part of their research, they discovered that even when using two different temporal conjunctions to introduce events and states, participants still struggled to distinguish them. Desolda et al. (2021a) designed and developed *SENSATION* an EUD authoring tool for creating complex trigger-action rules structured as "*Do <action(s)> When <event> While <state(s)>*". In a within-subjects study, they compared the performance and usability scores of *SENSATION* with *EFESTO-5W* (Desolda et al., 2017), an

interface that allowed multiple trigger conjunctions without using different conjunctions for highlighting the temporal aspects of states and events (i.e., "*When* <trigger> *AND/OR* <trigger>...*Then* <action(s)>"). From the analysis of the responses of the 38 participants, the authors found that for equal usability scores between interfaces, the use of the *when-while* structure seemed more effective in helping users to create correct rules and make fewer mistakes than using the *if-then*.

4.2 The Order of Rule's Elements

Closely related to the semantic linguistic aspects described above, another relevant linguistic aspect is the order of the trigger-action rule's elements, in other words, the sequence in which states-event-actions are presented to users. According to the *Iconicity* assumption (Givón, 1992) and the *Figure-Ground* theory (De Vega et al., 2007), which are Psycholinguistic theories that link the language used and individuals' mental representations, the syntactic order could influence the understanding, interpretation, and mental representations of individuals.

To our knowledge, these theories have not yet been investigated in the context of TAP. However, results from some Psycholinguistics studies (e.g., De Vega et al., 2007) can be easily related to TAP since, similarly, they both deal with temporal sentences that, in the context of TAP rule-based systems, are the verbal primitives describing states, events, and actions.

4.2.1 Psycholinguistic Theories: Iconicity and Figure-Ground

Iconicity is a very well-known and studied theory of Linguistics (for an overview on the topic see Downing and Stiebels, 2012) that addresses how language is mentally represented (or simulated) by individuals (Zwaan, 2008), overlapping and interconnecting with theories and research in Cognitive Psychology (Cooper and Klouda, 1995).

The core of this theory is that the language comprehension process does not generate mental representations of the linguistic input but rather an iconic representation of the situation/meaning of the linguistic stimulus (Zwaan, 2008). In the last century, many authors (Givón, 1992; Haiman, 1980; Landsberg, 1995; Steger and Schneider, 2012) have investigated and contributed to the formalization of the assumptions of *Iconicity*² that links language to the mental representations, in this manuscript referred

²In the literature several types of *Iconicity* are described (e.g., Givón, 1992 and Haiman, 1980). In this context, we consider the one classified as *diagrammatic* (Downing and Stiebels, 2012; Haiman, 1980), which often refers to examples related to the syntactic order of complex sentences of interest in this manuscript. Haiman (1980), referring to the influence of syntactic order in

as *working mental models* (Chapter 2).

According to *Iconicity* assumptions (Givón, 1992; Haiman, 1980), by default, readers/listeners assume that the order of clauses in a sentence corresponds to the actual order of events (Zwaan, 2008). For instance, the order of clauses in a sentence that is almost universally understood is related to the fact that it reflects the actual order of the events described (Downing and Stiebels, 2012).

When dealing with complex sentences (such as those involving occurrences with different temporal features and simultaneity between them), to construct a coherent mental representation of the events described, listeners/readers must be able to interpret connectives (e.g., adverbs and conjunctions) to establish the semantic relationship between clauses (Steger and Schneider, 2012). Steger and Schneider (2012) described three semantic relationships based on (i) **temporality** using, for example, "*after*", "*next*", "*before*", "*when*", "*while*"; (ii) **causality** using for example "*if*", "*because*", "*since*", "*as*"; or (iii) **concession** using, for example, "*although*", "*even if*". Considering **temporality** and **simultaneity** of events (introduced by temporal conjunction such as "*while*" or "*when*"), representations are always more difficult since they violate the natural sequential order of events described by *Iconicity* (De Vega et al., 2007).

For example, De Vega et al. (2004) observed that it was easier for participants to understand sentences that use "*after*" (indicating **sequentiality**) than those that use "*while*" (indicating **simultaneity**). Similarly, the other studies (Münste et al., 1998; Ohtsuka and Brewer, 1992) observed that participants tended to understand better sentences that begin with "*after*" or "*next*" than those that start with "*before*" as the first follows the order of events in real-world, making it easier to be mentally represented. In an *electroencephalogram* (EEG) study, Münste et al. (1998) observed that sentences starting with "*before*" (vs. "*after*") were related to specific differences in *event-related potential* (ERP), which were interpreted as an index of more working memory demands.

Similarly, considering multi-clauses sentence interpretation, De Vega et al. (2007) found that temporal conjunctions indicating **simultaneity** ("*while*" or "*when*") are more easily understood by individuals when presented at the beginning of the sentence rather than embedded in it (i.e., introducing the second clause). For example, the sentence "*while it is night, the alarm rings*" is more easily represented in mental models than "*the alarm rings while it is night*" since the first immediately informs about the simultaneity of the two occurrences, avoiding people's working memory overload. Indeed, according to the *Iconicity* meta-principle described by Givón (1985), a coded experience

comprehension and mental representation, refers to it as "*Iconicity of Motivation*", which is a specific subgroup of the *diagrammatic*.

is easier to store, retrieve, and communicate to others if the code is maximally *iconic* (or *isomorphic* for Givón, 1985), matching the real-world experience.

The *Figure-Ground* theory in Linguistics and Psycholinguistics (Talmy, 1983) is inspired by principles in visual perception, specifically the distinction between the *figure* – the focal element of a scene – and the *ground* – the background element (Rubin, 2001).

According to this theory, the *figure* represents the element being spatially or temporally located, whereas the *ground* serves as the reference frame. A fundamental assumption of the theory is the **asymmetry** between these entities. For example, in locative sentences such as “*The apple is on the table*”, the *figure* (the apple) is typically smaller, more movable, and perceptually salient, while the *ground* (the table) is larger, more permanent, and part of the visual background (Talmy, 1983). This asymmetry has semantic implications; for example, reversing the figure-ground roles in a sentence often results odd or a significant shift in meaning. For instance, “*The apple is on the table*” is more natural than “*The table is under the apple*” (Talmy, 1983).

This figure-ground distinction was extended to temporal sentences (De Vega et al., 2007), where the main clause refers to the *figure* (i.e., the occurrence being temporally located, such as the event “*someone enters the kitchen*”), while the adverbial clause to the *ground* (i.e., the temporal reference, such as the state “*while it is dark outside*”). A study by De Vega et al. (2007) found that when two simultaneous occurrences are presented within the same sentence – such as an event and a state – one is typically perceived as the *figure*, while the other serves as *ground*. Notably, longer occurrences tend to function as the ground and are more naturally introduced by “*while*” rather than “*when*” (De Vega et al., 2007).

4.2.2 Trigger-Action or Action-Trigger?

An overview of TAP-based platforms described in the literature reveals diverse orders of state, event, and action adopted across commercial platforms and research prototypes (see Table 4.2). We identified three major categories based on the sequence in which users can specify rule elements: *Trigger-Action* (e.g., Ur et al., 2014), *Action-Trigger* (e.g., Desolda et al., 2021a), and *Flexible Order* (e.g., Ghiani et al., 2017).

Trigger-Action Order

This order aligns with the *iconicity assumption*, presenting elements in a natural sequential order, i.e., starting with the trigger followed by the action definition. By struc-

turing rules as "*If <trigger> Then <action>*", this order closely mirrors how users perceive cause-effect relationships in the real world, where triggers always temporally precede actions. Pane et al. (2001), studying how non-programmers write instructions, observed that the trigger-action order was the most common and natural for their participants, as the majority of the statements started with conjunctions such as "*while*" or "*if*". This order appears to be the most widely adopted and is implemented in:

- **Commercial platforms:** e.g., *IFTTT*, *AutomateIt*, *SmartThings*, *Homey*, *Hubitat*.
- **Research prototypes:** e.g., Brich et al. (2017), Bellucci et al. (2019), Dey et al. (2006).

In their study, Huang and Cakmak (2015) employed an IFTTT-like prototype using a *Trigger-Action* order. Although the study did not explicitly distinguish states from events, the authors discussed possible ways to differentiate these occurrences and introduced alternative orderings, including *State-Event-Action* (SEA) and *Event-State-Action* (ESA).

- **SEA Order.** Defining the state first follows the *Iconicity assumption* and aligns with De Vega et al. (2007)'s findings. However, among EUD solutions that explicitly distinguish states and events, only a few have adopted the *SEA* order. For instance, in the LLM-based prototype *RuleBot*, Gallo et al. (2021) structured rules using the SEA format ("*If <state> When <event> Then <action>*").
- **ESA Order.** This structure prioritizes the event first, followed by states and actions, integrating *Figure-Ground theory* – which emphasizes highlighting the most salient element (the event) first – while maintaining the core principles of the *Iconicity Assumption* within a trigger-action framework. This ordering is widely observed in:
 - **Commercial platforms:** e.g., *SmartRules*, *SharpTools*, *Assistant*, *Microsoft Power Automate*, *Zapier*.
 - **Research prototypes:** e.g., *EFESTO-SE* (Ardito et al., 2018), as well as works by Zancanaro et al. (2022) and Gallitto et al. (2020), which proposed a "*When <event> While <state> Do <action>*" structure. Additionally, García-Herranz et al. (2010) adopted a similar order with the "*When <event> If <state> Then <action>*" pattern, while Brackenbury et al. (2019) implemented "*If <event> While <state> Then <action>*."

Table 4.2: This table summarizes the order of elements used in TAP-based research prototypes (“research”) and commercial tools (“comm.”).

	order	examples	
		research	comm.
trigger-action		Dey et al., 2006	IFTTT
	state-event-action	Gallo et al., 2024	
	event-state-action	Brackenbury et al., 2019	Assistant
action-trigger		Barkhuus and Vallgård, 2003	
	action-event-state	Desolda et al., 2021a	
flexible		Ghiani et al., 2017	

Action-Trigger Order

While most TAP-based systems adopt a trigger-first approach, an alternative paradigm prioritizes the desired action at the forefront. This ordering is more aligned with *figure-ground* theory, which emphasizes placing the most salient element first – i.e., the intended action, which serves as the goal of the automation rule – before defining the triggering conditions. This order appears to be less common in the literature. To our knowledge, no commercial solutions employ this order, though some research prototypes have adopted it, including Barkhuus and Vallgård (2003) and Barricelli and Valtolina (2017).

Among systems that explicitly distinguish states from events, the only identified example of strict adherence to the *figure-ground* approach is the *SENSATION* platform by Desolda et al. (2021a), which structures rules in an **Action-Event-State** (AES) format.

Flexible Order Composition

Beyond these structured approaches, some systems allow users to define rules starting from either the trigger (without constraints on the ordering of states and events) or the action. For instance, Ghiani et al. (2017) and Manca et al. (2019) introduced a flexible order in their platform *TARE*.

Additionally, LLM-based systems enable high flexibility in structuring trigger-action rules. For example, *RuleBot++* by Gallo et al. (2024) allows users to compose rules freely using natural language. However, despite the flexibility in composition, the system presents the final rule in a fixed ESA structure.

4.3 Abstract or Concrete Content

Research on human reasoning and mental models indicated that some **features of content** people interact with influence how individuals mentally represent scenarios, reason, and draw conclusions (Johnson-Laird, 2010; Markovits, 1988). In the context of a TAP-based system, this content may correspond to the verbal primitives presented to users for composing and debugging trigger-action rules.

Several studies have highlighted that specific content features – as the level of abstractness – can shape the adoption of at least two reasoning strategies related to mental models assumption (Markovits and Lortie-Forgues, 2011; Markovits and Vachon, 1990):

- **Concrete reasoning:** grounded in real-world knowledge and perceptual attributes, relying on tangible, case-specific objects to solve problems.
- **Abstract reasoning:** characterized by pattern recognition and generalization from seemingly specific contexts to identify high-level principles and structures.

For instance, in the home automation domain, a trigger-action rule for activating lights upon detecting movement can be expressed concretely as: *"IF sensor PIR_2 detects a person THEN the living room light.01 turns on"*. Alternatively, a more abstract representation could be: *"IF motion is detected in a space THEN the illumination in that space is activated"*. While abstract formulations enable more flexible and generalizable expressions, concrete rules may be more intuitive for beginners.

Non-programmers often encounter difficulties when interacting with EUD systems, potentially due to the presence of abstract elements in the interface (Blackwell, 2002). Many usability issues and bugs in EUD arise from the gap between users' mental models and the programming languages employed (Fischer, 1991; Hoc and Nguyen-Xuan, 1990; Norman, 1986). In particular, the use of abstract language can introduce ambiguity, as users tend to apply their natural language knowledge to programming tasks, increasing the probability of misinterpretations (Bonar and Soloway, 1985). Recently, Barricelli et al. (2023) proposed the *EUDability* framework and a related assessment scale (Barricelli et al., 2024) for evaluating and designing EUD tools. This framework incorporates the level of concreteness as a key factor influencing user interaction.

Corno et al. (2019b) examined levels of abstraction in automation descriptions, arguing that contemporary IoT device integration primarily relies on a one-to-one, low-level mapping, often constrained by brand-specific implementations. To address this

limitation, they introduced the *EUPont* ontology for EUD in IoT contexts, integrating it into an EUD platform and evaluating its feasibility and comprehensibility through user studies.

Despite the challenges associated with abstraction, it plays a fundamental role in learning and mental model development (Markovits and Vachon, 1990). While abstract reasoning may be more cognitively demanding than concrete reasoning, it is considered a key cognitive skill for programming (Ezeamuzie et al., 2022) and, therefore, it may benefit TAP as well.

Chapter 5

Studying Language and Mental Models in Trigger-Action Programming

As described in Chapter 3, EUD aims to enable non-programmer users to create programs to automate the behavior of their *digital artifacts* (Barricelli et al., 2019). Among several possible solutions (Paternò and Santoro, 2019), TAP is an event-based paradigm in which users can create rules to associate a specific trigger with a particular action to automate the behavior of both hardware and software artifacts (e.g., Ur et al., 2016). These rules are usually expressed in the form “If <trigger> Then <action>”, as visible in research prototypes (e.g., Dey et al., 2006) and popular automation platforms (e.g., *IFTTT*; Rahmati et al., 2017). Although easy enough to understand and be commercially successful, this approach has the limitation of being suitable only for simple automation defined by simple rules with a single trigger (Ur et al., 2014). Previous work focuses, among other aspects, on the distinction between the notion of events and that of states as a way to support the expression of more elaborate conditions in the trigger part of the rule (Huang and Cakmak, 2015; Zancanaro et al., 2022) suggesting to exploit different linguistic aspects to differentiate these two occurrences. For example, using specific and different conjunctions for states and events can help people create different mental models of states and events (as discussed in Chapter 4). A few authors (e.g., Huang and Cakmak, 2015; Manca et al., 2019) also suggested relying on verbal structure features to highlight differences between states and events. In some languages, for example, a different verbal structure may be more appropriate when expressing states rather than events. In Italian, the progressive periphrasis might more accurately describe a state (e.g., “*sta piovendo*” which can be translated into English as

"it is raining") vs. a verbal structure that includes the verb "start" or "begin" emphasizing its punctuality might more accurately describe an event (e.g., "inizia a piovere" which can be translated into English as "it starts to rain").

Referring to theories in Psycholinguistics and Linguistics, other linguistic features that may influence the assumption of more or less accurate and effective working mental models could be the order in which the elements of the state-event-action are presented in the rule (e.g., De Vega et al., 2007).

In the three studies described in this Chapter, we further explored the idea of exploiting lexical choices by analyzing a larger set of linguistic features such as alternative conjunctions, verbal structures, and order of primitives, to understand the role of these linguistic aspects in the distinct and more effective mental representation of states and events in TAP. Specifically, we focused on three research questions:

- **RQ1:** Do specific language features, such as conjunctions, verbal structures, and clause order, facilitate the creation of effective and distinct mental models for states and events in non-programmer users during the composition and interpretation of trigger-action rules?
- **RQ2:** Which specific subordinating conjunctions and verbal structures are naturally associated with the definition of states and events?
- **RQ3:** Is there a naturally preferred sequence (e.g., action-state-event) that aids non-programmers in composing and mentally representing trigger-action rules?

To answer these research questions on linguistic aspects, we designed three studies, described in the following sections (section 5.1, section 5.2, and section 5.3).

5.1 Study 1: A Qualitative Study on Trigger Interpretation in Trigger-Action Rules

The first study qualitatively investigated mental models and the role of specific linguistic elements in the composition and interpretation of trigger-action rules to address **RQ1** and **RQ3**. This study referred to the one presented in Andrao et al. (2023d).

The objective was to explore non-programmers' working mental models while creating and interpreting trigger-action rules. Specifically, we examined the impact of various temporal conjunctions ("as soon as", "when", "if", "while", and "as long as") and verbal structures (progressive periphrases which are analogous to the present progressive tense in English such as "it is raining", specific marks to signal instantaneousness

such as "*it starts to rain*", and the generic forms in present indicative such as "*it rains*"). These linguistic elements were explored for their effectiveness in distinctly representing states and events (Huang and Cakmak, 2015; Zancanaro et al., 2022). Additionally, we investigated how the order of the clause, presenting trigger or action first, influenced the understanding (De Vega et al., 2007).

The study was approved by the Ethics Committee of the *University of Trento* and was conducted during the year 2022.

5.1.1 Participants

A total of 12 participants (seven females and five males) aged between 21 and 33 years ($M = 27.67$, $SD = 3.49$ years; $M_f = 27.43$, $SD_f = 3.41$ years; $M_m = 28.00$, $SD_m = 3.41$ years) took part in the study. All participants were native Italian speakers with no prior programming experience; three were non-degree workers, and nine were university students or recent graduates. Participants were recruited through snowball sampling in the Trento and Brescia areas in northern Italy. The participants received no monetary compensation for participating.

Due to technical issues during recording, data from one female participant were excluded from the analysis.

5.1.2 Materials

All materials were created and presented to participants in Italian. For clarity, we will use the English translation within the text. In Table 5.1, we present the English and Italian translations of the linguistic aspects we are considering. The original Italian versions of the materials are reported in the tables throughout the text.

Scenario Descriptions. Twenty-four (24) scenario descriptions were designed to represent typical daily-life situations that home automation systems could address (see Table 5.2). Each scenario was presented as a concise and impersonal sentence (e.g., "*avoiding clothes getting wet*"). The scenarios set was constructed by identifying 12 different occurrences (e.g., "*to rain*") that supported both a state-based, protracted interpretation (e.g., "*it is raining*"), and an event-based, instantaneous one (e.g., "*it starts to rain*"). Based on these occurrences, half of the scenarios (12) were designed to encourage the creation of trigger-action rules correctly triggered by a state, while the other half by events.

Triggers. We identified linguistic primitives suitable for composing the trigger part. We identified five different **conjunctions**, specifically:

5.1. STUDY 1: A QUALITATIVE STUDY ON TRIGGER INTERPRETATION IN TRIGGER-ACTION RULES

Table 5.1: Translations in English and Italian of verbal structures and subordinating conjunctions considered in the study.

Linguistic aspect	English	Italian
verbal structures	present tense	presente indicativo
	"it rains"	"piove"
	"it starts to" prefix	prefissio "inizia a"
	"it starts to rain"	"inizia a piovere"
	present progressive tense	gerundio presente
	"it is raining"	"sta piovendo"
subordinating conjunctions	<i>if</i>	<i>se</i>
	<i>when</i>	<i>quando</i>
	<i>as soon as</i>	<i>appena</i>
	<i>while</i>	<i>mentre</i>
	<i>as long as</i>	<i>finchè</i>

- "*if*" was selected as it is commonly employed for introducing both states and events (Bellucci et al., 2019; Huang and Cakmak, 2015; Ur et al., 2016);
- "*when*" and "*as soon as*" were selected as they were used or proposed for introducing events (Huang and Cakmak, 2015; Manca et al., 2019; Zancanaro et al., 2022);
- "*while*" and "*as long as*" were selected as they were used or proposed for introducing states (Brackenbury et al., 2019; Huang and Cakmak, 2015; Zancanaro et al., 2022).

Moreover, we defined three distinct **verbal structures**:

- the present tense as a *generic form* supporting both the definition of states and events;
- an *event-specific form* prefixed with "*it starts to*" to elicit the instantaneousness of events;
- a *state-specific form* using the progressive aspect (e.g., "*it is raining*") to underline the protractedness of states (Broccias et al., 2022).

Actions. Each scenario was linked to a corresponding action. In state-based scenarios, actions must be sustained for a specific duration. To convey this ongoing nature, the word "*keep*" was consistently used in the descriptions of these actions. In contrast,

5.1. STUDY 1: A QUALITATIVE STUDY ON TRIGGER INTERPRETATION IN TRIGGER-ACTION RULES

Table 5.2: List of the 24 scenario descriptions paired for the triggering occurrence.

id	type	description	action	trigger
S1	state	avoiding clothes getting wet	umbrella open	<i>it rains</i>
E1	event	avoiding water entering the house	close the window	
S2	state	increasing the stimuli for waking up	volume up	<i>alarm rings</i>
E2	event	having a coffee ready in the morning	turn on coffee	
S3	state	avoiding light entering the room	blinds down	<i>sun shines</i>
E3	event	hanging the laundry outside	send notification	
S4	state	recording Carlo's speech	microphone on	<i>Carlo speaks</i>
E4	event	listening to Carlo's speech	turn down the music	
S5	state	avoiding missing parts of the lesson	recorder on	<i>Prof. teaches</i>
E5	event	avoiding interrupting the lesson	turn on silent mode	
S6	state	working in an optimal environment	temperature at 18°	<i>one works</i>
E6	event	avoiding inappropriate calls	not disturb mode on	
S7	state	increasing concentration in study	door close	<i>one studies</i>
E7	event	reducing distractions	turn off notifications	
S8	state	reducing background noise	noise cancellation on	<i>one runs</i>
E8	event	creating a quiet environment	open the door	
S9	state	max. effort at sports med. exam	heart rate monitored	<i>dog barks</i>
E9	event	recording miles of a workout	activate step counter	
S10	state	responsible traffic circulation	headlights on	<i>car moves</i>
E10	event	listening to a podcast in the car	turn on the radio	
S11	state	alerting people of a fire	fire alarm active	<i>detects</i>
E11	event	preventing a fire from spreading	turn on sprinklers	<i>smoke</i>
S12	state	avoiding household accidents	stove off	<i>one sleeps</i>
E12	event	going to bed safely	activate the alarm	

event-based scenarios highlighted the urgency of actions by using the word "*immediately*" in their descriptions. For example, an action for a state-based scenario is "*keep the umbrella open*", while for an event-based scenario, it would be "*close the window immediately*".

See Table 5.3 for examples of language primitives created for actions and triggers associated with state- and event-based scenarios.

5.1.3 Task and Procedure

Data were collected in a controlled setting: a quiet room with a desk and a computer. The participant sat in front of the computer screen while the researcher sat in a chair on

5.1. STUDY 1: A QUALITATIVE STUDY ON TRIGGER INTERPRETATION IN TRIGGER-ACTION RULES

Table 5.3: In the table are visible some examples of language primitives (original Italian version and English translation) describing actions and triggers used in our study.

id	action	trigger		
		<i>generic verb</i>	<i>state-specific verb</i>	<i>event-specific verb</i>
S1	<i>tieni l'ombrello aperto</i> keep the umbrella open	<i>piove</i> it rains	<i>sta piovendo</i> it is raining	<i>inizia a piovere</i> it starts to rain
E1	<i>chiudi subito la finestra</i> close the window immediately			
S2	<i>tieni alto il volume</i> keep the volume up	<i>la sveglia suona</i> the alarm clock rings	<i>la sveglia sta suonando</i> the alarm clock is ringing	<i>la sveglia inizia a suonare</i> the alarm clock starts to ring
E2	<i>accendi subito la macchinetta del caffè</i> turn on the coffee machine immediately			

the opposite side of the desk. After reading the information sheet, asking the researcher any clarifications, and providing informed consent, the participant began the tasks.

The study was organized into individual sessions in which each participant was asked to read 24 scenario descriptions (see subsection 5.1.2), presented one at a time through the online platform *Qualtrics*. Each trial¹ consisted of two parts: the **composition task** and the **order preference task**. Trials were divided into two blocks and were presented in a fixed order accurately defined to counterbalance between blocks the type of scenarios and occurrences, ensuring that in each block there were six state- and six event-based scenarios and that an occurrence (e.g., "*it rains*") was presented just one time in each block.

In the **composition task**, participants were asked to carefully read the scenario description and choose the linguistic components (conjunctions and verb structures) to create the most appropriate sentence, which would function as instructions for achieving the proposed scenario. As shown in Figure 5.1, participants read the scenario on the screen, with the action part presented immediately below. Two drop-down lists

¹In experimental psychology, participants are often asked to perform the same task multiple times, with each repetition referred to as a "trial", as delineated by APA's guidelines.

allowed participants to select a conjunction (from five options) and a verb structure (from three options) to formulate the rule.

Subsequently, in the *order preference task*, participants were asked to choose the order they thought was the most natural and accurate for composing an instruction to realize the scenario. On this second screen, participants viewed their selected options presented in two formats: trigger-first (e.g., "*if it rains close, the window immediately*") and action-first (e.g., "*close the window immediately, if it rains*").

Finally, participants completed a brief demographic form indicating their age, gender, and previous experience with programming languages.

Using a *thinking aloud* protocol (Boren and Ramey, 2000; Russell and Chi, 2014) participants were encouraged to vocalize their motivations and elaborate on their decisions. The researcher was instructed to facilitate this process without interrupting the participant's flow of thought, eliciting their working mental models. Each session lasted approximately 45 minutes and was both video- and audio-recorded.

5.1.4 Data analysis

The session recordings were transcribed, then coded and analyzed using inductive thematic content analysis (Braun and Clarke, 2006) by two independent researchers. After individually coding all verbal reports, the two researchers compared the codes that emerged and discussed the differences till an agreement was reached. Working jointly, they grouped the codes into themes. See Appendix A for the raw graphical representation of the work made by the two researchers in connecting, reorganizing, and interpreting the codes using *Miro*.

To contextualize the main qualitative results, descriptive statistics were calculated, specifically focusing on the frequency of choices made by participants.

5.1.5 Results

A thematic analysis of participants' verbal reports helped identify four main themes to understand the relationship between lexical choices and participants' working mental models of the programming activity.

Theme 1: "*As soon as*" effectively highlights the instantaneousness of events, especially when associated with the verbal structure "*starts to*".

Participants identified the conjunction "*as soon as*" as the most precise and accurate way to describe the timeliness of events. Indeed, this conjunction was the most se-

A. Composition task

[1] Componi la frase nella maniera che ritieni più accurata per realizzare lo scenario.
Ricorda di "pensare ad alta voce".

[1] *The instructions displayed: "Compose the sentence in the way you think is most accurate to realize the scenario. Remember to "think aloud".*

[2] Avere un caffè pronto al mattino

[2] *The scenario description: "Having a coffee ready in the morning".*

accendi subito la macchinetta del caffè	[3]	
---	--------------------------	----------------------

[3] *Dropdown to select one conjunction among: "as soon as", "when", "if", "while", and "as long as".*

accendi subito la macchinetta del caffè	se	[5]
---	----	--------------------------

[4] *The action: "turn on the coffee machine immediately".*

[5] *Dropdown to select one verbal structure among: "the clock alarm rings", "the clock alarm starts to ring", and "the alarm clock is ringing".*

B. Order preference task

[6] Seleziona l'ordine che preferisci per lo scenario: Avere un caffè pronto al mattino

[7] se suona la sveglia accendi subito la macchinetta del caffè

[8] accendi subito la macchinetta del caffè, se suona la sveglia

[6] *The instructions displayed: "Select the order you prefer for the scenario: Having a coffee ready in the morning".*

[7] *Trigger-first order "if the alarm clock rings turn on the coffee machine immediately".*

[8] *Action-first order "turn on the coffee machine immediately if the alarm clock rings".*

Figure 5.1: Two screenshots from *Qualtrics* platform showing the two tasks performed by the participants. (A) The top part of the figure shows an example of the **composition task** for the scenario "Having a coffee ready in the morning". (B) The bottom part of the figure represents the **order preference task** for the same scenario (light blue boxes contain English translations, and they are not part of the interface).

lected in event-based scenarios by our participants (see Table 5.4). This conjunction supported participants mentally representing a defined instant, as some participants described:

- Participant 3: *"'As soon as' makes me think of it as immediate because it is a gesture, an immediate activity"*.
- Participant 9 [referring to scenario n. E3]: *"The word 'as soon as' makes me think of an instant [...] the instant when the sun comes out"*.
- Participant 10 [referring to scenario n. E2]: *"'As soon as' [...] even though maybe the alarm clock rings for a few seconds, I interpret it as the first second it rings"*.
- Participant 12: *"As soon as this thing happens, so at that precise moment"*.

The conjunction *"as soon as"* appeared to give some participants confidence that the action would be carried out effectively, as explained by Participant 8:

- Participant 8 [referring to scenario n. S4]: *"'As soon as' gives me more confidence in having recorded everything that Carlo has to say."*

This conjunction was connected frequently with the verbal structure *"starts to"* (event-specific) to strengthen (see Table 5.5 for the frequencies in which participants associate each conjunction with each verbal form). For example, some participants motivated their choices by illustrating:

- Participant 5 [referring to scenario n. E9]: *"because it is 'as soon as one starts to run', immediately activates the step counter [...] to strengthen the sentence"*.
- Participant 8 [referring to scenario n. S3]: *"I would choose 'as soon as the sun starts to shine' [...] I need to do the action as soon as the first ray of sunshine enters the room"*.

The implicit link between the verbal structure *"starts to"* (event-specific) and the conjunction *"as soon as"* influenced Participant 11 choices:

- Participant 11 [referring to scenario n. E5] *"Again, implicitly I add 'the Professor starts to teach' because there is the 'as soon as'"*.

For the same participants, this association felt redundant and unnecessary if other factors (e.g., the verb' semantics) already convey the same temporal information:

5.1. STUDY 1: A QUALITATIVE STUDY ON TRIGGER INTERPRETATION IN TRIGGER-ACTION RULES

- Participant 11 [referring to scenario n. E2]: *"Always 'as soon as' because it tells me the beginning of the action [...] here I would even put without the 'it starts to' but just 'as soon as the alarm rings' because it gives me the idea of a shorter duration this thing [...] the alarm clock lasts up to 10 seconds. If I say, 'as soon as' is enough for me to tell 'as soon as it starts to ring'".*

Table 5.4: The table reports the absolute and percentage frequencies of time participants choose the five conjunctions in event-based and state-based scenarios.

conj	occurrence	frequency	percent
as soon as	event	87	89.69
	state	10	10.31
	total	97	100.00
when	event	29	48.33
	state	31	51.67
	total	60	100.00
if	event	20	54.05
	state	17	45.95
	total	37	100.00
as long as	event	2	7.41
	state	25	92.59
	total	27	100.00
while	event	6	8.55
	state	61	91.05
	total	67	100.00

For the most, the verbal structure *"starts to"* (event-specific) effectively represented the precise instant when the action needed to be performed. Almost all participants stressed this aspect multiple times. While selecting primitives in the *composition task*, some participants reported:

- Participant 2 [referring to scenario n. E4]: *"in my opinion, rather than 'as soon as Carlo speaks', 'as soon as Carlo starts to speak', it is already more precise".*
- Participant 10 [referring to scenario n. E4]: *"Because it is the instant when Carlo starts to speak [...]. So for me, it's instantaneous".*

5.1. STUDY 1: A QUALITATIVE STUDY ON TRIGGER INTERPRETATION IN TRIGGER-ACTION RULES

Some participants related the verbal form with the need to promptly execute the action, suggesting that lexical choice influences the criteria for execution actions in the trigger-action rule:

- Participant 4 [referring to scenario n. E6]: *“‘One starts to work’. This is the sentence construction when you want the action immediately”.*
- Participant 8 [referring to scenario n. E2]: *“‘It starts to ring’ gives me the idea of something you have to do instantaneously”.*
- Participant 9 [referring to scenario n. E5]: *“In this case, because it’s an action I have to do immediately, promptly, as soon as the Professor starts to teach, I have to put the silent mode. Yes, it seems that adding the verb to start gives me even more urgency.”*

Table 5.5: The table shows the absolute frequencies and percentages of times the participants jointly chose each conjunction with each verbal form.

conj	verb	frequency	percent
as long as	event-specific	0	0.00
	state-specific	8	29.63
	generic	19	70.37
as soon as	event-specific	86	88.66
	state-specific	1	1.03
	generic	10	10.31
if	event-specific	10	27.03
	state-specific	8	21.62
	generic	19	51.35
when	event-specific	16	26.67
	state-specific	7	11.67
	generic	37	61.67
while	event-specific	1	1.49
	state-specific	42	62.69
	generic	24	35.82

Theme 2: "While" effectively highlights the protractedness of events (especially when associated with the progressive form) and "as long as" seems to be more related to the end of the action.

The conjunction that best expresses the idea of duration and continuity seems to be the "while" (see Table 5.4):

- Participant 4: *"When I read 'while', for me, it is always something that lasts overtime".*
- Participant 9 [referring to scenario n. S12]: *"It has to last the whole time I sleep. So 'while' seems the most appropriate to me".*
- Participant 11: *"I would use 'while' to be even more specifically protracted".*

Similar to what emerged for "as soon as", the "while" was often associated with the occurrence-specific verbal structure, in this case, the progressive form, as it strengthened the notion of duration (see Table 5.5). For instance, comments from participants 3, 4, and 5 included:

- Participant 3 [referring to scenario n. S7]: *"In this case, 'keep the door close' is more continuous to me, so 'while' is doing an action".*
- Participant 4 [referring to scenario n. S1]: *"Maybe 'it is raining' [works better]. It gives more the idea of continuing to do the action".*
- Participant 5 [referring to scenario n. S4]: *"Carlo is speaking. [...] I could also select 'while Carlo is speaking' but 'while' and 'is speaking' give the same sense to the sentence [...] to reinforce the idea of temporality".*

However, Participant 12 preferred not to select the progressive form and to use the present tense (the generic form) because, according to him, other elements of the sentence already enclosed the concept of protractedness:

- Participant 12 [referring to scenario n. S6]: *"'While one works' and not 'is working' because it seems a repetition to me because 'while' already gives me the idea of something protracted".*

Overall, several participants (2, 3, 4, 5, 8, 9, 11, 12) recognized the progressive form as strengthening the representation of a prolonged occurrence, for example:

- Participant 11 [referring to scenario n. S3]: *"The sun shines for a duration of time, I would say 'it is shining' because it's a protracted action".*

This was the verbal structure to ensure the action would be executed even after it began, as Participant 3 pointed out:

- Participant 3 [referring to scenario n. S10]: *"More than 'it starts to move', which means that after you start you can turn off the lights, 'It is moving' gives the obligation to hold them while you are moving"*.

The conjunction *"as long as"* made it easier for participants to represent a prolonged action, but it also seemed to suggest or emphasize the end of the action, differently from *"while"*. For example:

- Participant 5 [referring to scenario n. S3]: *"'As long as the sun shines'. Because 'while' is during, 'as long as' gives it an end"*.
- Participant 4 [referring to scenario n. S7]: *"'As long as' is convenient here. [...] 'as long as' gives the idea that until you finish studying, you have to keep the door closed"*.
- Participant 7: *"Instead of something that is happening, 'as long as' gives me the idea that is going on in time together with the idea of the end"*.

Participant 3 related *"as long as"* with the presence of other rules assumably conveyed in the semantics of the conjunction:

- Participant 3 [referring to scenario n. S1]: *"It could be: 'as long as it rains'. I basically assume that at one point one opened it [the umbrella] when it started raining, and this [the rule he is composing] is after that moment. Before, there was 'open the umbrella as soon as it starts to rain' and now 'keep the umbrella open as long as it rains'"*.

Moreover, *"as long as"* led participants to imagine the situation in which the rule might not be triggered. Some examples:

- Participant 8 [referring to scenario n. S8]: *"As soon as it finishes barking, I'll turn off the noise cancellation mode"*.
- Participant 10 [referring to scenario n.S7]: *"'As long as', so that then I can also open the door if I'm not studying anymore"*.
- Participant 12 [referring to scenario n. S1]: *"'As long as it is raining', if it's not raining, you don't get wet so you can close the umbrella"*.

Theme 3: "If" and "when" are ambiguous and they do not consistently support the distinction among events with different temporal aspects; similarly, the present tense does not support the definition of temporal aspects and therefore may generate ambiguity.

Both "if" and "when" were used indistinctly to define instantaneity and continuity of occurrences depending on the context and other temporal factors of the rule, including the semantics of the verb associated with the trigger and the duration of the action (see Table 5.4). The ambiguity of the conjunction "when" is pronounced as participants used it to describe both timeliness and duration, often unconsciously. For example, initially, Participant 2 expressed:

- Participant 2 [referring to scenario n. E4]: *"'When' means as soon as he starts talking"*.

Later in the study, he contradicted himself by stating:

- Participant 2 [referring to scenario n. S1]: *"'When it rains' means something protracted in time because it is not 'as soon as it starts raining'"*.

Some other participants explicitly described their issues in using "when":

- Participant 9: *"I feel like I contradict myself"*.
- Participant 10 [referring to scenario n. E4]: *"When Carlo speaks, immediately turns down the volume of the music, for me this is tricky"*.

Participants 11 and 4 were able to realize that "when" does not help people in defining occurrences temporally:

- Participant 11 [referring to scenario n. E2]: *"When the alarm clock rings, however even 'when' is quite unclear"*.
- Participant 4 [referring to scenario n. E2]: *"'When the alarm clock rings'. It doesn't sound like something that happens immediately or an ongoing thing. It's in the middle"*.

Analogous observations were made regarding the conjunction "if". For instance, Participant 9, who had associated "if" with "as soon as" due to its connotation of immediacy, later commented:

- Participant 9 [referring to scenario n. E12]: *"Instead, in this case, it's different. [...] because 'if' with the verb 'sleep' sounds like something long. So the 'if' changes according to the verb that comes after."*

Further participants expressed uncertainty, confusion, and difficulties regarding the temporal representation of "if":

- Participant 3 [commented at different moments]: *"The ones with the 'if' are the most tricky [...] the if already puts you in an uncertain, doubtful form [...]. In all of them [scenarios] 'if' might also work, however, without giving certainty"*.
- Participant 4 [commented at different moments]: *"The 'if' indicates a condition of doubt and uncertainty [...] the 'if' associates well with all scenarios [...] I never used 'if'. The 'if' is too doubtful"*.

The conjunction "if" was found to be unhelpful for mentally representing the rule, as it was perceived as very general. For example, Participant 10 stated:

- Participant 10 [referring to scenario n. S3]: *"This one is difficult. [...] 'If the sun shines' it could be even for a few hours"*.

Nevertheless, the flexibility and temporal imprecision of "if" is sometimes preferable in situations of uncertainty because it limits the rule's temporal boundaries. For example:

- Participant 5 [referring to scenario n. E10]: *"It could be 'if the car moves'. Because for listening to a podcast in the car, the 'if' is the option that temporally limits you the least, [...] 'if it moves', when you want you can listen to it [the podcast]"*.

The present tense for the event verb did not help in assessing its temporal definition, as Participant 8 said:

- Participant 8: *"'One runs' [stressing on the present indicative tense] is the concept that meets the 'beginning', the 'while' and the 'end'"*.

Some participants clarified how the present tense verb defines general situations. Participants 7 and 8 explained:

- Participant 7 [referring to scenario n. S8]: *"'When it barks' not 'when it is barking' because it could bark at any time"*.
- Participant 8 [referring to scenario n. E2]: *"'It starts to ring' gives me the idea of something you have to do instantaneously, and 'it's ringing' gives me the idea of a period [...] 'it rings' instead is a broader concept that gives me more space"*.

Theme 4: The temporal order with the trigger at the beginning of the rule enhances understanding, but the most critical element to convey (mainly the action) should also be placed at the beginning of the sentence.

Some participants expressed a preference for the order of the trigger and action components to reflect the actual sequence of events. They preferred having the trigger placed at the beginning of the rule, regardless of whether it described a state or an event:

- Participant 5 [referring to scenario n. E2]: *"I would always select this way [trigger-first], with the right temporal order. That first, it happens that the alarm rings, then I turn on the coffee machine"*.
- Participant 10 [referring to scenario n. S5]: *"Again the sentence beginning with 'as long as' [...] it makes me understand that I have to keep it on for the whole duration"*.
- Participant 9 [referring to scenario n. E1]: *"Because from the temporal point of view, first, it rains, and then I close the window"*.

Inspection of the frequencies of the choices between trigger-first and action-first – reported in Table 5.6 – shows that the preferred order was trigger-first, both for states (79.90%) and events (71.53%).

Table 5.6: Frequencies of trigger-first and action-first order preference divided for observation on state- ($n = 144$) and event-based scenarios ($n = 144$).

occurrence	order	frequency	percent
event	action-first	29	20.14
	trigger-first	115	79.90
state	action-first	41	28.47
	trigger-first	103	71.53
total		288	100.00

Participants 11 and 10 explicated their decisions by considering the logical consequentiality of trigger and action:

- Participant 11: *"Because it gives me the indication first of the condition and then the action I have to do in this condition. I don't care to know the action if I don't know the condition under which this action should occur"*.
- Participant 10 [referring to scenario n. E1]: *"Because it gives me more of an idea of why I have to close the window. So it makes you understand immediately"*.

It emerged from the participants' words that the elements of the sentence should be arranged according to their importance. Some participants indicated that the trigger was the crucial part because, without it, the action would not be performed, as noted by Participant 4:

- Participant 4 [referring to scenario n. E1]: *"Because it gives you the indication, 'as soon as it starts to rain', and then close the window. It seems more important to me to say the first part."*

However, some participants argued that action is the main component and should be placed at the beginning. Following are some quotes from the verbal reports, as an example:

- Participant 7: *"The important thing is the action"*.
- Participant 8: *"The main diktat [i.e., order or imposition] is 'close the window'. And then there is the part of explaining why. But in the meantime, it [the action] directs you to do something"*.
- Participant 10 [referring to scenario n. S6]: *"This one [the action-first order] because it gives us a better understanding of what you need to do, which is to 'keep the temperature at 18 degrees'. Then, it is less important to know when, while you are working or studying"*.
- Participant 12 [referring to scenario n. E10]: *"But do I care more about the car moving or listening to the radio? 'Listening to the radio' is what interests me. [...] Because normally when people listen to sentences, they lose interest at the end"*.

The urgency to prioritize action in certain rules was more evident in scenarios involving potential danger or harm to safety, such as fire (scenarios S11; 33.33%), road traffic (S10; 41.67%), and home security (S12; 41.67%).

In these instances, more participants preferred to set the action at the beginning of the rule compared to other scenarios (see Appendix B). By placing the action at the start of the sentence, it becomes the focal point, while the trigger remains in the background, as reported by:

- Participant 3 [referring to scenario n. S10]: *"In this case, however, I would put 'keep the headlights on' first because it seems more important to me [...] the action of keeping the headlights on that [...] is always a safety issue"*.

- Participant 9 [referring to scenario n. S11]: *"I would imagine that the alarm siren should sound during the entire period of smoke detection. In that case, I would change my mind [in the other scenarios she preferred the trigger-first order] because the first part of keeping the fire alarm on seems more important to me".*

5.1.6 Discussion

While still preliminary, the findings of this qualitative study indicate that the three lexical aspects analyzed – conjunctions, verb forms, and order of elements – all play a role in shaping users' working mental models (**RQ1**). Certain combinations enhance the working model's accuracy (**RQ2**), whereas others introduce temporal ambiguity, potentially leading to interpretative errors.

Specifically, consistent with the observations of Huang and Cakmak (2015), our results show that using distinct temporal conjunctions for events (e.g., *"when"* or *"as soon as"*) and states (e.g., *"while"* or *"as long as"*) aids non-programmers in distinguishing occurrences with different temporal features effectively. Although the conjunction *"if"* is commonly employed in trigger-action rules (Rahmati et al., 2017), prior research (Desolda et al., 2021a; Gallitto et al., 2020; García-Herranz et al., 2010; Zancanaro et al., 2022) highlights its ambiguity. Our findings corroborate this, as participants perceived *"if"* as ambiguous, associating it indiscriminately with events and states. Recent studies advocate *"when"* and *"while"* for events and states, respectively (Desolda et al., 2017; Gallitto et al., 2020; Zancanaro et al., 2022). Our results support *"while"* as suitable for ongoing occurrences, especially when paired with progressive verb forms. Conversely, *"when"* appears less precise for instantaneous events, aligning with conclusions from García-Herranz et al. (2010), while *"as soon as"* better conveys instantaneous occurrences, particularly alongside *"starts to"*. The conjunction *"as long as"* evokes a clear end-oriented representation, which, although sometimes advantageous, may also cause errors.

Similarly to the results of other studies Pane et al. (2001), our findings demonstrate a preference for placing triggers at the start of rules, aligning with the *Iconicity assumption* (Givón, 1985), where sequence order mirrors event chronology (De Vega et al., 2007). Conversely, some participants opted to prioritize the action in contexts involving personal safety, coherently with the *Figure-Ground theory* statements De Vega et al. (2007). These results on clause order suggest that flexible ordering, as proposed by Manca et al. (2019), would enhance user choice when creating trigger-action rules.

Limitations

Apart from the limitations shared with the other studies described in section 9.1, this qualitative study has several specific limitations. The sample size of 11 participants is small. A larger sample could have revealed additional perspectives and enriched our findings. Second, the study constrained participants by not allowing them to freely write trigger-action rules in natural language, which limited the investigation of language variability and the spontaneous use of linguistic structures. This restriction may have influenced participants' reasoning strategies and mental models, as they were guided by predefined linguistic options. Indeed, other linguistic elements, such as verb aspects or more nuanced syntactic structures, could also play significant roles in shaping working mental models of states and events during the composition of trigger-action rules. Future studies should explore a broader range of linguistic features or allow more freedom in expressing rules to explore language variability in trigger-action rule definition.

5.2 Study 2: Investigating Which Linguistic Clues Effectively Guide Non-Programmers

The second study aimed to investigate in depth the role of conjunctions and verbal structures in the interpretation and comprehension of trigger-action rules by non-programmer users, answering **RQ2**. The described study referred to the work reported in Andrao et al. (In submission). Specifically, through a two-task experiment, we investigated:

- Which subordinating conjunctions ("*if*", "*while*", "*as long as*" for states; "*if*", "*when*", "*as soon as*" for events) and verb forms (generic; state-specific or event-specific) can be considered as most appropriate in instructions to achieve automation goals involving either states or events (*Conjunction Selection Task*, described in subsection 5.2.3);
- Which, among those conjunctions, better supports users in identifying states and events accurately (*Scenario Classification Task* described in subsection 5.2.3).

We designed two categories of stimuli (i.e., generic or specific language elements) to support occurrences with different temporal features (states and events; Huang and Cakmak, 2015). The subordinating conjunction of each rule could be specific for the occurrence or generic (suitable for both states and events). In addition, rule pairs could

be presented either with a verb form specific per occurrence or with a generic one, resulting in a 2 (occurrence: state, event) $\times$ 2 (conjunction: generic, specific) $\times$ 2 (verb: generic, specific) experimental design. All variables were manipulated within participants.

Based on the literature (Desolda et al., 2021a; Huang and Cakmak, 2015; Zancanaro et al., 2022), we hypothesized that:

H1. Participants would prefer sentences containing conjunctions that elicit the difference between occurrences with different temporal features (e.g., for states "*while*" vs. "*if*"; for events "*when*" vs. "*if*"). In addition, we expected that different combinations of conjunction and verb type (generic or specific) would modulate users' preferences.

H2. Temporal conjunction (e.g., "*while*" and "*when*") would better support the identification of protracted vs. punctual occurrences compared to the conditional "*if*".

The study was conducted between March 2022 and May 2023 after approval by the Ethical Committee of the *University of Trento*.

5.2.1 Participants

Eighty-five (85) non-programmer participants aged between 20 and 34 ($M = 25.16$, $SD = 3.55$ years) attended our online study. Recruitment was carried out through social networks and snowball sampling. The sample included 46 females ($M_f = 25.07$, $SD_f = 3.17$ years) and 39 males ($M_m = 25.28$, $SD_m = 3.99$ years). All of them were native Italian speakers with no formal training in programming.

Thirty-nine (39) participants had a High school diploma, 21 had a Bachelor's degree, 23 had a Master's degree, and two had a Ph.D. or specialized diploma. The participants received no monetary compensation for participating.

5.2.2 Materials

We employed the 24 scenarios, the five conjunctions, and the three verb forms from the first study described in subsection 5.1.2.

Half (12) scenarios were designed to elicit the creation of event-based rules correctly triggered by states and the other 12 by events (two-level independent variable labeled "*occurrence*"). By manipulating the trigger part, we created six versions of a trigger-action rule for each scenario. While the action part remained invariant, the trigger part

differed by the type of subordinating conjunction (two-level independent variable labeled "*conjunction*") and associated verb form (two-level independent variable labeled "*verb*"). The "*conjunction*" could be either generic for every occurrence ("*if*") or specific ("*while*", "*as long as*" for states; "*when*", "*as soon as*" for events). Similarly, the "*verb*" could be either generic (e.g., "*it rains*") or specific (e.g., "*it is raining*" for states and "*it starts to rain*" for events).

As a result, we obtained 144 **trigger-action rules** (12 scenarios x 2 occurrences x 3 conjunctions x 2 verbs). All rules were grammatically correct and successfully supported the realization of the scenario. All scenarios and stimuli were designed and employed in Italian (see Table 5.7 for two examples of scenarios and Appendix C for the complete set of stimuli).

5.2.3 Task and Online Procedure

The study was conducted remotely using *Zoom*, the measures response time was collected using *PsychoPy* (hosted by *Pavlovía*; Peirce et al., 2019), and the questionnaire administered using *Qualtrics*. All information was collected anonymously, and each participant was assigned a unique identification code.

At the initial contact, participants were asked to read the information sheet and provide informed consent via *Qualtrics*. Subsequently, an experimental session was scheduled. Participants were instructed to perform the tasks using a computer in a quiet, distraction-free environment. On the day of the session, a researcher explained the procedure and the two tasks. After directing the participant to the *Conjunction Selection Task* on *Pavlovía*, the researcher remained in the virtual *Zoom* room to assist if needed.

Conjunction Selection Task

In the task, participants evaluated pairs of trigger-action rules that differed only in the conjunction that introduces the trigger part of the rule (e.g., "*while* <event>, <action>" vs. "*if* <event>, <action>").

Each participant saw 72 trials in the *Conjunction Selection Task*, divided into three 24-trial blocks. Each trial involved a scenario description followed by the presentation of two trigger-action rules (see Figure 5.2 – left panel). The task consisted of choosing which of the two trigger-action rules presented was more accurate as an instruction to realize the proposed scenario (adapted from De Vega et al., 2007). Half of the trials (36) displayed a state-scenario, and the other half (36) an event-scenario. For each of these

Table 5.7: In the table is visible an example of a state-scenario and an event-scenario paired for the verb used in the trigger part ("to rain") and the related trigger-action rules (original Italian version and English translation).

occurrence	verb	specific conjunction	generic conjunction
State: avoiding clothes getting wet	generic	<i>mentre piove, tieni l'ombrello aperto</i> (while it rains, keep the umbrella open) <i>finché piove, tieni l'ombrello aperto</i> (as long as it rains, keep the umbrella open)	<i>se piove, tieni l'ombrello aperto</i> (if it rains, keep the umbrella open)
	specific	<i>mentre sta piovendo, tieni l'ombrello aperto</i> (while it is raining, keep the umbrella open) <i>finché sta piovendo, tieni l'ombrello aperto</i> (as long as it is raining, keep the umbrella open)	<i>se sta piovendo, tieni l'ombrello aperto</i> (if it is raining, keep the umbrella open)
Event: avoiding water entering the house	generic	<i>quando piove, chiudi subito la finestra</i> (when it rains, close the window immediately) <i>appena piove, chiudi subito la finestra</i> (as soon as it rains, close the window immediately)	<i>se piove, chiudi subito la finestra</i> (if it rains, close the window immediately)
	specific	<i>quando inizia a piovere, chiudi subito la finestra</i> (when it starts to rain, close the window immediately) <i>appena inizia a piovere, chiudi subito la finestra</i> (as soon as it starts to rain, close the window immediately)	<i>se inizia a piovere, chiudi subito la finestra</i> (if it starts to rain, close the window immediately)

groups, 18 trials presented a generic verb form, and 18 a specific one.

In each trial, the two rules presented together (one at the top and one at the bottom of the screen) differed only for the subordinating conjunction. Participants saw each of the 24 scenarios three times (once for each block) in order to cover all possible comparisons across conjunctions ("if-while", "if-as long as", "while-as long as" for states; "if-when", "if-as soon as", "when-as soon as" for events). Each participant saw every (six) conjunction comparison six times with a specific verb form and six times with a generic one. In total, participants saw 24 times trials displaying the comparison between generic-specific conjunctions with specific verb form (12 times for states and 12 for events), 24 times with a generic one (12 times for states and 12 for events), and 24 times trials displaying the comparison between two specific conjunctions (control comparison, 12 for states and 12 for events). We counterbalanced sentences disposition (top-bottom) and blocks-order across participants. The stimuli were presented in pseudo-randomized order and were well-balanced among blocks. The participant could express their preferences for each trial by pressing the correspondent key (up or down arrows) on their computer keyboard. Once completed the *Conjunction Selection Task*, the last screen redirected the participant to the meeting with the researcher.

Scenario Classification Task

In the second part of the study, participants were engaged in the *Scenario Classification Task* (in *Qualtrics*) composed of 36 trials presented in randomized order. Each trial consisted of a scenario description and a single trigger-action rule with a generic verb form (i.e., the present tense "it rains"). The trigger part of the rule was colored green to highlight it (see Figure 5.2 – right panel). All sentences presented were the same as those seen by the participant during the *Conjunction Selection Task*. The *Scenario Classification Task* consisted of choosing in multiple response formats whether the part in green (the trigger) represented a prolonged (i.e., a state) or punctual occurrence (i.e., an event).

Finally, the participant filled out a short demographic form (required information: age, gender, previous experience with programming languages). The whole experiment duration did not exceed 30 minutes.

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE NON-PROGRAMMERS

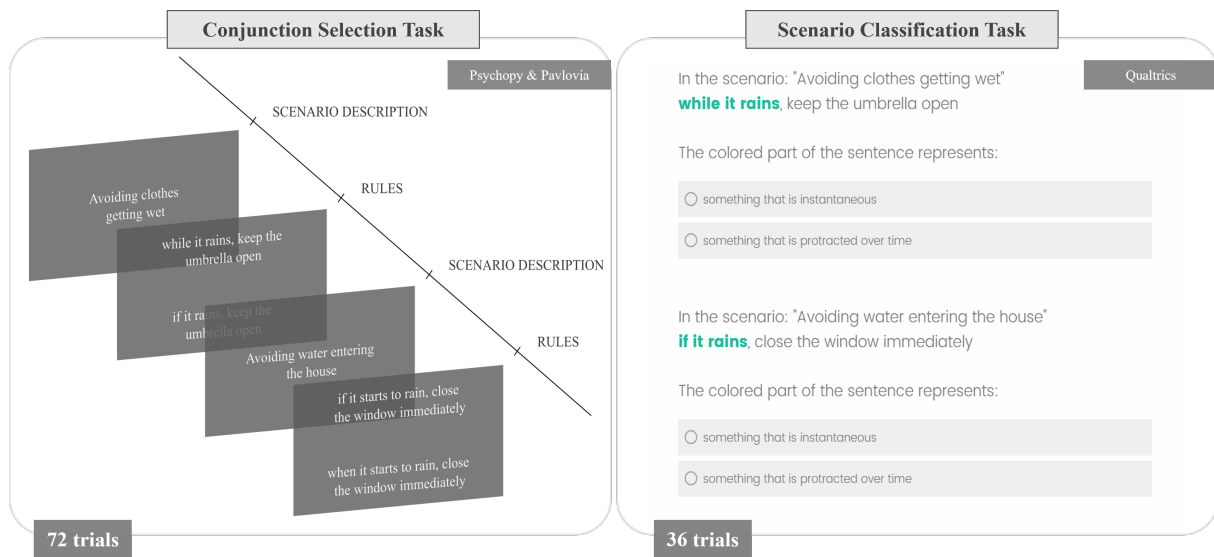


Figure 5.2: The image represents the two tasks of the experimental procedure. The left panel represents two trials of the *Conjunction Selection Task*: one concerning a state-scenario comparing "while" and "if" with a generic verb and one an event-scenario comparing "when" and "if" with a specific verb. The right panel represents two trials of the *Scenario Classification Task*: one concerning a state-scenario questioning the temporality of "while" and one an event-scenario questioning the "if".

5.2.4 Results

Conjunction Selection Task

Due to technical issues, data from one participant was lost. Additionally, data from two participants were excluded from the analyses as 10% of their response times (RTs) were less than 400 ms, indicating an elevated probability of random responses. Therefore, the analysis was conducted on a total of 82 participants.

Overall, participants showed a preference for the specific conjunctions ("as soon as", "when", "as long as", or "while") over "if": when asked to choose between the specific and generic conjunctions, participants chose the specific one in 59.43% of trials.

In order to assess whether the preference for the specific conjunction was of similar magnitude for events and states, as well as for generic and specific verbs, the percentage of choices for the specific conjunction was calculated for each participant, each type of occurrence, and each verb form. These percentages were submitted to an Analysis of Variance (ANOVA) with two factors within-participant: type of occurrence (state, event) and verb (specific, generic).

The ANOVA (see Table 5.8 and Table 5.9) showed that the percentages of choices for

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE NON-PROGRAMMERS

Table 5.8: The table reports the ANOVA output showing the effect of *occurrence*, *verb*, and their *interaction* on the choice for the specific conjunction.

effect	<i>DFn</i>	<i>DFd</i>	<i>F</i>	<i>p</i>	<i>ges</i>
occurrence	1	81	42.13	< 0.001	0.08
verb	1	81	11.43	0.001	0.02
occurrence:verb	1	81	31.11	< 0.001	0.04

Table 5.9: Post hoc comparisons using the *Bonferroni* correction. The table presents mean differences (M_{diff}) between conditions, standard errors (*SE*), and adjusted *p*-values.

contrast		M_{diff}	<i>SE</i>	<i>t</i>	p_{bonf}
event	state	11.84	2.19	5.41	< .001
verb generic	verb specific	-5.34	2.19	-2.44	0.02
event verb generic	event verb specific	-13.42	3.10	-4.33	< .001
state verb generic	state specific	2.74	3.10	0.89	1.00
event verb generic	state verb generic	3.76	3.10	1.21	1.00
event verb generic	state verb specific	6.50	3.10	2.10	0.22
event verb specific	state verb generic	17.18	3.10	5.55	< .001
event verb specific	state verb specific	19.92	3.10	6.43	< .001

the specific conjunctions were higher for events than for states (65.35% vs. 53.50%), as indicated by the main effect of the type of occurrence [$F(1, 81) = 42.13, p < 0.001$]. The preference for the specific conjunction was stronger when the verb form was specific than generic (62.10% vs. 56.76%), as indicated by the main effect of the verb form [$F(1, 81) = 11.43, p < 0.001$]. The interaction between the verb form and the type of occurrence was also significant [$F(1, 81) = 31.11, p < 0.001$].

As shown in Figure 5.3, in the case of events, the specific conjunction preference was higher for specific than generic verbs (72.05% vs. 58.64%; $p_{bonf} < 0.001$, *Bonferroni* post-hoc test; see Table 5.9). Indeed, a look at the number of choices for the different conjunctions made in the case of scenarios describing events revealed that “*when*” was chosen more often than “*if*” whether the verb associated was specific or generic [both $\chi^2_s(1, 82) > 35.42, p_s < 0.001$]. However, the preference for “*as soon as*” over “*if*” was significant only in the case of scenarios including a specific verb [$\chi^2(1, 82) = 70.32, p < 0.001$] and not a generic verb [$\chi^2(1, 82) = 2.94, p = 0.09$], see Table 5.10 and Figure 5.4A.

As evident in Figure 5.3, in the case of states, the preference for specific conjunctions did not differ between generic and specific verbs (54.88% vs. 52.13%; $p_{bonf} = 1.00$

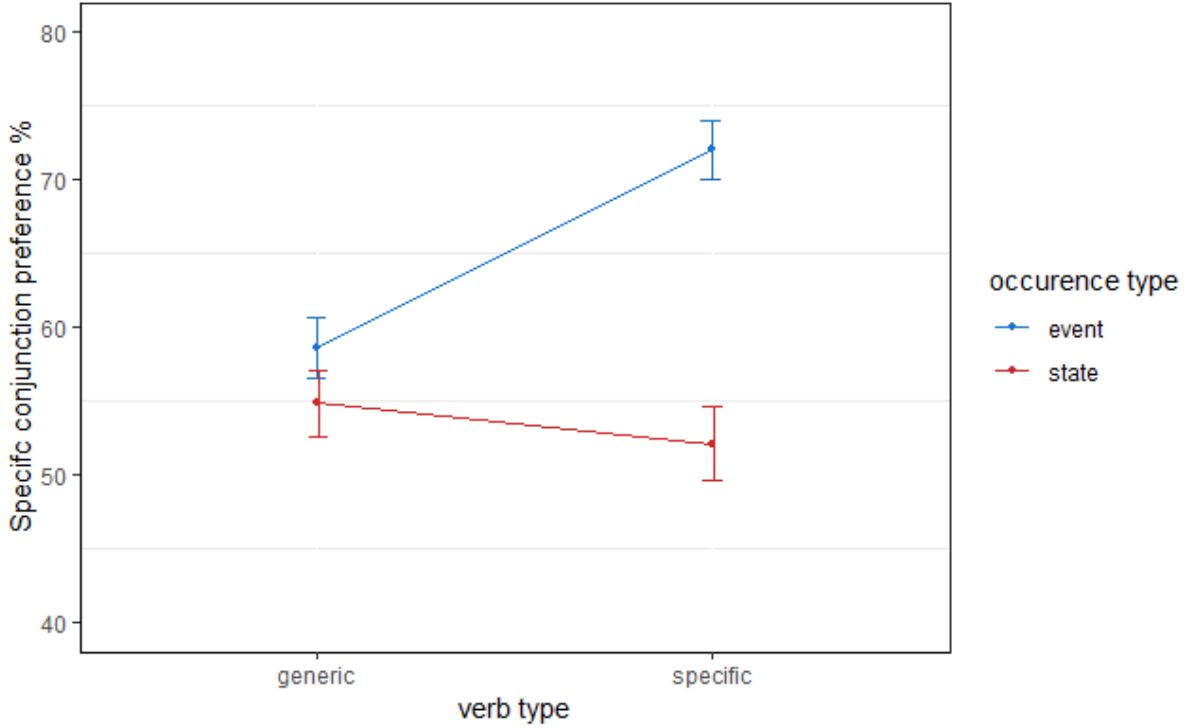


Figure 5.3: Comparison of specific conjunction choices for the type of occurrence (state, event) and type of verb (generic, specific).

Bonferroni post-hoc test). Indeed, for states, the same pattern of findings was observed regardless of the verb form: “*while*” was chosen more often than “*if*”, whether the verb was specific or generic [both $\chi^2_s(1, 82) > 14.34, p_s < 0.001$], whereas the number of “*as long as*” choices did not significantly differ from the “*if*” choices, either when the verb was specific [$\chi^2(1, 82) = 3.58, p = 0.06$] or generic [$\chi^2(1, 82) = 0.07, p = 0.79$], see Table 5.10 and Figure 5.4A.

The scenarios requiring a choice between two specific conjunctions were also analyzed. It was found that “*when*” was chosen more often than “*as soon as*” in trials with a generic verb [$X^2(1, 82) = 14.34, p < 0.001$], whereas it was observed an opposite pattern (i.e., a preference for “*as soon as*”) when the verb was specific [$X^2(1, 82) = 17.96, p < 0.001$]. “*While*” was chosen more than “*as long as*” in both generic and specific verb conditions [$X^2(1, 82) = 29.27, p < 0.001$, and $X^2(1, 82) = 22.84, p < 0.001$, respectively], see Table 5.11 and Figure 5.4B.

In order to control for possible speed-preference trade-offs (e.g., people tend to

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE NON-PROGRAMMERS

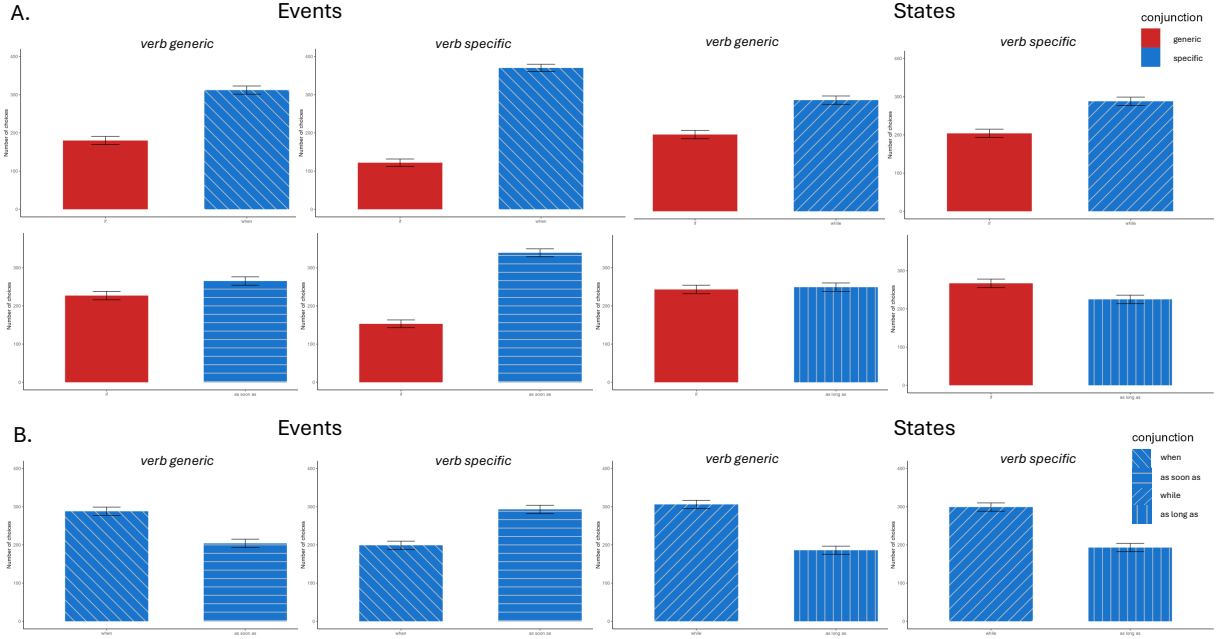


Figure 5.4: Number of choices (absolute frequency) for each conjunction for generic (left side) and specific verb form (right side). Panel (A) shows the comparison of generic-specific conjunctions, while panel (B) shows the control comparison (specific-specific conjunctions).

choose a given conjunction but are actually faster in choosing the other conjunction or do not show any preference in terms of number choices but are faster in making certain choices), participants' RTs were also analyzed. Clearly, mean RTs of specific vs. generic conjunction choices can only be compared in the case of participants who did not exclusively choose one of the two types of conjunctions. We decided to consider only participants who made at least 3 choices for each type of conjunction (i.e., participants who did not actually show a strong preference for a conjunction over the other, thus making a sufficient number of responses for each type of conjunctions). We only observed significantly faster selections of specific than generic conjunctions (6.92 s vs 7.92 s) in the case of event-scenarios with specific verbs [$t(51) = 2.23, p < 0.05$]. All the other comparisons were not significant (all $t_s < 0.42$, all $p_s > 0.68$; see Table 5.12 and Appendix D).

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE
NON-PROGRAMMERS

Table 5.10: Frequencies for conjunction selection and chi-square (χ^2) values for each comparison in experimental trials (generic vs. specific verb).

trial	verb	resp	frequency	percent	χ^2	df	p
<i>if-when</i>	generic	when	312	63.42	35.42	1	< .001
		if	180	36.60			
	specific	when	370	75.20	125.01	1	< .001
		if	122	24.80			
<i>if-as-soon as</i>	generic	as soon as	265	53.86	2.94	1	0.09
		if	227	46.14			
	specific	as soon as	339	68.90	70.32	1	< .001
		if	153	31.10			
<i>if-while</i>	generic	while	291	59.15	16.46	1	< .001
		if	201	40.85			
	specific	while	288	58.54	14.34	1	< .001
		if	204	41.46			
<i>if-as long as</i>	generic	as long as	249	50.61	0.07	1	0.79
		if	243	49.39			
	specific	as long as	225	45.73	3.58	1	0.06
		if	267	54.27			

Scenario Classification Task

In addition to the participants excluded from the first task, data from another participant was lost. The following analysis is therefore based on 81 participants.

In order to understand if some conjunctions better support participants in correctly classifying the type of occurrence (state or event), the accuracy percentage of responses was calculated for each participant, each type of occurrence, and each conjunction.

The percentages for scenarios involving states and those for scenarios involving events were separately subjected to two ANOVAs (one for states and one for events) with one within-participant factor: the type of conjunction ("*if*", "*while*", "*as long as*" for states, and "*if*", "*when*", "*as soon as*" for events). Overall, participants answered correctly in 66.87% of the event-scenarios (see Table 5.13). The ANOVA showed a significant difference in the accuracy percentages for classifying events depending on the type of conjunction [$F(2, 80) = 103.67, p < 0.001$]. Specifically, participants showed higher accuracy with the conjunction "*as soon as*" (93.80%) compared to the other two (all $p_{\text{bonf}_s} < 0.001$ Bonferroni post-hoc test, see Table 5.14) and tended to be more accurate with "*if*" (64.00%) than with "*when*" (42.8%; $p_{\text{bonf}} < 0.001$ Bonferroni post-hoc test).

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE
NON-PROGRAMMERS

Table 5.11: Frequencies for conjunction selection and chi-square (χ^2) values for each comparison in control trials (generic vs. specific verb).

trial	verb	resp	frequency	percent	χ^2	df	p
<i>when-as-soon as</i>	generic	when	288	58.54	14.34	1	< .001
		as soon as	204	41.46			
	specific	when	199	49.45	17.96	1	< .001
		as soon as	293	59.55			
<i>while-as long as</i>	generic	while	306	62.20	29.27	1	< .001
		as long as	186	37.81			
	specific	while	299	60.77	22.84	1	< .001
		as long as	193	39.23			

Table 5.12: The table presents mean RTs for choosing specific vs. generic conjunction (in trials with less than three choices), where n represents the participants' number out of the total sample ($N = 82$). The t -test are reported (Fisher's F -test confirmed no significant difference in variance between the two samples).

occurrence	verb	n	conj	$M(s)$	$SD(s)$	t	df	p
event	generic	66	generic	6.69	5.94	0.06	65	0.95
			specific	6.65	5.57			
	specific	52	generic	7.93	5.85	2.23	51	0.03
			specific	6.92	5.52			
state	generic	71	generic	6.14	5.59	0.42	70	0.68
			specific	6.47	5.70			
	specific	62	generic	6.59	5.25	0.37	61	0.71
			specific	7.38	6.11			

For state-scenarios, participants answered correctly 79.2% of the time (see Table 5.13). The ANOVA revealed a significant difference in accuracy percentages depending on the type of conjunction [$F(2, 80) = 91.55, p < 0.001$]. Specifically, participants were more accurate with either "while" (95.06%) or "as long as" (92.80%) compared to "if" (50.00%; $p_{\text{bonf}} < 0.001$ Bonferroni post-hoc test).

No significant differences were found between "while" and "as long as" ($p_{\text{bonf}} = 1.00$ Bonferroni post-hoc test; Table 5.14). See Figure 5.5.

5.2. STUDY 2: INVESTIGATING WHICH LINGUISTIC CLUES EFFECTIVELY GUIDE NON-PROGRAMMERS

Table 5.13: The table presents the percentage of accurate answers in event and state trials, divided by conjunction, and the output of two one-way ANOVAs showing the effect of conjunction on the accuracy of identifying the occurrence.

occurrence	conj	mean	sd	df	F	p	η^2
event	if	63.99	26.15	2	103.67	< .001	0.46
	when	42.80	24.99				
	as soon as	93.83	15.24				
state	if	50.00	32.49	2	91.55	< .001	0.43
	while	95.06	13.81				
	as long as	92.80	21.56				

Table 5.14: Post hoc comparisons using the *Bonferroni* correction. The table presents mean differences (M_{diff}) between conditions, standard errors (SE), and adjusted p -values.

occurrence	contrast		M_{diff}	E	t	p_{bonf}
event	as soon as	when	51.03	3.56	14.33	< .001
	as soon as	if	29.84	3.56	8.38	< .001
	when	if	-21.19	3.56	-5.95	< .001
state	as long as	while	-2.26	3.75	-0.60	1.00
	as long as	if	42.80	3.75	11.41	< .001
	while	if	45.06	3.75	12.01	< .001

5.2.5 Discussion

Our results suggest that specific conjunctions were preferred by non-programmers for describing occurrences that prompt trigger-action rules compared to the generic conjunction "if" and some of them, more than others, led to a more accurate distinction between expressions denoting states and those denoting events. While providing more comprehensive evidence, these results are in line with other recent studies that observed that highlighting states and events temporal differences and distinguishing them by using specific temporal conjunctions for each type of occurrence seems to effectively help users in creating (Desolda et al., 2021a; Manca et al., 2019) and understanding trigger-action rules (Zancanaro et al., 2022). Furthermore, as hinted Huang and Cakmak (2015) and implemented in other work (Ghiani et al., 2017; Manca et al., 2019), the verb form can facilitate event-state distinction. Indeed, we observed that the preference for specific conjunctions was stronger for describing events (vs. states), particularly when associated with the specific verb form (e.g., "it starts to rain"), em-

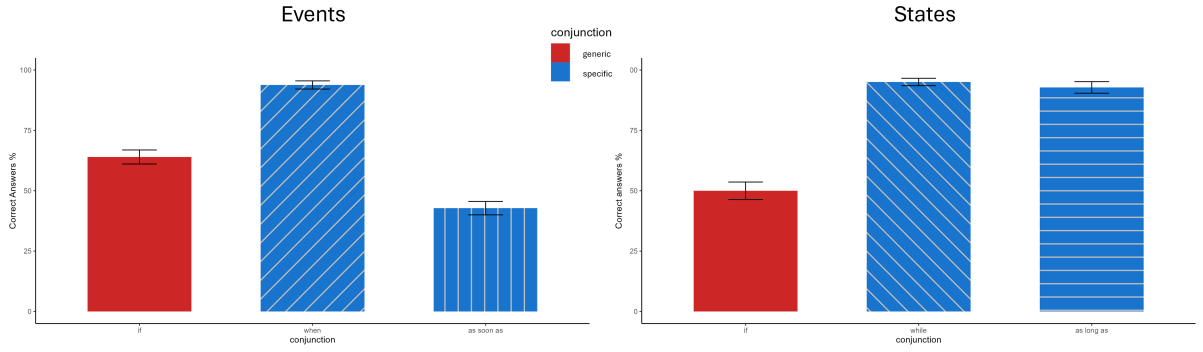


Figure 5.5: The bar plots represent the average percentage of times participants accurately identified the type of occurrences in the *Scenario Classification Task*.

phasizing, even more, the punctual nature of the occurrence.

Previous studies (Brackenbury et al., 2019; Desolda et al., 2021a; Manca et al., 2019; Zancanaro et al., 2022) focused on the use of *"when"* for introducing events or *"while"* for introducing states instead of the generic conjunction *"if"*. In our study, we also considered other specific conjunctions (*"as soon as"* for events and *"as long as"* for states) to explore additional alternatives that could support users. Our results showed that among specific conjunctions for events, *"when"* was preferred over *"as soon as"* with ambiguous verb forms (i.e., present tense), while *"as soon as"* was favored when the verb form clearly indicates the punctual nature of the event (e.g., *"it starts to rain"*).

However, despite *"when"* being preferred by non-programmers, it was the only one among specific conjunctions that did not help to correctly identify the occurrence type for events (below 50%, worse than *"if"*). This result is in line with the findings of other authors who noticed the conjunction *"when"* is naturally used to introduce both states and events (Andrao et al., 2023d; Broccias et al., 2022), and it is often used interchangeably with *"if"* (Dey et al., 2006; García-Herranz et al., 2010; Pane et al., 2001). In contrast, *"as soon as"* was the most helpful for guiding participants in correctly classifying the occurrence.

For states, we found that *"while"* is consistently preferred over *"as long as"*, but both equally helped in accurately identifying occurrences compared to *"if"* (which neither helps nor is favored).

Limitations

One of the main limitations of this study is that participants were not given the freedom to write their rules in natural language, constraining the investigation of the language

variants used. We decided to restrict and control the pool of linguistic aspects to fit an experimental design. However, studying other linguistic variants and enabling individuals to be more "natural" in the rule definition may enrich our knowledge and give us other guidelines/alternatives for implementing TAP-based EUD systems effectively. Many other existing language options were not considered in this study, even if these can probably support the distinction between states and events. For example, other linguistic cues highlighting the temporal distinction could be "*during*" for states and "*precisely when*" for events. Future studies should investigate those other linguistic cues, perhaps adopting an approach in which participants write in natural language. In addition, future studies should address more ecological interaction rather than experimentally controlled conditions. Despite those limitations, we are confident this study can provide a baseline for grounding future research on language temporal aspects in TAP.

5.3 Study 3: Exploring Natural Linguistic Patterns in Trigger-Action Rules

In the third online study, we investigated natural linguistic descriptions of rules for defining complex home automation scenarios (i.e., a combination of a state and an event that triggered the desired action) by non-programmer and expert-programmer participants (similarly to Galotti and Ganong, 1985), in order to answer **RQ3**. Specifically, we investigated:

- The existence of an **order** in which state-event-action is most frequently represented and described by participants;
- Whether the **level of expertise in programming** is related to differences in the linguistic patterns naturally adopted.

Through the analysis of natural language descriptions, we hypothesized:

- H1.** To find that the trigger would be placed first (trigger-first order) in most sentences, replicating the findings of (Pane et al., 2001), who observed that non-programmers most used the sentences begin with "*if*" or "*when*" structure in their statements. Furthermore, we hypothesize that participants would describe the rules according to the order found in (De Vega et al., 2007). Therefore, from the description analysis, we expected that the trigger part (state or event) would be before the ac-

tion part and the longer occurrence or “ground” (i.e., the state) before the “figure” (i.e., the event).

H2. To find differences between programmers and non-programmers, particularly in the expression of the details but not in the linguistic patterns used (due to the standard conditional structures in programming).

Before conducting the main study, two pilot studies were performed to refine the participant instructions (described in subsection 5.3.4) and evaluate the suitability of the stimuli and procedure (described in subsection 5.3.5). The study was approved by the Ethics Committee of the *University of Trento* and was conducted during 2023-2024.

5.3.1 Participants

For this study, we recruited 174 participants aged between 19 and 59 ($M = 24.34$, $SD = 7.29$ years). Sixty-two (62) were male ($M_m = 27.47$, $SD_m = 9.16$ years), 109 were female ($M_f = 22.57$, $SD_f = 5.30$ years), and three preferred not to answer ($M_{na} = 24.00$, $SD_{na} = 6.93$ years). All participants were native Italian speakers, mostly from northern Italy (78.95%), but some of them were from central (9.36%) and southern Italy (11.70%). See Figure 5.6 for the representation of participants distribution in Italy.

Regarding the education levels among participants, the majority ($n = 120$) held a High school diploma, 24 had a Bachelor’s degree, 19 had a Master’s degree, seven had a Ph.D., one had a junior high school diploma, and three did not respond.

Inclusion criteria required participants not having experience with programming languages (*non-expert group*) or being experienced programmers (*expert group*).

A total of 108 participants (age range: 19–59 years; $M = 22.72$, $SD = 6.31$) were part of the *non-expert group*, having no experience with programming languages. Among them, 83 were female, 22 were male, and three preferred not to disclose their gender. Regarding experience with home automation systems, 72 participants had no prior experience, 17 had rare exposure, 11 had used such systems occasionally, and one reported frequent use, while seven did not respond.

The remaining 66 participants (age range: 20–56 years; $M = 26.99$, $SD = 8.04$) formed the *expert group*, as they were expert programmers or had experience with programming languages. Among them, 26 were female and 40 were male. Regarding experience with home automation systems, 29 participants had no prior experience, 12 had rare exposure, 16 had used such systems occasionally, six reported frequent use, and three did not respond.

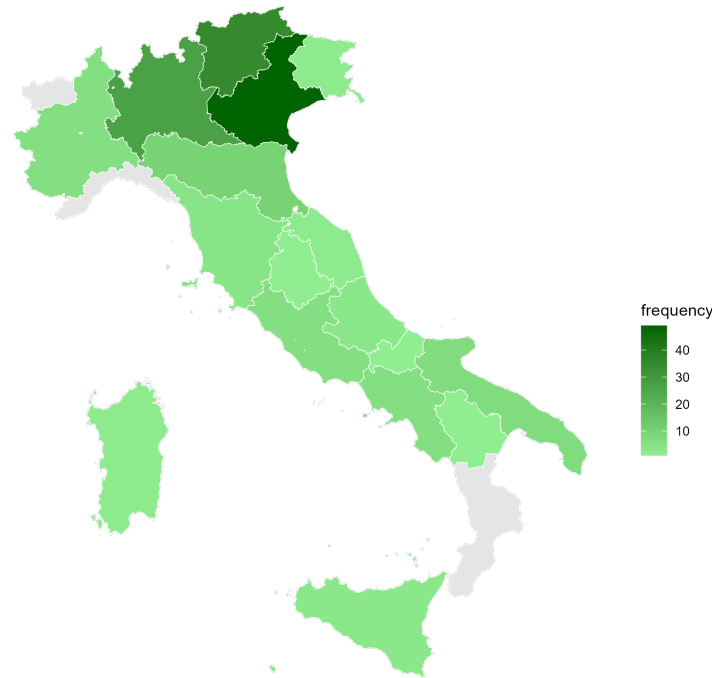


Figure 5.6: The figure represents, through a heatmap, the Italian regions of origin of the individuals who participated in our study ($N = 174$).

Participants in this group were further divided into two subgroups based on their experience with programming languages:

- *High expertise*: participants who use programming languages daily ($n = 41$). Within this subgroup, 23 had more than five years of experience, five had between three and five years, eight had between one and three years, two had less than one year, and three did not respond.
- *Previous experience*: participants who do not currently use programming languages ($n = 25$) but had experience in the past. Of these, five had used programming languages for more than three years, eight for one to three years, and 12 for less than one year.

5.3.2 Materials

Twelve (12) silent videos were created (to avoid language bias – as suggested by Pane et al., 2001) representing different home automation scenarios (see Figure 5.7). Each video lasted around 15 seconds ($M = 13.42$, $SD = 0.90$ s) and represented a combina-

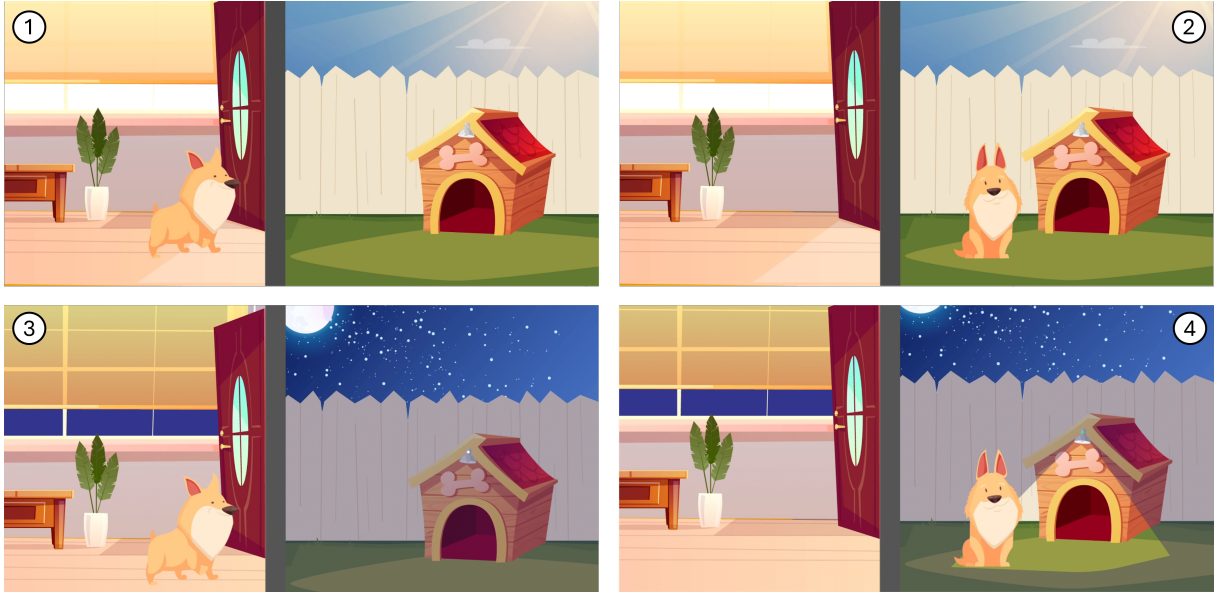


Figure 5.7: The figure presents four frames from one of the videos used in the study. The frame (1) represents the event (*"the dog goes out"*) with the false state (*"it is daytime"*), which results in no action in frame (2). The frame (3) depicts the same event when the state is true (*"it is night"*), leading to the execution of the action (*"turn on the dog kennel light"*).

tion of one state (e.g., *"it is raining"*), one event (e.g., *"it becomes night"*), and one action (e.g., *"close the window"*). The initial segment of each video portrayed the smart home's response when the state is false (e.g., *"It's day, the dog goes out, and no action occurs"*), followed by the representation of when the state is true. To create the video scenarios, we identified six occurrences' topics that can act as events or states:

- Occurrences involving **home devices**. For example, *"the TV is turned on"* and *"the TV is switched on"* represent a state and an event within this category.
- Occurrences related to **day-night** transitions, such as the state *"it is night"* or the event *"it becomes night"*.
- Occurrences associated with **weather** conditions. For example, *"it rains"* and *"it starts raining"* correspond to a state and an event related to weather, respectively.
- Occurrences involving **temperature**, such as the state *"it is 10 degrees"* or the event *"the temperature reaches 10 degrees"*.
- Occurrences related to **danger**, such as the state *"smoke is detected"* or the event *"smoke starts to be detected"*.

Table 5.15: The table reports the balanced distribution of state and event topics for each video and a brief description of the elements characterizing it. The duration (in seconds) is also presented for each video.

id	topic	state	event	action	(s)
s1	<i>home devices / danger</i>	elevator	smoke	close door	13
s2	<i>day-night / weather</i>	night	it rains	close window	13
s3	<i>day-night / presence</i>	night	dog exits	lights on	13
s4	<i>home devices / weather</i>	TV on	it rains	close window	13
s5	<i>temperature / presence</i>	it is cold	dog enters	fireplace on	16
s6	<i>presence / day-night</i>	dog outside	it becomes night	open door	14
s7	<i>temperature / day-night</i>	it is cold	it becomes night	fireplace on	13
s8	<i>presence / danger</i>	dog inside	smoke	open door	14
s9	<i>weather / temperature</i>	sun shines	temperature increases	water plants	13
s10	<i>weather / home devices</i>	sun shines	TV on	close curtains	13
s11	<i>danger / temperature</i>	smoke	temperature increases	sprinkler on	13
s12	<i>danger / home devices</i>	smoke	elevator	close door	13

- Occurrences linked to the **presence** of a living being. For example, "*the dog is in the living room*" and "*the dog enters the living room*" represent a state and an event in this category, respectively. Human representations were deliberately avoided to mitigate potential biases related to over- or under-identification with symbols associated with gender or physical attributes (e.g., skin, eyes, hair).

The 12 videos were designed to balance different state-event-action topic combinations as visible in Table 5.15. We also created a mirrored version of each video (displaying the opposite left-right spatial pattern) to account for any potential effects related to the spatial orientation of the scenes.

The initial version of the videos was used in a pilot study (see subsection 5.3.5), from which we discovered that some elements of the videos were confounding, then adjusted accordingly for the final study.

5.3.3 Procedure and Task

Informed consent and data were collected through an online form in *Qualtrics*. All participants watched the 12 videos presented one at a time. Participants were asked to carefully write instructions in Italian to handle each video as if they were describing it to an *"automated system"* that can understand Italian but had no previous experience with the elements of the scenario. The instructions were adopted from Galotti and Ganong (1985) and re-structured considering the results of our pilot study (see subsection 5.3.4).

After the screen with general instructions, in each trial, the participant was presented with a screen with a video, a "play" button, and a text box where one could write the instructions in natural language. We modified the play button so that you could never pause the video but could play it multiple times.

Each participant viewed half of the videos in their mirrored versions. The videos were presented in a pseudo-random order and divided into two blocks to prevent participant fatigue. The order of the blocks was counterbalanced across participants.

After the task, participants were asked to complete a short demographic form, including questions about previous experience with programming languages and home automation systems. The estimated duration of the study was approximately 20 minutes.

5.3.4 Pilot Study A: Appropriateness of Instructions

The first pilot study aimed to understand which instructions accurately and exhaustively elicit spontaneous trigger-action rule writing. We adapted the study design of Galotti and Ganong (1985) to define which type of target (a *"person"* vs. an *"automated system"*) elicits more detailed instructions.

Participants

We recruited eight participants ($N = 8$) aged between 24 and 50 ($M = 28.63$, $SD = 8.75$ years) through snowball sampling. Four were non-programmers (two females and two males) and four expert-programmers (two females and two males).

Materials

We designed and used two videos (similar to those described in subsection 5.3.2). These were part of the first version of the videos that was slightly different in structure from the final version (e.g., there was no "false state" scene).

Procedure and Task

The pilot study was conducted online as an individual session using *Qualtrics* and *Zoom*. Participants took part in two trials, each requiring them to perform the same task. In each trial, participants read the task's instructions, watched a video, and described the rules for handling the scenario presented. Each trial offered a specific version of the task instructions, differing only in the target, which could be a "person" or a "automated system"). The task-instruction was as follows:

*"For each video, your task is to describe to **TARGET** the instructions that operate the smart home in the scenario presented. **TARGET** can read the sentences in Italian, but has no other prior knowledge and experience about the scenario you are going to describe".*

In addition, after each trial, participants were asked their opinion (in the form of an open-ended question) about the instructions they received and whether they needed more details to understand the task. The order of the videos and instructions was counterbalanced among participants. We used the technique of *thinking aloud* to elicit participants' mental models (Boren and Ramey, 2000), and the sessions were video-audio recorded.

Results

Interview recordings were transcribed and analyzed by two independent researchers. The first pilot study's results suggested instruction changes, which we modified accordingly:

- Instructions with the "automated systems" target were adopted as this variant prompts programmers and non-programmer participants to write with more details their answers.
- The instructions were *detailed* enough. Participants reported that receiving longer instructions with more details would not affect their understanding and responses; in fact, it would even be misleading to have longer instructions.

- We highlighted *keywords* in the task's instructions: some participants suggested highlighting in bold some words ("*instructions*", "*smart home*", "*sentences in Italian*") to grasp participants' attention.

In the end, the task instructions presented were:

*"For each video, your task is to describe to an **automated system the instructions that operate the smart home** in the scenario presented. The automated system can read the **sentences in Italian**, but has no other prior knowledge and experience about the scenario you are going to describe".*

We also learned some limits on the procedure, that we tried to avoid in the final version of the study:

- Some participants could not remember the task instructions during the study-session. We added a *summary of the instructions* above the video in each trial:

"Use the space below the video to write the instructions that operate the smart home in the scenario presented to an automated system that can read sentences in Italian".

- In the study information sheet, we specified and highlighted that the videos represented *hypothetical scenarios*. In fact, some of the pilot study participants focused on safety aspects if these scenarios were implemented in real life.
- Some participants paused the video several times during the session. Additionally, the default audio control led some participants to believe there was sound in the video. To prevent any confusion, we removed the standard video *controls* and customized the "*play*" button. This way, participants could not pause the video, but they could replay it multiple times after it had finished, ensuring they effectively engaged with the whole content.

5.3.5 Pilot Study B: Refinement of Materials and Procedure

In the second pilot study, we tested and qualitatively explored the whole procedure with the adapted instructions (resulting from the pilot study described in subsection 5.3.4) and the 12 videos (described in subsection 5.3.2).

Participants

Sixteen (16) participants, eight females and eight males, aged between 24 and 60 years ($M = 32.81$, $SD = 11.44$ years), were involved in the second pilot study. Eight (four males and four females) had no previous experience with programming language. Two had no experience with home automation tools, while six had minimal experience with smart home environments. The other eight (four males and four females) were experienced programmers. Five of them had experience with home automation systems. All participants were native Italian speakers. See details about participants in Table 5.16.

Table 5.16: The table shows the demographic details of our 16 participants (age, gender, education, experience with home automation systems).

id	age	g	group	degree	degree field	smart home
P1	26	F	expert	Master	Engineering	Never
P2	28	M	non-expert	Master	Medicine	Almost never
P3	24	M	non-expert	Diploma	NA	Almost never
P4	29	F	non-expert	Master	Psychology	Almost never
P5	26	F	expert	Bachelor	Engineering	Sometimes
P6	29	M	expert	PhD	Computer Science	Always
P7	59	F	non-expert	Master	Civil Engineering	Never
P8	28	F	non-expert	Master	Psychology	Almost never
P9	37	F	expert	PhD	Architecture	Almost never
P10	60	M	expert	Diploma	NA	Almost never
P11	26	M	expert	Bachelor	Mathematics	Almost never
P12	25	F	non-exp	Master	Architecture	Never
P13	27	M	non-expert	Master	Psychology	Almost never
P14	29	M	non-expert	Diploma	NA	Almost never
P15	43	M	expert	Bachelor	Computer Science	Never
P16	29	F	expert	Master	Languages	Never

Materials

We used the first version of the 12 videos described in subsection 5.3.2. In this version of the videos, the devices involved in the action displayed a green light when the state was true, a red light when the state was false, and a blue light when the event was not detected.

Procedure and Task

The core procedure and tasks were consistent with those outlined in subsection 5.3.3. After providing informed consent, participants viewed 12 video scenarios presented one at a time. For each video, participants were asked to write the instructions needed to implement the automation. Differently from the final procedure, this phase employed the *thinking aloud* technique to elicit participants' mental models (Boren and Ramey, 2000) and deeply explore their interpretations of the videos. The sessions were video and audio recorded, and the transcriptions were analyzed by two researchers.

Results

We collected 199 instructions (some participants wrote multiple independent/alternative rules to define the same scenario) from the pilot study. From a preliminary analysis of the verbal reports, it was found that the device lights represented in the videos were often misinterpreted by both programmers and non-programmers.

Indeed, participants often interpret the light color change as part of the action, including it in the rule, as participants P7, P8 (from the non-expert group) and P11, P15 (from the expert group) did:

- P7 - non-expert group [scenario s1]: *"In case smoke comes out of the oven, the elevator doors, if they are open, should close, **the light from green (open door) should turn red.** On the other hand, if the doors are closed, they must remain closed. **The light from blue must turn red**".*
- P8 - non-expert group [scenario s6]: *"When the door is closed, the **blue light turns on.** At 8 p.m., when there is no dog, the **blue light turns off, the red light turns on,** and the door stays closed. At 8 p.m., when the dog is there, the **blue light turns off, the green light turns on,** and the door opens".*
- P11 - expert group [scenario s4]: *"If there is TV on and it starts to rain close the open windows and **change the color of the LED light to green,** otherwise the **color of the light turns red**".*
- P15 - expert group [scenario s2]: *"When it is not raining the **warning light should be blue.** During the day when it rains the **warning light should be red.** During the night when it rains, the **warning light should be green** and any open windows should be closed, unless there are people or objects in the way".*

Some participants clearly expressed their difficulties in interpreting the color change of light, for example:

- P7 - non-expert group: *"I didn't understand this thing about the **little light**. It turns from light blue to green [...] I don't know why the green light turns on. Should I write that too [in the rule]? I don't know!"*.
- P8 - non-expert group: *"By the way, I don't even know if I have to consider that the blue light turns on. No, the blue light maybe just indicates that it is in function"*.
- P12 - non-expert group: *"Why is it then it turns red, and then it turns green? I don't know, maybe it's the version with and without domotics"*.

We think it was not clear the meaning of the colors, considering the participants' comments reported above and the question asked by P9:

- P9 - expert group: *"I see the red sensor [light], then blue, then I see the green sensor. Why three colors?"*

In addition, the red-green color is symbolically associated with a danger-safe way, confounding P1 in interpreting a scenario dealing with danger as a topic:

- P1 - expert group [scenario s1]: *"For example, here nothing actually happens, but it gives me the red light, which is usually a danger signal. While here with the smoke, it shows the green signal"*.

We decided to remove the device lights from the videos because they were more ambiguous than helpful, which may have led to confusion. Otherwise, the instructions were understood by all participants, so no other changes were made to the procedure and materials.

5.3.6 Data analysis

Instructions were analyzed by a researcher who compared the instruction created by participants with the "prototype rule". The prototype rule is an instruction that presents all the elements of the video (i.e., the ones presented in Table 5.15). For instance, a prototype rule could be *"night [state], rain [event], close window [action]"*, while an instruction could be *"if it rains at night, close the window"*. The researcher annotated each instruction by identifying:

- The **position** of the state, event, and action: labeled with numbers ranging from 1 to 3. The researcher annotated "na" if the element (state, event, or action) was not present in the instruction or if it was not related to the video. For example, considering the prototype rule "*night* [state], *rain* [event], *close window* [action]", the instruction "*switch off the light when it rains if I'm not at home*" identified only the event, thus the annotator would just consider the event in the analysis.
- The **order** of elements: labeled with a string containing "A" for action, "S" for the state, "E" for events (e.g., ESA, AE, SEA).
- The **type of order**: trigger-first (when the first element of the sentence was the state or the event) or action-first (when the first element was the action).
- The **conjunctions** introducing state and event: labeled with the actual conjunction in the case of "*if*", "*when*", "*while*", "*as soon as*", "*as long as*", or with "*others*" when there were propositions or constructs meant to introduce the element temporally or conditionally (e.g., "*in case*", "*during*"). When the state/event elements were connected by a logical preposition (e.g., "*and*"; "*or*"), the conjunction used for the first element is assumed in the second, and was annotated in parentheses. It was used "na" if the corresponding element (state or event) was not present there was no introductory conjunction/clause or the clause/conjunction did not invoke either element of conditionality or temporality (e.g., "*for*").
- The **type of instruction**: (i) "*rule*" when the instruction was presented in the form of a rule or it's likely similar to it (e.g., "*it's daytime, close the window*"); (ii) "*command*" when the instruction describe just an action (e.g., "*open the window*"); (iii) "*description*" when the participants described the scene (e.g., the ones starting with "*in the video there is*"); (iv) "*rule/description*" when elements of descriptions are combined with rules (e.g., "*the video shows a home automation system that activates the heating when it is cold and dark*"); (v) "*other*", for example, when the participants expressed personal comments (e.g., "*I would not let the window open during the day*").
- The **type of rule**: that could be either "*simple*" when it contained just one element in the trigger or "*complex*" when it contains both state and events.

See Table 5.17, for some examples of annotated instructions. For compound instructions – where there were more clauses describing the scenario – the researcher analyzed the syntactic and linguistic aspects of one clause, following two rules:

1. Prioritize the most **exhaustive** clauses. If there is one clause that contains state, event, and action and the other describes only the state give priority to the most complete one. For example, in the instruction *"During the day keep the living room light off. During the night if someone is moving, switch on it"*.
2. In case of equal exhaustiveness of the clauses, the **first** one that was written was considered.

Table 5.17: The table shows some examples of annotated instructions. We have chosen some instructions related to scenario s2, whose rule prototype is *"night [state], rain [event], close window [action]"*. In the table, "S" refers to the state, "E" to the event, "A" to the action, "conj." to conjunction, "descr." to description, "comm.d" to command.

instruction	S	E	A	order	conj.S	conj.E	type
"if it rains at night, close the window"	2	1	3	ESA	other	if	rule
"close window when it rains"	na	2	1	AE	na	when	rule
"if the sun has set and it is raining, then close the window"	1	2	3	SEA	if	(if)	rule
"close the windows"	na	na	1	A	na	na	comm.d
"the system during the night has the ability to control the opening/closing of windows"	1	na	2	SA	other	na	rule/ descr.
"it starts to rain"	na	1	na	E	na	na	descr.
"habits"	na	na	na	na	na	na	other

5.3.7 Results

From the main study, a total of 2109 instructions were collected. While some participants did not respond to all videos, others provided multiple instructions for a single video, with the number of instructions per participant ranging from 1 to 30 ($M = 12.12$, $SD = 2.64$). Given the research objective of exploring linguistic aspects, the 199 instructions from the second pilot study were also included in the analysis, as the procedure and the videos remained the same, with the only difference being the devices' lights mentioned in subsection 5.3.5. Consequently, a total of 2308 participant-generated instructions were analyzed.

Among these, 1896 were classified as rules, 171 as rules/descriptions, 126 as descriptions, 92 as commands, and 23 as others. For the analysis, only the 2067 instructions classified as *rules* or *rules/descriptions* (hereinafter referred to as “rules”) were considered. Seventeen (17) rules were also excluded despite their classification, as they lacked sufficient elements of the prototype rule. Specifically, they were excluded because:

- The action was missing, as in the case of the rule “*it’s daytime turn on the TV*” for describing s10 (rule prototype: “*sun shines [state], TV on [event], close curtains [action]*”). In this case, the only common element between rule-generated and rule prototype is the state.
- Only the action corresponded to the prototype rule. For example, the rule “*turn on the sprinkler system at 11 a.m.*” for handling the scenario s9 has only the action that fits the rule prototype “*sun shines [state], temperature increases [event], garden sprinkler [action]*”.

In conclusion, a total of 2050 rules were included in the final analysis of which 835 were classified as simple and 1215 as complex trigger-action rules.

Order of Elements in Rules

Participants predominantly wrote rules in a trigger-first order (76.88%), a pattern observed in both simple (66.36%) and complex (84.12%) rules, as demonstrated by the χ^2 tests [both $\chi^2_s > 89.26$, $p_s < .001$; see Table 5.18].

Considering the qualitative results we obtained in Study 1 we check if there were differences in the preference for trigger-first vs. action-first order between scenarios with different topics, without finding any differences (see Appendix E for χ^2 results of each scenario), as indicated by the χ^2 test, which found no significant differences in the distribution of preferences across scenarios [$\chi^2(11) = 7.74$, $p = 0.74$].

We evaluated simple and complex rules separately to explore how the components of state, event, and action were positioned within the rules, as visible in Table 5.19.

In the case of **simple rules**, which contain two elements, the state is most frequently omitted, occurring as “na” 59.52% of the time. Independently from the temporal feature, the trigger element included was commonly positioned first (see Table 5.19).

For **complex rules**, which contain all three elements, the state was most often placed in the second position (49.30%), situated between the event and the action. The event is commonly described first (in 50.73% of cases). While the action, generally appeared in the third position (72.10%). Indeed, the most prevalent order among complex rules

5.3. STUDY 3: EXPLORING NATURAL LINGUISTIC PATTERNS IN TRIGGER-ACTION RULES

Table 5.18: This table presents the frequency and percentage of rules ($N = 2050$) written in a trigger-first or action-first order, separately for simple and complex rules. It also includes the results of χ^2 tests comparing the orders' distribution in each category.

rule	n	order				χ^2	df	p
		action-first	%	trigger-first	%			
simple	835	281	33.65	554	66.38	89.26	1	< .001
complex	1215	193	15.89	1022	84.12	565.63	1	< .001
total	2050	474	23.12	1576	76.88	592.39	1	< .001

Table 5.19: The table presents in more detail how state-event and action were positioned or omitted in the simple and complex rules.

rule	pos.	state n	(%)	event n	(%)	action n	(%)
simple	1	225	(26.95)	329	(39.40)	281	(33.65)
	2	113	(13.53)	168	(20.12)	554	(66.36)
	na	497	(59.52)	338	(40.48)	0	(0.00)
complex	1	410	(33.75)	612	(50.37)	193	(15.89)
	2	599	(49.30)	470	(38.68)	146	(12.02)
	3	206	(17.00)	133	(10.95)	876	(72.10)

is event-state-action (ESA), observed in 43.13% of cases [$\chi^2(5) = 904.20$, $p < .001$; see Table 5.20 and Figure 5.8].

Expertise in Programming

In this subsection, we present the results considering the participants' varying levels of programming experience.

A total of 837 rules were written by participants in the *expert group* ($n = 66$), with 557 rules created by individuals with *high expertise* and 280 by those with *previous experience*. Non-expert participants ($n = 108$) wrote 1213 rules.

To examine whether the distribution of **simple** and **complex rules** differed across the three groups, we calculated the proportion of complex rules within each group. Specifically, 73.61% of the rules written by participants with *high expertise* were classified as complex, compared to 56.10% in the *previous experience* group and 53.42% in the *non-expert* group. A χ^2 test indicated a significant association between expertise level and rule complexity [$\chi^2 = 65.81$, $p < 0.001$]. Post-hoc comparisons with *Bonferroni* correction revealed that participants with *high expertise* generated significantly more

Table 5.20: The table presents the frequency and percentage with which elements were positioned within the complex rules ($n = 1215$). In addition to the action-first and trigger-first order, we classified as “other” those that were trigger-first but with the action placed in the second position.

	order	frequency	percent
action-first	AES	118	9.71
	ASE	75	6.17
trigger-first	ESA	524	43.13
	SEA	352	28.97
other	EAS	88	7.24
	SAE	58	4.77

Table 5.21: Chi-square test (χ^2) results for simple and complex rules (“r.”) distribution across participants with different levels of expertise in programming. The table includes observed frequencies, χ^2 values, and post-hoc comparisons with *Bonferroni* correction ($\alpha = 0.02$).

group	simple r.	complex r.	χ^2	df	p
High expertise	147	410	65.81	2	< .001
Previous experience	123	157			
Non-expert	565	648			
post-hoc pairwise comparisons			χ^2	df	p_{bonf}
High expertise vs. Previous experience			25.43	1	< .001
High expertise vs. Non-expert			63.30	1	< .001
Previous experience vs. Non-expert			0.50	1	0.48

complex rules than both the *previous experience* and *non-expert* groups (see Table 5.21).

Similarly, we analyzed whether there was a difference in the distribution of trigger-first vs. action-first orders based on participants’ level of expertise. The 88.51% of the rules written by participants with high expertise followed a trigger-first order, as did 76.43% of the rules written by participants with previous experience, and 71.64% of those written by non-experts. A χ^2 test showed that the trigger-first order was significantly more frequent in rules written by *high expertise* participants [$\chi^2(2) = 61.15$, $p < 0.001$; see Table 5.22].

5.3.8 Discussion

From the results, we observed that the **trigger-first** order was the most used by the participants when writing trigger-action rules in natural language (**RQ3**). This aligns

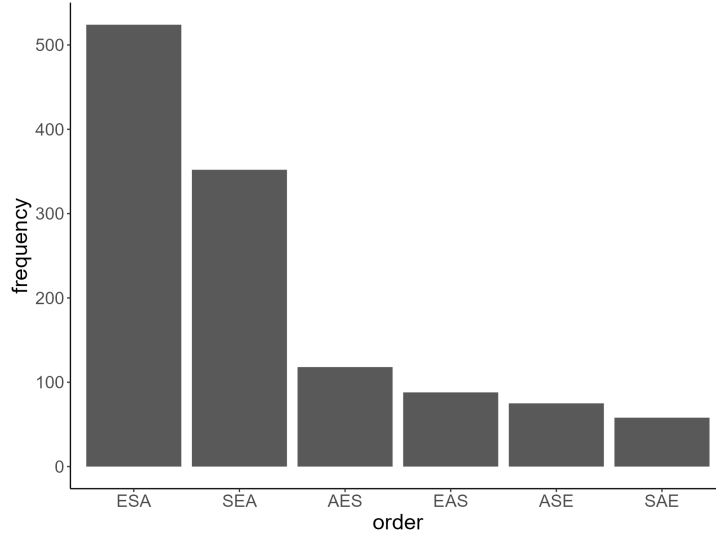


Figure 5.8: The bar plots shown in the figure represent the absolute frequency with which the state-event-action elements were presented in the complex rules ($n = 1215$) written by our participants.

with the findings of Pane et al. (2001) and supports the tendency of EUD TAP-based systems to adopt the *Trigger-Action* order, which is predominant in commercial platforms (e.g., *IFTTT*) and research prototypes (e.g., Brich et al., 2017, Bellucci et al., 2019, Dey et al., 2006) described in the literature (see section 4.2).

When analyzing how participants structured their complex rules, we observed that the ESA order was the most frequently used. This result suggests that users may find ESA a more natural way of conceptualizing automation rules. This finding somewhat contradicts the temporal sequence assumption of *Iconicity*, at least in terms of how participants generated prompts. Instead of strictly following a temporal sequence, as SEA does, participants prioritized defining the event first, aligning with the principles of *Figure-Ground theory*, which emphasize focusing on the most salient element first (i.e., the event). This tendency supports the adoption of the ESA order, as implemented in previous prototypes (e.g., Ardito et al., 2018; Brackenbury et al., 2019; García-Herranz et al., 2010; Zancanaro et al., 2022).

Expertise-related aspects. Experts and non-experts did not differ in their preference for trigger-first ordering. However, expert participants exhibited a stronger preference for the *trigger-first* structure compared to other participants (non-experts and those with previous experience). Additionally, we observed that high-expertise participants created more complex rules, incorporating all elements (state, event, and action), com-

Table 5.22: Chi-square test (χ^2) results for trigger-first vs. action-first order distribution across participants with different levels of expertise in programming. The table includes observed frequencies, χ^2 values, and post-hoc comparisons with *Bonferroni* correction ($\alpha = 0.02$).

group	action-first	trigger-first	χ^2	df	p
High expertise	64	493	61.15	2	< .001
Previous experience	66	214			
Non-expert	334	869			
post-hoc pairwise comparisons			χ^2	df	p_{bonf}
High expertise vs. Previous experience			20.73	1	< .001
High expertise vs. Non-expert			61.24	1	< .001
Previous experience vs. Non-expert			2.62	1	0.12

pared to non-experts or participants with previous experience. These findings suggest that, while the general tendency in ordering elements when writing natural language rules remains the same across expertise levels, individuals with less experience tend to create less structured rules. We can envision that TAP-based EUD systems should provide guidance for non-experts to consider all elements (state, event, and action), as they more frequently generate simpler rules. Furthermore, as greater variability can be expected in the order of elements in rules created by non-programmers, creating a structured environment that guides users in composing rules in an ESA order may be perceived as more natural by them supporting their working mental models.

Limitations

This study has several limitations. The sample was not balanced regarding expertise, with a higher number of non-expert participants ($n = 108$), compared to expert participants ($n = 66$). Additionally, individuals who are identified as experts but do not use programming languages daily may differ from other experts in their linguistic patterns when writing trigger-action rules – as suggested by our results. Therefore, recruiting more experts of both kinds would be necessary to enhance the generalization of the findings. Moreover, the analyses presented in this manuscript are still preliminary, as they lack an examination of the verbal cues employed by participants to describe events, states, and actions. Investigating these patterns could provide valuable insights.

Chapter 6

Studying Interaction with End-User Development Systems

Linguistic clues, such as the ones studied and discussed in the previous Chapter 5, can be implemented in different EUD TAP-based tools leveraging diverse compositional approaches. As a second objective of this dissertation, we examined two approaches that leverage linguistic clues with corresponding application examples:

- **Form-Based Compositional Visual Interfaces** (section 6.1). Implementing accurate linguistic constraints can better guide users and minimize ambiguous triggers, enhancing user comprehension and supporting the assumption of effective mental models. We studied mental models and reasoning strategies of elementary school teachers while using an EUD TAP-based solution. The platform guided them in composing states, events, and actions to define new behaviors and games in an educational smart device.
- **LLM-based system** (section 6.2). In systems where users express rules in natural language, variability in word choice increases ambiguity. We explored how LLMs handle the ambiguity of user-written trigger-action rules. We investigated the potential and limits of employing an LLM-based system to support people with intellectual disabilities in defining home automation.

We focused on complex use cases, such as controlling IoT environments in education and accessibility contexts. In both scenarios, it is required to manage numerous sensors, along with considering multiple individual factors, such as those of students and people with intellectual disabilities. In both cases, the possibility of customizing

and adapting the environment to individual potentials, characteristics, and needs is crucial, and the involvement of educators/teachers/caregivers is essential, given their expertise in the field and their knowledge of the specific needs of the actors involved.

6.1 Investigating Linguistic Clues in Form-Based Compositional Visual Interfaces

We investigated how a form-based compositional visual interface based on specific linguistic aspects can guide non-programmer users in defining trigger-action rules supporting the assumption of effective mental models and reasoning strategies.

To study challenges and potential in a complex and real-world application, we decided to focus on a specific field: the educational one. The **education domain** is an example of an IoT application field. Indeed, IoT in education is gaining increasing attention for the well-known potential of educational technology to enhance and support various aspects of the learning-teaching process (Al-Emran et al., 2020; Moreira et al., 2017; Ramlowat and Pattanayak, 2019).

However, teachers' and educators' limited knowledge of computers and programming represents a constraint in the customization of these tools (Delgado et al., 2015; Moreira et al., 2017). Furthermore, their lack of motivation and time to master the required technical skills hinder the use of educational technology in schools (Delgado et al., 2015; Moreira et al., 2017). Since teachers are the primary experts in the educational field, they should be able to directly modify/program their educational smart devices to create purpose-specific environments and integrate these devices into education strategies and plans (Wiedenbeck, 2005; Wiedenbeck and Engebretson, 2004). The use of EUD systems could effectively enable experts in education to customize their smart devices according to their expertise, goals, creativity, and student needs (Wiedenbeck, 2005).

However, when dealing with complex trigger-action rules for handling all the multiple aspects involved in customizing a teaching-learning path, non-programmers can face several challenges leading to errors and misinterpretations during the interaction with the system (e.g., Brackenbury et al., 2019; Huang and Cakmak, 2015).

As described in previous chapters, both from the literature (Chapter 4) and from our results (Chapter 5), language is a powerful tool in shaping the mental models of individuals. For this reason, we decided to study teachers' mental models and see how visual and compositional interfaces with specific linguistic features can guide the

assumption of more or less effective mental models.

6.1.1 Context: End-User Development for Educational Technology

Although many studies have focused on EUD solutions targeted at students, aiming to enhance pupils' programming-related skills (Abdollahinami et al., 2022; Andersen et al., 2021; Fronza and Pahl, 2018; Gennari et al., 2022; Mørch et al., 2019), other authors investigated EUD for educational domain experts, to allow them to define and customize teaching materials (Garzotto and Gonella, 2011; Romano et al., 2023; Valtolina and Matamoros, 2023; Zhong and Liu, 2014).

Wiedenbeck and Engebretson (2004) and Wiedenbeck (2005) explored the attitudes, necessities, and strategies of teachers already interested in programming to customize their teaching materials. The results of this unique approach yielded valuable insights. Teachers encounter many difficulties in using EUD platforms, which are mostly related to the lack of time, along with their poor programming knowledge (Wiedenbeck, 2005). Furthermore, they struggle to identify which events trigger a given output (Wiedenbeck and Engebretson, 2004). On the other hand, teachers' motivation and knowledge about their subject can help them navigate the system, and create EUD programs (Wiedenbeck, 2005). Indeed, teachers are often guided by their expertise in the school context when developing EUD solutions (Wiedenbeck, 2005). Finally, the authors highlight how brief, focused training is important to guide teachers in creating functioning EUD programs.

Valtolina and Matamoros (2023) proposed utilizing an Artificial Intelligence (AI)-based virtual agent to guide teachers in engaging with EUD systems. Their findings revealed that while non-programmers and novice programmers exhibit overlapping strategies and learning paths (similarly to Wiedenbeck and Engebretson, 2004), there remains a crucial need for EUD research to address the motivational and knowledge gaps among teachers regarding technology and programming. This focus is essential to fully leverage the potential of technological tools in educational settings.

6.1.2 Research Questions

Two research questions were addressed:

- **RQ4.** What are the characteristics of users' mental models when interacting with a "constrained" TAP-based EUD system?
- **RQ5.** Do the linguistic features (abstract vs. concrete descriptions) implemented

in a compositional TAP-based interface influence the assumption of different reasoning strategies leading to more effective mental models during EUD tasks?

Before conducting the main study (subsection 6.1.5), we designed a tangible IoT device connected with an EUD platform (subsection 6.1.3) and we conducted a pilot study (subsection 6.1.4).

6.1.3 The System

In education, tangible tools have a recognized potential for improving and supporting various aspects of learning. The physical interaction with the world (i.e., the grasping and manipulation of concrete, physical objects) allows children to concretely explore abstract concepts, which supports their cognitive development and learning (Piaget, 1964). In recent decades, a great effort has focused on creating and studying the application of tangible smart devices in the educational field to support the learning-teaching process and exploit its potential (Al-Emran et al., 2020; Moreira et al., 2017; Ramlowat and Pattanayak, 2019). For example, tangible and interactive devices were designed for supporting music (Waranusast et al., 2013), math, and language learning for young children (e.g., Almukadi and Stephane, 2015; Girouard et al., 2007).

Using smart devices creates an interactive and engaging learning environment (e.g., Al-Emran et al., 2020; Marquez et al., 2016). They may enable students to construct their knowledge (Forman and Pufall, 1988), while teachers can receive real-time data and information about students' learning achievements, performance, and errors (Al-Emran et al., 2020; Moreira et al., 2017; Moreira et al., 2018). All these elements can facilitate the customization of curricula to accommodate individual student abilities, which is essential to the learning process, particularly for those with specific learning disabilities (Benavides-Varela et al., 2020). Indeed, some recent works focused on creating tangible interaction devices for supporting individuals with specific learning disorders, such as dyslexia (Fan et al., 2016; Fan et al., 2017; Pandey and Srivastava, 2011) and dyscalculia (Erfurt et al., 2019). Manipulating tangible objects supports children during learning and development, while digital technology creates a motivating and engaging environment for children. Tangible interactive tools can effectively combine these two aspects. Nonetheless, to exploit their potential as educational devices, it is essential to empower teachers to customize their behaviors.

In collaboration with the *University of Bari*, we have designed and developed a system to support and promote math teachers in effectively using and programming a tangible smart device with elementary school children. The system includes a tangi-



Figure 6.1: The figure on the left side represents *SMARTER* 2.0 device, while the one on the right shows some mathematical tiles available for creating games on the device.

ble IoT device, named *SMARTER* (see Figure 6.1; Andrao et al., 2022b; Andrao et al., 2023b) and an EUD platform to personalize it, named *SENSATION* (see Figure 6.4; Desolda et al., 2021a).

Tangible Interactive Device: *SMARTER*

SMARTER (*SMART E-Rods*; Andrao et al., 2022b) is a tangible interactive tool based on IoT technology for math, inspired by the non-digital “Cusinarie rods” (Abreu-Mendoza et al., 2021; Lucow, 1964), also designed as a possible dyscalculia training device (see Appendix F for a figure representing the prototype). The underlying idea was to support teachers in teaching mathematical concepts (specifically, comparing quantities and arithmetic operations) by providing a tangible, interactive, and engaging environment suitable for children. Originally, *SMARTER* (Andrao et al., 2022b) consisted of:

- A **cardboard box** with a surface of 29.70×21.00 cm in which the game area is printed (five slots corresponding to the five RFID readers attached under the surface) and containing a speaker and an RGB LED, respectively for audio and visual feedback;
- A set of **tangible tiles** (with RFID tags attached) representing rods (for the quantity from 1 to 10), math operators, and symbols.

The five slots in the game area had different sizes and shapes to guide children in composing the games by placing in the correct slots the corresponding class of tiles (e.g., rod tiles should be placed in the 3rd and 5th slots in comparing quantities game;

see Appendix F).

Since teachers are the primary experts in the educational field, the actual version of the tool *SMARTER* 2.0 (Andrao et al., 2023b) evolved to give them more freedom/control and expressiveness in designing their own games and application contexts on the device. For this purpose, we adapted the tool design and architecture, aiming to create a more flexible system in which teachers can define the games' structures and their applications in different subjects, including math, language, music, etc. (see Figure 6.1 and Appendix G for the project design).

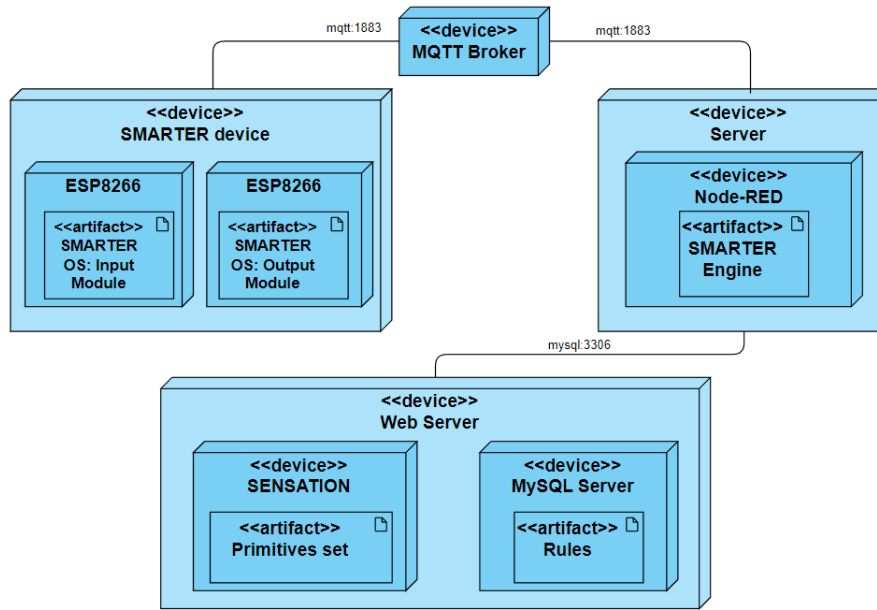


Figure 6.2: Deployment Diagram for the architecture of *SMARTER* 2.0 (cf. Andrao et al., 2023b).

We adapted the tool design and re-engineered the system architecture, aiming to enable teachers to define learning activities and subject fields more expressively and flexibly. Version 2.0 of *SMARTER* strives to increase the flexibility of its application contexts and empower teachers to define its behavior. The new architecture of *SMARTER* was designed to provide a modular system with as little coupling as possible between modules (see Figure 6.2). The following paragraphs describe the structural changes and re-engineering processes we have performed to achieve this goal.

SMARTER device. From a technological point of view, the structure of *SMARTER* 2.0

is identical to the previous version: a NodeMCU *ESP8266* board connected with five RFID readers (each one representing a slot for placing tiles), a speaker, and an RGB LED providing audio and visual feedback, respectively.

The most important update regards the **design of the form** (see Appendix G): the current version is a plywood box with the surface resized to 26.00×11.00 cm (see Figure 6.1 and Figure 6.3). We made this design choice by considering the real classroom context, i.e., the size of school desks in elementary school, assuming a possible future scenario where, for each child (in a single desk), several *SMARTER* boxes can be joined together (horizontally or vertically) to create a larger box to accommodate designs for more complex games. In addition, the five slots now have the same shape and dimension. Although children are no longer intuitively guided to place certain classes of tiles in defined slots (as in the first version), removing this constraint has allowed us to increase the system flexibility, the range of games, and contexts/subjects in which *SMARTER* can be employed. For example, *SMARTER 2.0* is suitable for supporting arithmetic operations, single- or multi-digit quantities comparisons, and sorting numbers in ascending or descending order.

An additional improvement involves tiles that represent various objects within the chosen domain. In the current implementation, these tiles depict mathematical characters. Specifically, this version includes a set of tiles embedded with Mifare® Ultralight RFID tags that represent digits and quantities ranging from 0 to 9, as well as operators (" $+$ ", " $-$ ", " $\times$ ", " $\div$ "), and symbols (" $=$ ", " $<$ ", " $>$ "). However, tiles can be adaptable to other domains. For example, they can be configured to represent letters, musical notes, and other symbolic elements.

SMARTER Operating System. *SMARTER 2.0* operates using a custom *SMARTER* Operating System (OS), which facilitates communication, basic information retrieval, and the execution of fundamental actions through the MQTT protocol. Specifically, the available *actions* include turning on/off the device LED, playing a sound, stopping the speaker, and resetting the board. Additionally, the OS allows the device to communicate basic *events*, namely the insertion and removal of a tile. Finally, it also supports querying the values on individual readers at any time, as well as retrieving the complete configuration of the board (five values, one per reader), which represents the board's *state*. The formalization of the MQTT commands is reported in Appendix H. The *SMARTER* OS is implemented in C++ and is divided into two modules, each deployed on one of the two *ESP8266* boards that constitute a *SMARTER* device. Both

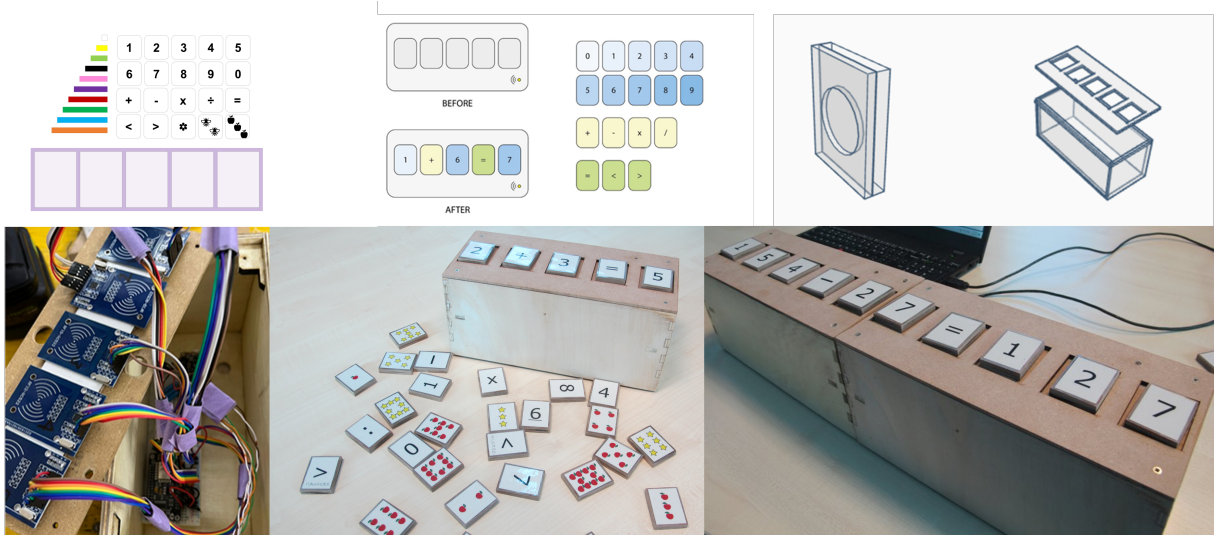


Figure 6.3: The figure represents *SMARTER* design (upper images) and implementation (lower images).

modules require the same *device_ID*. The first module is the input one, which manages the readers and handles MQTT messages related to *events* and *states*. It detects events on the RFID readers and promptly communicates them to the MQTT broker. It also listens for state requests and responds with either the full board configuration or the value of a specific reader, depending on the request.

The second module is the output module, which controls the actuators (the LED and the speaker), corresponding to the *actions*. This module listens for command messages and executes the relevant action. Tiles can store any string value of up to 255 bytes. The OS does not interpret the semantic content of the tiles (e.g., whether they represent digits, quantities, or symbols); this interpretation is managed by the *SMARTER* Engine, as discussed in the next paragraph. Tiles can be written or updated using any RFID writer compatible with Mifare®.

The implementation of this operating system allows the device to function independently, without requiring integration with other software modules. Consequently, a *SMARTER* device can be controlled remotely by directly sending and receiving MQTT messages that conform to the formalism specified in Appendix H. This design facilitates debugging and testing by ensuring the device’s functionality is encapsulated and uncoupled from the game logic (regardless of whether the tiles represent numbers, letters, or other symbols). Additionally, complex behaviors can be simulated by a human agent using a Wizard-of-Oz approach.

SMARTER Engine. A *SMARTER* Engine constitutes the core of the game’s logic. It defines the semantics and high-level behavior of rules and interprets the values of *tiles*. A *rule* is defined as a single event that, if verified, triggers one or more actions. It also includes zero or more states that are logically disjunct or conjunct (i.e., connected by “or” or “and” statements). Each rule is assigned a unique identifier. A *game* consists of one or more rules and operates with a specific *tile set*. A *tile*, which carries a string value, is interpreted according to the tile set used by the engine. For instance, the tile set “Arithmetic” may include tiles such as digits (0 to 9), symbols (“=”, “<”, “>”), and operators (“+”, “−”, “×”, “÷”). These values are parsed from the raw string representing the corresponding digit, symbol, or operator contained in a tile.

Each engine specifies a set of primitives (actions, events, and states) that can be arbitrarily complex and/or domain-specific. For example, an engine can define the logic for the event “A digit greater than ‘5’ is placed” or for the state “The board is empty”. The engine reads and interprets rules defined in a specific JSON format. These rules can be manually written or generated by an EUD interface like *SENSATION* (as described in the next subsection; Desolda et al., 2021a), enabling the use of human-readable rules (referred to as “Rules” in Figure 6.2). This design allows the creation of distinct engines to implement different logic in various domains. To ensure compatibility with the system, an EUD tool must generate JSON rules that exclusively contain primitives defined in the engine. A *SMARTER* Engine manages an MQTT channel to communicate with one or more *SMARTER* devices, receiving basic events, requesting states, and executing actions. Each *SMARTER* device can connect to at most one *SMARTER* Engine. This architecture allows a developer to design an engine that controls multiple *SMARTER* devices as if they were a single board, significantly increasing the available positions for games using up to 10, 15, or more tiles. The engine can be implemented with any server as long as it can function as an MQTT client. We used *Node-RED*, a visual task automation tool that facilitates MQTT communication and allows the definition of JavaScript-written submodules for specific logic (as shown under “Server” in Figure 6.2).

End-User Development Platform: *SENSATION*

The functionalities of *SMARTER* can be customized through *SENSATION* (Desolda et al., 2021a), an EUD form-based compositional visual platform that supports and guides users in the creation of trigger-action rules with drop-down menus (Figure 6.4). Following the guidelines of Huang and Cakmak (2015), the platform provides different

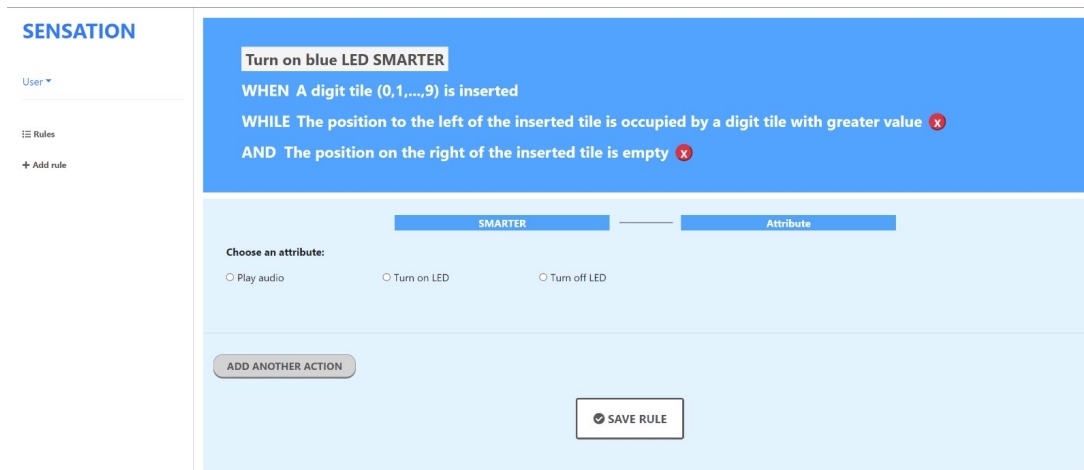


Figure 6.4: The figure presents a screenshot of the compositional visual interface of *SENSATION*.

conjunctions to introduce states and events, separating these two occurrences at the interface level to facilitate users' comprehension.

Initially, this authoring tool was created to support users in defining trigger-action rules for home automation systems, guiding them in composing actions, events, and states (Desolda et al., 2021a). We have implemented language primitives in *SENSATION* to enable teachers themselves to define new games of *SMARTER* – currently only for math games (see Figure 6.4). Teachers can customize *SMARTER* behavior (specifically, its visual and audio feedback on children's actions) according to their educational goals and the children's needs, ages, and skills. In the first version of the interface, users could compose complex trigger-action rules in the form: "Do <action(s)> When <event> While <state(s)>". Considering the results of the pilot study (subsection 6.1.4), we changed the order of the elements accordingly in "While <state(s)> When <event> Do <action(s)>".

Primitives for **actions** consist of commands to control visual and acoustic feedback (e.g., "Turn on blue LED *SMARTER*"). States and events describe, respectively, the operations that teachers and children can perform on the tool (i.e., insertion and removal of tiles) and the tool configuration at that moment. An example of an **event** description is "*when a digit tile is inserted*", while an example of a **state** is "*while the position to the right of the inserted tile is empty*".

SENSATION allows the software developer to define custom primitives sets (for *actions*, *events*, and *states*) to reflect the logic inside a *SMARTER* Engine ("Primitives sets" in Figure 6.2). Obviously, the same primitives must be already defined within

the *SMARTER* Engine. *SENSATION* is written in Laravel and Vue.js and runs on an Apache web server. User-defined rules are saved in a MySQL database in a JSON format compatible with a *SMARTER* Engine; the latter can access these rules via MySQL Client/Server protocol and interpret them.

Children Using *SMARTER*

SMARTER 2.0 was used by children during the *Muse Tech* MUSE-TechParty event in 2023. On that occasion, 19 elementary school children (aged 7 to 10) freely interacted with *SMARTER*, which was programmed to support arithmetic operations and numerical comparisons. The children gave us drawings of *SMARTER*, shown in Figure 6.5.

In the same year, *SMARTER* 2.0 was also employed in a study investigating the role of emotions in learning mathematical concepts by Saccardo et al. (2024). Approximately half of the recruited elementary school children ($n = 31$; 13 females and 18 males; aged 6 to 7; $M = 6.50$, $SD = 0.29$ years) used *SMARTER* to complete tasks on comparing quantities, while the other half practiced with paper-and-pencil. Notably, using *SMARTER* appeared to influence the role of emotions in learning mathematical concepts, reducing the differences between positive and neutral emotional cues observed in the paper-and-pencil group. This effect may be attributed to the positive emotions, curiosity, and interest stimulated by the novelty of *SMARTER*.

These two occasions revealed to us that elementary school children could easily use *SMARTER* without requiring specific training and that they could enjoy using it.

6.1.4 Study 4: Exploring Teachers Mental Models in Composing and Debugging Trigger-Action Rules

We conducted a first pilot study with five elementary school math teachers in which they were asked to interpret, create, and debug trigger-action rules through *SENSATION* (Desolda et al., 2021a) to define new games of *SMARTER* (Andrao et al., 2023b). The pilot study described in this subsection refers to the one described in Andrao et al. (2022a) and Manfredi et al. (2022).

The results of these pilot studies highlighted the existence of different composition (Andrao et al., 2022a) and debugging strategies (Manfredi et al., 2022). Certain reported strategies were found to be more effective in creating functioning and accurate rules and identifying errors (**RQ4**).

Table 6.1: The table reports the demographic information of the participants, including their years of experience as teachers (referred to as "teaching") and their use of digital devices in their work as teachers (referred to as "digital devices").

id	age	gender	education	teaching	digital devices
P1	27	F	Master	1	Always
P2	24	M	Bachelor	2	Always
P3	30	F	Master	6	Always
P4	26	F	Master	2	Always
P5	40	F	Master	10	Sometimes

Participants

Five (5) elementary school math teachers (one male, four females) with no previous programming experience were recruited. Teachers were aged between 24 and 40 ($M = 29.40$, $SD = 6.31$ years). Details about participants can be found in Table 6.1.

Materials

In this pilot study, we used an early version of *SENSATION* in which we implemented linguistic primitives to define mathematical games on *SMARTER*. In this version, participants could compose trigger-action rules in the form "*Do* <action(s)> *When* <event> *While* <state(s)>" as it was in the original version of the platform presented in Desolda et al. (2021a).

We created two sets of rules to run the game "sort the numbers in ascending order" on *SMARTER*:

- **Correct rule set** – composed of nine rules. See Table 6.2 for rules descriptions.
- **Non-functioning rule set** – composed of 11 rules. Specifically, two additional rules were included: one inconsistent (R10 introduces a conflicting action to the same trigger as R1, R2, R3, and R4) and one redundant (R11 produced the same outcome as R7).

Task and Procedure

The study was conducted remotely (using *Qualtrics* and *Zoom*) through semi-structured individual interviews with a *thinking aloud* protocol (Boren and Ramey, 2000).

6.1. INVESTIGATING LINGUISTIC CLUES IN FORM-BASED COMPOSITIONAL VISUAL INTERFACES

Table 6.2: The table lists the nine rules of the correct set of rules to make the “sort numbers in ascending order” game work on *SMARTER* and the two incorrect rules.

id	rule	description
<i>correct rules set</i>		
R1	Turn on the blue LED WHEN a digit tile (0,1,...,9) is inserted WHILE the position to the left of the inserted tile is empty AND the position to the right of the inserted tile is empty	Incomplete
R2	Turn on the blue LED WHEN a digit tile (0,1,...,9) is inserted WHILE the position to the left of the inserted tile is occupied by a digit tile with a smaller value AND the position to the right of the inserted tile is empty	Incomplete
R3	Turn on the blue LED WHEN a digit tile (0,1,...,9) is inserted WHILE the position to the right of the inserted tile is occupied by a digit tile with a larger value AND the position to the left of the inserted tile is empty	Incomplete
R4	Turn on the blue LED WHEN a digit tile (0,1,...,9) is inserted WHILE the position to the left of the inserted tile is occupied by a digit tile with a smaller value AND the position to the right of the inserted tile is occupied by a digit tile with a larger value	Correct
R5	Audio.mp3 error sound WHEN a digit tile (0,1,...,9) is inserted WHILE the left position of the inserted tile is occupied by a digit tile with greater value	Error
R6	Audio.mp3 error sound WHEN a digit tile (0,1,...,9) is inserted WHILE the position to the right of the inserted tile is occupied by a digit tile with a smaller value	Error
R7	Turn off LED WHEN any tile is removed	Removing
R8	Turn on the red LED WHEN an operator tile (+, -, "×", "÷") is inserted	Alternative
R9	Turn on the red LED WHEN a symbol tile ("=", "<", ">") is inserted	Alternative
<i>incorrect rules</i>		
R10	Turn on the green LED WHEN a digit tile (0,1,..., 9) is inserted	Inconsistent
R11	Turn off LED WHEN any tile is removed WHILE the position to the left of the inserted card is occupied by any digit tile AND the position to the right of the inserted tile is occupied by any digit tile	Redundant

After giving informed consent, each participant was introduced to the system and the task with a brief demonstration of the prototype tool and the interface (in *Qualtrics*), supported by videos and images showing the functionality of the system. After that, participants were involved in six phases:

- **Familiarization phase:** Participants explored *SENSATION* and interacted with a pre-created correct rule set for the game “sort the numbers in ascending order” (see Table 6.2).
- **Debugging task:** Once back on *Qualtrics*, participants were explicitly asked to think aloud about how they would test the functionality of the rules. They were then asked to indicate one (or two or three) actions they would take to verify if the rule set they created was functioning as intended. Specifically, they were asked:

“How would you test the functionality of the rules you created?” and “If you

could take only one (or two or three) actions to check if the rule set you created works, what would they be?"

- **Error-finding task:** Participants were then asked to return to *SENSATION*, where a different version of the rules for the game "sort the numbers in ascending order" containing two errors was prepared (see Table 6.2). They were asked to identify the reasons why the rules were not working.
- **Compose trigger-action rules:** Participants were then asked to create rules for the task "compare two quantities" where two quantities, composed of one or more digits, are first placed, and the feedback – positive or negative – is given by the device after entering the symbol ($=$, $<$, $>$) that completes the expression.
- **Demographic survey:** On *Qualtrics*, participants answered demographic questions (e.g., gender, age, education, use of digital devices in school, and teaching experience).
- **Debriefing and final comments:** Finally, participants were debriefed and asked for comments, impressions, and suggestions for improving the system.

Each session lasted approximately 90 minutes and was video-audio recorded. The qualitative data collected were analyzed using thematic analysis by two independent researchers (Braun and Clarke, 2006), who also examined the rules created in *SENSATION*.

Preliminary Results

Debugging. The qualitative analysis revealed three distinct debugging strategies that participants would employ (see Manfredi et al., 2022 for more details and Table 6.3). These strategies showed partial similarities to those previously identified in novice programmers (Fitzgerald et al., 2008; Fitzgerald et al., 2010; Li et al., 2019). Initially, all participants envisioned testing the rules in real-world scenarios, specifically in classroom settings with their students. Some adopted a *trial-and-error* approach (e.g., "I would try to make a rule; if I realized something was wrong, I would go back and change it"), while others (P2 and P3) preferred a more functional, child-focused procedure by testing directly with the children.

Some participants (P2, P4, P5) employed a "step-by-step" strategy, suggesting the sequential insertion of tiles – first a correct tile, followed by an incorrect one – while

Table 6.3: In the table are described the three strategies we highlighted from the qualitative analysis of participants' answers to the debugging task and the two approaches adopted in the error-finding one.

strategy	description	participant
<i>debugging</i>		
<i>step-by-step</i>	placement of one tile at a time, checking the result at each step	P2, P4, P5
<i>direct solution</i>	placement of all tiles, with only one final check	P1
<i>alternative case</i>	placement of symbol tiles to test alternative cases	P1, P3
<i>error-finding</i>		
<i>rule-by-rule</i>	error search limited to individual rules	P2, P4
<i>rule-set</i>	error search extended to rule relationship	P1, P3, P5

evaluating the outcomes after each step. Conversely, P1 imagined a different strategy, arranging all tiles at once to observe the overall result. This approach, aimed at achieving a *direct solution*, aligns with behaviors noted in novice programmers apparently linked to misunderstandings about system functionality (Fitzgerald et al., 2008; Fitzgerald et al., 2010; Li et al., 2019). Additionally, P1 and P3 imagined testing by placing symbol tiles to explore the *alternative case*, whereas P5 considered this strategy unnecessary due to the task's objective of sequentially sorting numbers.

Error-Finding. Consistent with the literature (De Russis and Corno, 2015), inconsistency was the most frequently identified error (noted by 3 out of 5 participants), whereas redundancy proved to be the most challenging to detect (only P5 identified it). Participants employed two distinct strategies during the error-finding task (see Table 6.3):

- *Rule-by-rule approach:* This strategy involved examining each rule individually without considering their interrelationships. Participants using this approach (P2 and P4) were unable to identify errors within the set of rules. Additionally, they attributed the errors to the excessive number of rules, which they perceived as unnecessary and confusing.
- *Rule-set approach:* This method involved analyzing the rules collectively, taking into account their relations. Participants who adopted this approach (P1, P3, and P5) were able to identify some of the errors. Eventually, only participant P5 was able to identify all errors.

The error-finding results for each participant are summarized in Table 6.4.

Table 6.4: The table shows error-finding task results for each participant.

participant	approach	identified errors
P1	<i>rule-set</i>	inconsistency
P2	<i>rule-by-rule</i>	none
P3	<i>rule-set</i>	inconsistency
P4	<i>rule-by-rule</i>	none
P5	<i>rule-set</i>	all

Composition. The qualitative analysis of verbal reports revealed that during the composition task, nearly all participants expressed the need for *concrete visual* support to facilitate rule construction. For example:

- P2: *"It would be interesting to be able to visually see how this mini-scenario develops while I compose the rules"*.
- P4: *"It might be useful, I think, to create rules with pictures that actually show what there is to do"*.

However, P2 appeared capable of mentally visualizing the scenarios without relying on concrete graphical support, enabling the creation of functioning rules while considering alternative scenarios:

- P2: *"I am thinking of a graphical representation [of the rule], and it seems to me that the rule could work in this way"*.

Instead, P1 helped herself with mental representations of specific and concrete configurations:

- P1: *"I have to compare quantities, so, for example, I have to place [in SMARTER] 'one less than two'"*.

Similarly, P5 reasoned using *specific configurations* of tool spaces, aiding her in visualizing alternative scenarios, while P3 utilized pen and paper to sketch tool spaces and configurations. Additionally, P3 applied abstract reasoning during rule composition, generalizing the verbal primitives to multiple cases. Notably, when composing rules, P3 and P5 focused on concrete configurations of the tool (i.e., states), enabling them to produce a functional and exhaustive set of rules while considering alternative scenarios. Although P1 focused on describing possible configurations, she eventually created accurate yet case-specific rules without abstract generalizations.

Table 6.5: The table reports the composition task results, including the number of rules created, the presence of errors in the rule-set, and the focus of their reasoning.

id	approach	n. rules	errors
P1	<i>focusing on states</i>	7	none
P2	<i>focusing on actions</i>	8	redundancy
P3	<i>focusing on states</i>	6	none
P4	<i>focusing on actions</i>	2	none
P5	<i>focusing on states</i>	5	none

Conversely, P2 and P4 did not engage in reasoning about state compositions and instead relied exclusively on primitives to express whether an expression was true or false. For instance, P4 wrote only two rules, finding the state primitives irrelevant to her case, and concluded that she preferred simplicity. Participant P2 created eight redundant rules, likely due to his focus on the action components of the rules, aiming to customize feedback for the child.

Some participants envisioned the game scenario with the child before interacting with the interface and attempted to adapt the available primitives to achieve their objectives. For example, P1 worked to create a rule for final feedback and looked for the primitive event "*when all tiles are placed*". However, comparison with the interface primitives sometimes prompted her to reconsider her logic regarding trigger-action rules and to consider the effect of one individual rule on the whole rule set. She questioned her reasoning because the primitives allowed for only one tile at a time, contrary to her intention of using all tiles simultaneously.

Despite differences in strategies, nearly all participants successfully composed meaningful rules without errors (see Table 6.5).

Overall, the participants were motivated and enthusiastic about customizing the tangible tool, offering suggestions to enhance the interface's usability and make it more suitable for educational settings. They highlighted the tool's potential for creating complex games or supporting children with learning difficulties.

The findings of the pilot study provide insights into strategies for debugging, finding errors, and composing trigger-action rules. Specifically, the results suggest that participants who focused on the tool's *specific configurations* (states) were more successful in creating complete and comprehensive rules, whereas those who concentrated on *feedback mechanisms* (actions) tended to produce overly simplified or redundant rules.

Consequently, in the main study, the graphical layout of rule elements on the platform was modified to guide users through composing states first, followed by events, and finally actions (as discussed in and detailed in subsection 6.1.5). Furthermore, the findings indicate that some verbal primitives were overly abstract, necessitating more concrete examples. On the other hand, abstract content can potentially support users in questioning their approach. These insights informed the development of two versions of primitives described in subsection 6.1.5.

6.1.5 Study 5: Studying Teachers Mental Models on Trigger-Action Programming

This study investigated reasoning strategies and mental models of math teachers during the composition and debugging of trigger-action rules (Desolda et al., 2021a) to understand how they interpret and represent the functionality of *SMARTER* (Andrao et al., 2023b). The objective was to identify effective ways to guide teachers in developing effective mental models and reasoning strategies that enhance their interaction with the system and TAP-based platforms more broadly. Examining teachers' mental models is essential for understanding how they perceive, adopt, and use smart devices in educational contexts. These insights are crucial for designing effective EUD solutions integrated into educational technologies, eventually bridging the gap between innovative tools and meaningful teaching outcomes.

This research focuses on a specific factor influencing the usability and effectiveness of TAP: the level of abstraction of verbal primitives describing events, states, and actions presented to the user, categorized as either "concrete" or "abstract" (Corno et al., 2017; Corno et al., 2019b). Concrete triggers and actions relate to specific sensors and actuators in the environment, whereas abstract ones refer to general categories (see section 4.3). Although abstract rules enable the creation of more general and flexible procedures, concrete rules may be more intuitive for non-programmers.

In the study, we focused on teachers' mental models during the interaction (**RQ4**), and using a within-subject design, the research examined how concrete and abstract verbal primitives affect teachers' understanding of events, states, and actions, as well as their combination during trigger-action rules manipulation (**RQ5**). This study corresponds to the research presented in Andrao et al. (2025). It was approved by the ethics committee of the *University of Trento* and conducted between June and July 2023¹.

¹Study design, system development, and pilot studies started in 2022.

Participants

Twenty-eight (28) female elementary school math teachers, aged between 27 and 61 ($M = 47.04$, $SD = 8.76$ years), participated in this study. Although no gender-specific inclusion criteria were set, approximately 95% of elementary school teachers in Italy are female (according to *UNESCO Institute of Statistics*). Participants were recruited through snowball sampling from 13 elementary schools located in the surrounding areas of Trento, Verona, and Padova (northern Italy). Their teaching experience ranged from 2 to 40 years ($M = 21.39$, $SD = 9.41$ years).

Nineteen (19) teachers taught a single grade, while nine taught across multiple grades within the elementary school. Specifically, six participants taught year 1, four taught year 2, 10 taught year 3, nine taught year 4, and eight taught year 5. In terms of educational background, 14 participants held a Master's degree, three had a Bachelor's degree, and 10 had a High school diploma. All participants were native Italian speakers and had no prior experience with formal programming languages. However, eight reported familiarity with tools like *Scratch* (Resnick et al., 2009), and one had experience with an EUD platform. Additionally, all participants reported regularly using digital devices in the classroom, with 16 indicating "always" and 12 reporting "sometimes".

Materials

We used the system described in subsection 6.1.3, composed of a tangible device, *SMARTER* (Andrao et al., 2023b), and a web-based EUD platform, *SENSATION* (Desolda et al., 2021a).

We prepared two versions of the EUD platform: one version using concrete and case-specific primitives (e.g., "*The tile 7 is placed in position number 2*" is a concrete event), and the other version using abstract and generalizable ones (e.g., "*The last tile placed is a digit tile*" is an abstract event). Apart from the primitives, the two platforms were identical (see Figure 6.6 and Appendix I for the list of verbal primitives).

Procedure and Task

Before the study, participants provided informed consent. During data collection, they were seated at a table where the experimenters placed the tangible device, a set of tiles (including digits, symbols, and operators), and a computer for accessing the EUD platform (see Figure 6.7). A *thinking aloud* protocol (Boren and Ramey, 2000) was employed

ANONYMIZED

User
Instructions
Add instruction

While In the set of tiles to the left of the last tile placed there is a math operation
AND In the set of tiles to the right of the last tile placed there is a number
AND The last tile placed is a symbol ($>$, $<$, $=$)
When The expression becomes true
Turn on green LED SMARTER

SMARTER

The expression becomes ...

Choose an attribute

☐ A tile ... is placed
☐ A tile ... is removed
☒ The expression becomes ...

The expression becomes
true

SAVE INSTRUCTION

ANONYMIZED

User
Instructions
Add instruction

While Position 1 is occupied by 5
AND Position 2 is occupied by +
AND Position 3 is occupied by 2
AND Position 5 is occupied by 7
When = is placed in position 4
Turn on green LED SMARTER

SMARTER

... is placed in position ...

Choose an attribute

☒ ... is placed in position ...
☐ ... is removed from position ...

=

Position(s):
☐ 1
☐ 2
☐ 3
☒ 4
☐ 5

SAVE INSTRUCTION

Figure 6.6: The figure shows how the same rule can be created with the abstract (A) or concrete (B) version of the platform.

to elicit teachers' mental models during the interaction. After a brief introduction to the device and the EUD platform, participants completed two tasks. Each task involved:

- I. Watching a video showing someone performing a math exercise on the tangible device (either comparing or subtracting two quantities);
- II. Using one version of the EUD platform (e.g., the concrete version) to create trigger-action rules governing the device's behavior.

Upon finishing the first task, participants repeated the procedure using the other version of the platform (e.g., the abstract version) for creating the second math exercise. In both exercises, correctly placing the tiles activated a green LED light, while incorrect placements triggered a red LED light. The order of platform versions was counterbalanced across participants. Additionally, the order of tasks (comparing or subtracting two quantities) was balanced across participants and conditions. This counterbalancing controlled for potential *learning effects*.

A researcher was present during each session to assist participants if they encountered difficulties². After completing the tasks, participants filled out a questionnaire. Each session was video and audio recorded, lasting approximately 54 minutes (range: 32.04 – 84.29 min; $M = 54.04$, $SD = 12.21$ min).

Analysis

Inductive thematic analysis (Braun and Clarke, 2006) was employed to examine the video-audio recordings of the sessions. Three participants were excluded due to technical issues with recording or system usage. For the remaining 25 participants, two researchers (R1, R2) transcribed the recordings, and three researchers (R1, R2, R3) independently coded the verbal reports. While R1 and R3 were involved in the platform's design, R2 was not. An inductive approach was adopted (Clarke and Braun, 2017), as no prior assumptions were made about math teachers' mental models. Following the initial coding round, disagreements were resolved through discussion, with the process repeated iteratively until a consensus was reached. The categorization in *metaphors*³ emerged from the inductive analysis of the codes and was not predefined. Although the thematic analysis covered a broader scope, only the findings related to mental models are presented here for simplicity.

²In such cases, the rules were excluded from the analysis.

³In this context, we define *metaphors* as specific aspects of working mental models that affect how individuals interact and perform.

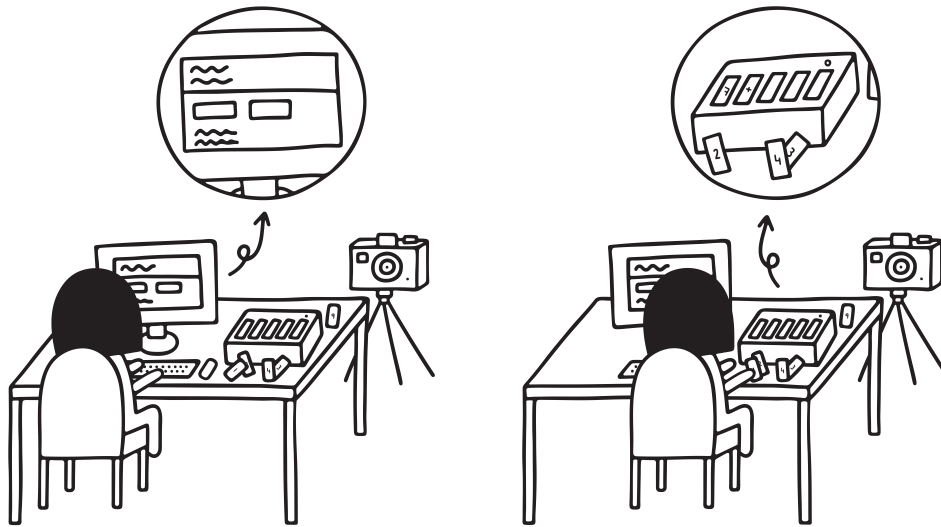


Figure 6.7: The figure represents the study setting: the participant is in front of the table with the web-based platform and the tangible device. Behind the table, the video camera captured the interaction with the tools and recorded the verbal interaction. On the left, the participant is represented during the interaction with the platform to create the trigger-action rules, while on the right, during the testing rules on the device.

Since mental models are dynamic and participants learned to use the tools through interaction, metaphors were classified as either *primary* (dominant during interaction) or *secondary* (emerging but less influential). The analysis also revealed an unexpected interaction between participants' conceptual models of computers and their metaphors, influencing how they manipulated the devices – this interaction became apparent during the coding sessions.

Participants' performance was also evaluated by R1, R4, and R5, based on rule correctness (0% = "incorrect", 50% = "partially correct", 100% = "correct") and rule-set completeness (0% = "incomplete", 50% = "partially complete", 100% = "complete"). Detailed descriptions of these metrics are provided in Table 6.6. Assessing correctness and completeness helped identify which metaphors were most effective in supporting users' rule manipulation, thereby reducing potential bias (e.g., presuming trigger-action logic to be the optimal approach).

Table 6.6: Description of metrics used for evaluating rules. For each participant, the aggregate scores of the created rules were calculated (i.e., average rule correctness and rule-set completeness).

metric	score	description
<i>rule correctness</i>	0%	The rule does not work (e.g., " WHILE the position number 3 is empty , WHEN the tile '<' is placed in position number 3, Turn on LED green" - the condition is unmet and the tile placement fails to trigger any action: position 3 cannot be empty if a tile is placed there)
	50%	The rule works but has some errors (e.g., " WHILE the position number 2 is occupied by the tile '7' AND the position number 4 is occupied by the tile '9' AND position 3 is occupied by '<' WHEN the tile '<' is placed in position 3, Turn on LED green" - contains redundant/unnecessary conditions)
	100%	The rule works perfectly without errors (e.g., " WHILE the position number 2 is occupied by the tile '7' AND the position number 4 is occupied by the tile '9' WHEN the tile '<' is placed in position 3, Turn on LED green")
<i>rule-set completeness</i>	0%	The rules are not complete for the required task (e.g., missing conditions to satisfy the task)
	50%	The rules are partially complete for the task (e.g., rules address part of the "state" but leave some other conditions out)
	100%	The rules satisfy the task requirements completely

6.1.6 Results

From the thematic content analysis, a total of 86 codes emerged and were organized by topic, resulting in nine themes⁴. Three of these themes guided the definition of *metaphors*, underlying participants' working mental models: "**command**", "**react**", and "**instruct**". As explained in the following subsections, we found that some metaphors, rather than others, helped or hindered the creation and testing of trigger-action rules effectively. Further, three other themes focused on participants' *conceptual mental models* of computers, the process of *learning* by interacting with the system, and the influence of *abstract-concrete* primitives, respectively.

Metaphors in Rules Composition and Testing (RQ4)

Three themes that emerged from the analysis led to the definition of three distinct metaphors, described in detail in the below paragraphs.

Command – Commanding the System: Instructions Step-by-step. Participants whose working mental model led to the definition of the "command" metaphor were driven by the belief that they needed to direct/control the device with a step-by-step approach (e.g., P4 - concrete primitives: *"I have to do it precisely step by step [...] so I give [the system] the instructions exactly step by step"*). The "command" metaphor appears in varied forms (see Table 6.7 for examples of each occurrence of the metaphor):

- **Creating a rule for each tile placement.** Participants wanted to define rules for every single tile placement – covering all the possibilities – so that the system would "register" them regardless of being linked to an action (LED on/off/color).
- **Creating a state for every device slot.** Participants assumed the states to be interpreted sequentially by the system, similarly to a flow.
- **Declaring a state/event as rules without linking them to actions.** Participants tried to save these incomplete rules, complaining about not wanting to define the action.

The "command" metaphor was the most frequently emerged from participants' verbal reports (see Table 6.10). Of 25 participants, 21 showed instances of this metaphor while composing rules, and eight participants made it their primary metaphor. Although "command" was the most common and spontaneous metaphor assumed, it did not

⁴Codes and themes are reported in Appendix J.

Table 6.7: Examples of participants' quotes where the "command" metaphor is observable while using concrete primitives and abstract ones.

metaphor occurrence	concrete condition	abstract condition
<i>Trying to create a rule for each tile placement</i>	P20 – "[reading the rules created] Turn on the white LED when '1' is placed in position 2, while position 1 is occupied by '1' [...] Turn on the white LED when '7' is placed in position 5, while position number 4 is occupied by '3' [...]"	P6 – "I wanted to create all the rules... but that's not feasible."
<i>Trying to create a state for every tile placement</i>	P4 – "Okay, I add another state: position 4 is occupied by 'equals' symbol, position 5 is occupied by '5' [...]"	P15 – "What I didn't understand is why does it [the platform] ask me to insert the 'last tile'? Shouldn't I start from the first tile? [...] Why does it mention the 'last tile'?"
<i>Creating a single state or event with no action</i>	P1 – "What I don't understand is why it asks me to turn the light on or off at every instruction."	P24 – "The first thing I have to do is to place the number, right? Then, a tile... digit is placed. Can I go on? [tries to save the rule composed only of the event]"

help teachers in rules composition because it conflicted with trigger-action logic. Those who consistently relied on this metaphor showed lower levels of correctness (32.83%) and completeness (33.33%).

React – Reacting to Events: Understanding the Trigger-action Logic. Participants whose working mental models led to the definition of the "react" metaphor embraced – at least partially – a trigger-action logic. This approach often enabled them to create effective trigger-action rules. They recognized states as protracted circumstances, while events as signals that trigger the action when a given circumstance is met. The "react" metaphor occurred in various ways (see Table 6.8):

- **Understanding that the last tile placed triggers the action.** Participants correctly stated that the actions are triggered depending on the value of the last tile placed on the board.
- **Understanding the concept of "last tile".** Participants identified that "last tile"

in the abstract primitives refers to the tile placed/removed last rather than the rightmost slot on the board.

- *Grasping the importance of the order of states and events.* Participants understood that the defined state(s) must be verified at the moment of the event to trigger the rule.

Table 6.8: Examples of participants' quotes where the "react" metaphor is observable while using concrete primitives and abstract ones.

metaphor occurrence	concrete condition	abstract condition
<i>Understanding that the last tile placed triggers the action</i>	P6 – "[...] the digits are indifferent in the end, it is the connector that makes the difference: whether I choose the 'greater than' or 'less than' [symbols]."	P7 – "[...] the position of the last tile placed [...] could be the result, which I place to turn on the green light."
<i>Understanding the concept of "last tile"</i>	N/A (concrete primitives did not mention the term "last tile")	P17 – "If I want to make a comparison between numbers [...] the last tile placed is not the last one in the device in order [indicated the fifth slot] but as in this case [indicated the fourth slot] the fourth one because that's how the tool compares [the quantities]."
<i>Grasping the importance of the order of states and events</i>	P9 – "So, position number 5 is empty... this is the starting situation, right? Ok, so let's place a digit in position 5 [...]"	P17 – "So I place the tiles... I have to think that the last one that I will place must match the rule I gave."

Participants did not always fully adopt the "react" metaphor. Sometimes, they consider that just defining the state or the event alone is enough to trigger an action on the system, leading to what can be described as a "state-based" or "event-based" approach. For instance, participant P13 states (concrete primitives): *"I mean, absurdly, I could only set the slot number 3 [in the state]"*.

This tendency often occurs as participants approach the "react" metaphor and recognize states and events as separate and distinct elements. However, they still struggled to fully embrace the trigger-action logic. This was also evident when participants composed a state and an event representing the same element, thus redundant (e.g., *"WHILE the last tile placed is a symbol WHEN a symbol is placed, turn on the green LED"*

with the abstract primitives or "*WHILE position number 3 is occupied by > WHEN > is inserted in position number 3, turn on the red LED*" with the concrete primitives). In these cases, even though they were approaching the trigger-action logic, the metaphor was still rudimentary and incomplete because state and event were not completely distinguished. Employing the "react" metaphor led to an average correctness of 75.50% and a completeness of 65.00%.

Instruct – Instructing the System: Teaching Experience. Participants whose mental model accommodated the "instruct" metaphor approached rules composition mirroring the classroom setting. Generally, the "instruct" metaphor was observed when participants mentally simulated scenarios involving either a student or an entire class to create and test trigger-action rules. This approach might be a natural expression of teachers' expertise. The "instruct" metaphor has two subgroups (see Table 6.9):

- ***Instruct-assignment.*** This metaphor emerged when participants focused on defining the objectives of the task, attempting to "instruct the system as if it were a student". Only two participants (P21, P23) adopted the "instruct-assignment" as their primary metaphor (see Table 6.10). Participant P21 was able to create rules that functioned (at least partially), while participant P23 could only create rules with substantial support from the researcher.
- ***Instruct-child.*** Participants who adopted this metaphor focused on the child's actions, mentally simulating what the child should or should not do to complete a class exercise. They represent the initial situation provided to the child as a state and the child's move as the event. This approach leans towards the trigger-action logic but sometimes encloses the idea of sequence between state-event-action in terms of before and after. Although only one participant adopted the "instruct-child" as the primary metaphor (see Table 6.10), it appears to support teachers in formulating trigger-action rules.

The "instruct-assignment" metaphor resulted in lower outcomes (correctness = 45.42%, completeness = 47.50%). Conversely, the "instruct-child" metaphor was the most effective in guiding teachers toward accurate rule composition (correctness = 83.33%, completeness = 66.67%).

6.1. INVESTIGATING LINGUISTIC CLUES IN FORM-BASED COMPOSITIONAL VISUAL INTERFACES

Table 6.9: Examples of participants' quotes where the "instruct" metaphor is observable while using concrete primitives and abstract ones.

metaphor occurrence	concrete condition	abstract condition
<i>Instruct-assignment</i>	P5 – concrete primitives: <i>"[I expect] that the LED turns red because I have already told him that red means wrong."</i>	P8 – abstract primitives: <i>"But where should I define the goal? [...] for example here [indicates the platform] it says 'placed into position 3' so the teacher's goal for the child is to take a digit tile and insert it."</i>
<i>Instruct-child</i>	P8 – concrete primitives: <i>"Then I have to leave always one [slot] empty, to allow the child to insert one or more [tiles], right? That should be the logic, one or more empty."</i>	P24 – <i>"So I place [...] the number '66' on the first two slots, and then I put the 'equal' sign on the third one... Now the kid should..."</i>

Table 6.10: Metaphors frequency across participants. *"As primary"* shows how many participants showed the metaphor as their primary reference, and *"overall"* shows how many participants displayed such metaphor, even when it did not represent their primary reference.

metaphor	as primary	participants	overall	participants
<i>Command</i>	8	P1, P4, P5, P10, P13, P14, P18, P27	21	P1, P3, P4, P5, P6, P7, P8, P10, P11, P13, P14, P15, P18, P19, P20, P21, P22, P23, P24, P26, P27
<i>React</i>	2	P17, P25	17	P1, P3, P5, P7, P8, P9, P11, P13, P15, P16, P17, P19, P20, P22, P24, P25, P26, P27
<i>Instruct-assignment</i>	2	P21, P23	10	P3, P6, P8, P9, P11, P14, P20, P21, P23, P24
<i>Instruct-child</i>	1	P9	13	P5, P7, P8, P9, P10, P11, P15, P16, P19, P23, P24, P26, P27
<i>No prevalent metaphor</i>	12	P3, P6, P7, P8, P11, P15, P16, P19, P20, P22, P24, P26		

Conceptual Mental Model of Computer (RQ4)

In addition to the metaphors, from participants verbal reports, it emerged that they sometimes recalled their conceptual mental model of computers during complex and unexpected interactions with the system. Participants considered the computer to be:

- (i) **"smart"** when they expected the system to automatically process or infer information;
- (ii) **"mindless"** when the system was expected to require clear and detailed explanations, even when unnecessary.

"Smart" conceptual mental models were observable when participants' high expectations towards the tools' automation capabilities were unmet. For example, P10 (concrete primitives) expected the LED to turn off automatically under a condition that was not defined and, thus, would never trigger (*"it doesn't turn [the LED] off anymore [...] why does it not turn off?"*). "Mindless" conceptual mental models were observed in participants who underestimated the capabilities of the system. For example, P24 (concrete primitives) did not expect the tools to have algebraic capabilities (*"Does the program know what the symbols $\times$ and $=$ mean?"*).

Learning to Interact With the System: Primary and Overlapping Metaphors (RQ4)

Thirteen (13) participants relied primarily on a specific metaphor that guided their whole interaction with the system (see Table 6.10). Although these 13 participants occasionally were observed assuming other metaphors, they tended to revert to their primary one, especially during challenging interactions with the system, such as crucial decision-making in rule composition and unexpected outcomes during testing. For the remaining participants, it was difficult to identify a dominant metaphor, since multiple metaphors often overlapped or quickly interchanged due to the dynamic nature of mental models and human-computer interactions Johnson-Laird (2010). In particular, five participants (P11, P19, P20, P7, P3) did not exhibit a prevalent metaphor in either of the two tasks.

While most of the combinations did not provide a clear advantage to participants, participant P7, who mixed "instruct-child" and "react" metaphors, succeeded in both tasks. Interestingly, seven participants (P6, P8, P15, P16, P22, P24, P26) showed learning progress and began to adopt a "react" metaphor during the first task, which was retained during the second one, as it proved to be effective in rule creation (except for a

few participants, e.g., P22, P26). Unlike participants who showed multiple overlapping metaphors, those displaying learning progress often switched metaphors, probably because the interaction with the system proved their metaphors to be ineffective.

Interacting With the System: Abstract vs. Concrete Primitives (RQ5)

Participants showed a preference for the concrete version of the primitives over the abstract one, as reflected in their performance (correctness = 57.84%, completeness = 54.00% for the concrete primitives vs. correctness = 37.62%, completeness = 38.00% for the abstract primitives). However, the simplicity of the rules created using concrete primitives often obscured the recognition of this metaphor's limitations in TAP.

Regarding the interaction between the emergent metaphors and the two types of primitives, a greater number of metaphors were employed in the first task, regardless of the primitives used (range = 1 : 9; $M = 3.08$, $SD = 2.20$), compared to the second task (range = 1 : 8; $M = 1.96$, $SD = 1.51$). This difference was particularly pronounced among participants who began with abstract primitives (range = 1 : 9; $M = 3.31$, $SD = 2.32$), as their use of metaphors was notably reduced in the second task – when using concrete primitives (range = 1 : 2; $M = 1.54$, $SD = 0.52$). In contrast, participants who started with concrete primitives (range = 1 : 9; $M = 2.83$, $SD = 2.12$) showed less variation in metaphor usage between tasks (range = 1 : 8; $M = 2.42$, $SD = 2.10$).

Notably, when beginning with concrete primitives, two participants (compared to none when starting with abstract primitives) increased the number of metaphors adopted in the second task involving abstract primitives. Participants who initially explored a variety of metaphors were better able to identify a reference metaphor for rule composition. The count of participants who adopted each metaphor per task is presented in Table 6.11.

Table 6.11: The number of participants who adopted each metaphor per task and the initial primitives used.

	ABSTRACT FIRST			CONCRETE FIRST		
	abstract only	concrete only	both	abstract only	concrete only	both
<i>Command</i>	5	1	7	1	1	6
<i>Instruct-assignment</i>	4	0	2	2	1	1
<i>Instruct-child</i>	3	1	1	1	3	4
<i>React</i>	0	1	9	0	1	6

The effectiveness of metaphors varied based on their primitive type. The "com-

mand” metaphor was found to hinder the definition of abstract rules; however, it was supported by concrete primitives, allowing participants to specify the tile content. The “instruct-child” metaphor was more commonly observed when participants began with concrete primitives, as this aligned with their understanding of “starting situations” (the teacher) and “goals” (the child). Conversely, when starting with abstract primitives, participants often shifted from the “instruct-child” metaphor to the “react” metaphor in two-thirds of the cases, indicating a learning process. Learning effects were more prevalent when beginning with abstract primitives, occurring in six out of seven cases. This suggests that starting with the most challenging condition prompted participants to abandon their spontaneous yet ineffective metaphors, helping them adopt the trigger-action logic.

Overall, performance improved slightly in the second task, with correctness at 50.20% and completeness at 50.00%, compared to the first task, which had correctness at 46.46% and completeness at 42.00%. When starting with abstract primitives, scores improved (correctness: 45.75%, completeness: 42.30%) compared to starting with concrete primitives (correctness: 62.85%, completeness: 60.71%). However, performance declined when the order was reversed, with abstract primitives yielding a correctness of 31.25% and completeness of 33.33%, and concrete primitives resulting in a correctness of 47.16% and completeness of 38.46%. Additionally, scores for abstract primitives were higher when they were introduced first (correctness: 45.75%, completeness: 42.30%) than when they were introduced second (correctness: 31.25%, completeness: 33.33%). See Figure 6.8.

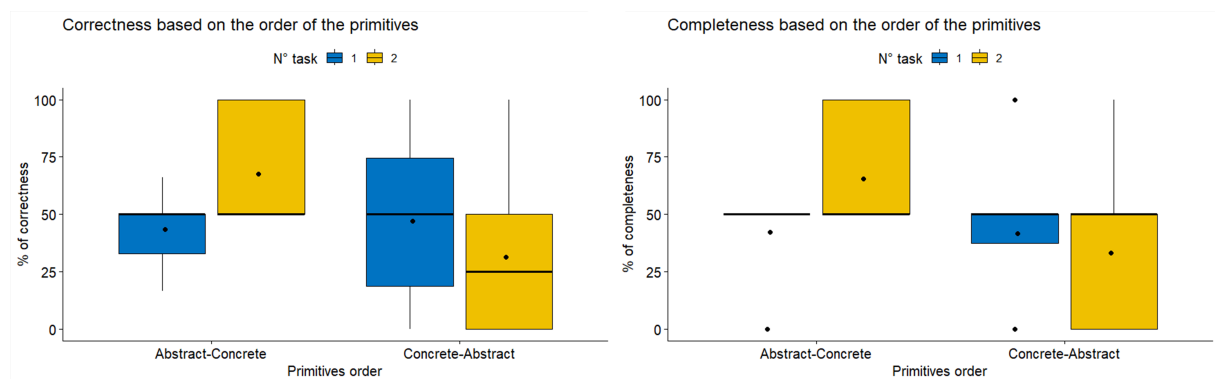


Figure 6.8: On the left, the participants’ correctness percentages are divided by conditions and task order (abstracts vs. concrete first). On the right, the participants’ completeness percentages are divided by conditions and task order (abstracts vs. concrete first).

6.1.7 Discussion

Through thematic analysis of verbal reports, three metaphors were identified, representing aspects of participants' working mental models elicited during their interaction with the system. The "command" metaphor was the most frequently observed, yet it proved the least suitable for manipulating TAP rules (**RQ4**; in line with Lukkarinen et al., 2021). It recalls the strategies we observed from the pilot study (subsection 6.1.4), specifically the *rule-by-rule* and *step-by-step* approaches, respectively for error-detecting and debugging. While this metaphor functioned adequately for simple rules in the concrete version, its effectiveness diminished in the abstract version (**RQ5**), particularly when participants first engaged with the concrete version. Conversely, the "react" and "instruct-child" metaphors demonstrated better effectiveness, aiding participants in rule generation. Although it may seem unexpected that "instruct-child" was more beneficial than "react" for rule manipulation, this finding aligns with Wiedenbeck (2005), who suggested that expertise-related or familiar class scenarios can facilitate participants in EUD tasks. Teachers, in particular, benefited from the mental simulation of the student placing the trigger tile, as it closely aligns with trigger-action logic (**RQ4**).

Beyond metaphors, the platform version (abstract-concrete) also influenced teachers' performance (**RQ5**). The abstract version was generally perceived as more complex, leading to lower correctness and completeness in trigger-action rules. Participants demonstrated higher correctness in the second task when the abstract version preceded the concrete one, whereas performance declined when the concrete version was encountered first. This suggests, similarly to what was observed in the pilot study (subsection 6.1.4), that abstract primitives allow one to reason and try multiple strategies and identify the best one. Interaction with the EUD platform contributed to participants' familiarity, resulting in improved performance in the second task, irrespective of condition (Eriksson and Jansson, 2016; Xie et al., 2019). However, this familiarity could become a drawback when strategies from the first task were inappropriately transferred to the second task despite their ineffectiveness (Eriksson and Jansson, 2016; Xie et al., 2019), as seen when teachers applied the "command" metaphor in the concrete condition and persisted with it in the abstract version. Teachers' challenges further highlighted the incongruence between users' conceptual mental models and those envisioned by designers (Norman, 1986; Norman, 1983; Norman, 2013), as difficulties with abstract primitives often stemmed from their formulation. In contrast, concrete primitives appeared to align better with participants' working mental mod-

els, consistent with what we observed in the pilot study (subsection 6.1.4) and with Johnson-Laird (1983, 2010), who described mental models as *iconic* and represented through concrete instances (Johnson-Laird et al., 1972).

A third factor influencing teachers' interaction with the system was their conceptual mental model of the system itself (Fischer, 1991; Fischer, 2001), particularly when participants encountered uncertainty or the system behavior deviated from expectations. As noted by Fischer (2017), users learning to master a tool often focus on understanding the system rather than the task. The fact that most participants referred to a conceptual model of a "smart" or "mindless" computer suggests a stronger concern with the tool rather than the task. Furthermore, consistent with Fischer (2001) and Fischer (2017) and the concept of the "invisible computer", participants referenced their conceptual mental model less frequently in the concrete version, indicating that this version was easier to use and more aligned with teachers' conceptual world.

During system interaction, the evolving nature of working mental models became evident (**RQ4**), aligning with Norman's observations regarding the "messy" nature of mental models Norman (1983). As participants became more familiar with the system, they often transitioned from less effective to more effective metaphors. However, when faced with difficulties, such as unexpected outcomes during testing, they tended to revert to their initial metaphor. This underscores the importance of the platform in guiding participants in rule composition, which is essential for designing EUD solutions that accommodate teachers' metaphors while facilitating the adoption of trigger-action logic.

Although metaphors are challenging to generalize due to their grounding in professional practices, the mistakes and misconceptions observed among participants closely resemble those identified by Brackenbury et al. (2019). Teachers struggled with understanding "concurrency" in trigger-action rules, frequently leading to *timing errors* (Brackenbury et al., 2019; Meerbaum-Salant et al., 2011). They also faced difficulties distinguishing between states and events, resulting in what Brackenbury et al. (2019) termed *flipped triggers* – where users misidentified which element of a trigger should represent a state and which should represent an event. Additionally, participants exhibited what the authors described as *inaccurate user expectations*, a phenomenon commonly observed among novice programmers (Bonar and Soloway, 1983; Bonar and Soloway, 1985; Pea, 1986). Some teachers expected the system to possess greater intelligence than it actually had ("smart computer" as in Brackenbury et al., 2019), while others underestimated its capabilities ("mindless computer"). For example, partici-

pants often assumed that turning off the LED would be handled automatically by the system – failing to recognize the necessity of creating a rule – that aligns with the *missing reversal* and *secure-default bias* bugs (Brackenbury et al., 2019; Yarosh and Zave, 2017).

Limitations

The study presents several limitations. It was conducted with a small sample of participants who shared similar backgrounds and experiences with digital educational devices. To ensure the generalizability of the findings, future studies should explore different tasks and include participants with more diverse backgrounds. Additionally, as this study was exploratory, the insights obtained require validation through further research.

Moreover, the specific implementations of the abstract and concrete primitives may have influenced teachers’ understanding of the task, potentially shaping their working mental model assumptions during the interaction. Lastly, the absence of training before the study may have contributed to lower performance results. However, this decision was intentional, as no training (apart from a brief introductory video) was provided to focus on pre-conceptual working mental models during teachers’ initial interaction with the system.

6.2 Exploring How Large Language Model-Based Systems Addresses Ambiguities in Language

Advances in NLP and the increasing capabilities of AI-assisted code generation have gained significant attention, offering promising opportunities to support both expert programmers and non-programmers in writing code (Ouyang et al., 2023). In particular, LLMs, such as ChatGPT-4 (OpenAI) and other specialized models, can generate executable code from natural language prompts (Calò and De Russis, 2024; Liu et al., 2024; Yetiştirilen et al., 2023; Zhong and Wang, 2023).

The integration of LLMs into EUD TAP-based systems (e.g., Gallo et al., 2021; Gallo et al., 2024; Monge Roffarello and De Russis, 2023; Perera et al., 2015) has the potential to address limitations associated with fixed rule structures, which may not always align with users’ mental models, lead to working memory overload, and restrict the expressiveness of automation programs created by non-programmers (Blackwell, 2002). As previously observed (Fogli and Tetteroo, 2022; Tetteroo, 2023), enabling users to

define automation rules using natural language – through chatbot or conversational-based systems – can enhance their expressiveness powers, allowing them to define their needs more freely. This approach also eliminates the necessity of learning new formalisms or acquiring technical programming skills.

However, non-expert programmers interacting with LLMs may encounter challenges due to natural language ambiguities, as reported in previous studies (Monge Roffarello and De Russis, 2023; Scoccia, 2023). These ambiguities can affect how LLMs interpret and execute automation rules, potentially leading to unintended behaviors (Breve et al., 2023). A key challenge in LLM-based natural language parsing lies in managing ambiguities or under-specified instructions, which can lead to misinterpretation of user intent and unintended outcomes (Petrick, 1976).

Recent studies have demonstrated ChatGPT’s ability to translate non-programmer prompts into trigger-action rules (e.g., Monge Roffarello and De Russis, 2023). However, to our knowledge, research on how LLMs detect and address language ambiguity remains unexplored.

In collaboration with the *Polytechnic University of Milan*, we examined how **LLM-based systems** interpret and process natural language trigger-action rules, supporting negotiation and feedback mechanisms to explore the potential and challenges of this approach. We conducted an initial study analyzing how LLMs handle natural language ambiguities in rules created by end users for home automation scenarios (see subsection 6.2.3). This was followed by a similar study focusing on rules generated by individuals with intellectual disabilities (see subsection 6.2.4). As detailed in the following subsection 6.2.1, we decided to focus on this specific use case as:

- Home automation systems are gaining increasing attention in the context of accessibility and co-housing projects (Bacchin et al., 2021);
- Language-based communication can be particularly challenging for some individuals with intellectual disabilities (APA, 2013; Coppens-Hofman et al., 2017).

6.2.1 Context: Home Automation Systems and Accessibility

Advancements in IoT-based technologies have led to increased interest in home automation’s role in enhancing the autonomy and quality of life for individuals with disabilities (Bacchin et al., 2021; Bridge et al., 2021; Gervais et al., 2023; Nauha et al., 2018)⁵. Home automation systems provide concrete assistance in performing daily

⁵For a review on the topic see Ulloa. et al. (2021).

activities more independently, engaging more actively with their surroundings, and contributing to personal safety (Ahmed et al., 2016; Cleland et al., 2024a; Dewsbury et al., 2003; Dobkin and Dorsch, 2017; Gentry, 2009; Semary et al., 2024). Additionally, these systems can support caregivers by enabling remote monitoring, optimizing intervention management, and supporting controlled transitions toward independent living (Cleland et al., 2024b).

Intellectual disability is a neurodevelopmental disorder characterized by significant limitations in intellectual functioning and adaptive behavior (APA, 2013). Individuals with intellectual disability often experience challenges in their daily lives, including difficulties in developing autonomy and managing cognitive tasks such as problem-solving, reasoning, and decision-making. Moreover, many individuals with intellectual disability experience physical and developmental challenges, including limited fine motor skills, hyperactivity, attention deficits, and impairments in both verbal and non-verbal communication (APA, 2013). According to the *Diagnostic and Statistical Manual of Mental Disorders – 5th Edition* (DSM-5; APA, 2013), intellectual disability frequently occurs in comorbidity with other conditions, including neurodevelopmental disorders (e.g., Attention-Deficit Hyperactivity Disorder, Autism Spectrum Disorder), genetic syndromes (e.g., Down Syndrome, Fragile X Syndrome), and medical conditions (e.g., Epilepsy, Fetal Alcohol Syndrome). Consequently, intellectual disability includes a wide and highly variable sample with specific individual needs according to the severity of the intellectual disability, the level of autonomy and adaptive behavior, and the presence of other symptoms, conditions, or disorders.

According to Nauha et al. (2018) and Gervais et al. (2023), assistive technologies⁶ integrated into home automation systems can improve the quality of life of people with intellectual disability, improve personal autonomy, and support caregivers' work. This is particularly important as 107.62 million individuals live with intellectual disability (1.74% of the global population), with associated inequalities continuing to increase, especially among lower sociodemographic index areas, as reported by Nair et al. (2022).

For example, Lussier-Desrochers et al. (2012) assessed how home automation can assist individuals with intellectual disability in meal preparation. The study involved 12 participants who prepared recipes both with and without the aid of technology. The results indicated that the use of home automation significantly reduced the amount of caregiver support needed, particularly in terms of verbal and gestural prompting,

⁶Assistive technologies refer to devices, programs, or systems designed and utilized to minimize the restrictions imposed on individuals due to their disabilities (e.g., Nauha et al., 2018).

while not impacting the time required to complete the tasks.

IoT-based home systems are trying to provide diverse support for individuals with intellectual disability, such as enabling control of living spaces via voice commands or adaptive interfaces (Al-Karawi and Mohammed, 2021; Almusaed et al., 2023). Despite the potential of using technologies to support individuals with intellectual disabilities and their caregivers, there are still several challenges to face. For example, mainstream Intelligent Personal Assistants (IPAs), such as *Amazon Echo* and *Google Home*, enhance user experience and agency but, as observed by Smith et al. (2023), issues remain regarding speech recognition accuracy and well-being impact at the group level. Moreover, the lack of involvement of individuals with intellectual disabilities in technology design has resulted in insufficient recognition of accessibility needs (Moyà-Köhler and Domènech, 2022).

However, recent advancements in low-cost sensors and *Do-It-Yourself* approaches have shifted the focus toward user-centered design involving individuals with intellectual disabilities (Meissner et al., 2017; Rajapakse et al., 2015; Sas and Neustaedter, 2017). Studies show that involving individuals with intellectual disabilities in technology development enhances system usability while fostering skill-building, self-confidence, and agency (Robb et al., 2021; Safari et al., 2021; Safari et al., 2023). Nowadays, accessible electronic toolkits enable individuals with disabilities to create and personalize IoT solutions (e.g., Gennari et al., 2024; Morra et al., 2024).

Integrating LLMs into EUD systems with interpretation and code-generation capabilities is a promising approach (Calò and De Russis, 2024; Tetteroo, 2023), assisting users in overcoming the challenges of using EUD form-based compositional visual interfaces. For individuals with intellectual disabilities, these advancements could enhance freedom of expression and alleviate the cognitive demands of adapting to a “fixed” EUD system.

To better support individuals with intellectual disabilities – considering the wide and variable range of language-related skills of people in this sample (APA, 2013; Coppens-Hofman et al., 2017) – recent research has explored conversation-based strategies and multimodal approaches integrating LLMs (e.g., Calò and De Russis, 2024; King et al., 2024).

Some studies proposed conversational-based approaches to improve input disambiguation of LLM-based systems (King et al., 2024; Rivkin et al., 2023), including follow-up clarifications, rephrasing, suggestive questions (Keyvan and Huang, 2022), and chain-of-thought prompting (Wei et al., 2022). Calò and De Russis (2024) presented

a multimodal approach that integrates textual and visual cues that successfully enhanced the alignment between user intent and system responses (Calò and De Russis, 2024). These approaches enabled users to articulate their intents through a guided path of input disambiguation (King et al., 2024; Rivkin et al., 2023), including follow-up clarifications, rephrasing, suggestive questions (Keyvan and Huang, 2022), and chain-of-thought prompting (Wei et al., 2022). These findings suggest that iterative interactions clarify ambiguities more reliably than direct code generation (Lainwright and Pember-ton, 2024).

Despite the significant potential of LLMs in improving accessibility and supporting to some extent EUD tasks in home automation systems, further research is needed to investigate how LLMs address language ambiguities and interpret users' intent, as well as explore how these systems can support individuals with intellectual disabilities in EUD tasks.

6.2.2 Research Questions

We addressed two research questions:

- **RQ6.** Which language ambiguities are identified and addressed by an LLM while interpreting trigger-action rules written in natural language?
- **RQ7.** Which disambiguation questions can an LLM system generate to align with user intent?

6.2.3 Study 6: Exploring How LLMs Handle Linguistic Ambiguities

For answering **RQ6**, we conducted an initial investigation to evaluate *ChatGPT-4*'s⁷ ability to interpret end-user instructions for defining automation rules in smart home applications and its capability to translate these instructions into deployable Python code. The study described in this section corresponds to the one presented in Andrao et al. (2024b). The study was approved by the ethics committee of the *University of Trento* and was conducted between November 2023 and January 2024.

Our study had two main objectives:

1. Analyzing how ChatGPT-4 translates and interprets ambiguous home automation rules into deployable Python code.

⁷The latest available version in November 2023.

2. Examining how ChatGPT-4 identifies and addresses ambiguities that may introduce security and safety risks.

To achieve these objectives, we used a subset of the instructions collected in the pilot study described in subsection 5.3.5.

Participants

Participants were the 16 described in subsection 5.3.5 ($N_f = 8$, $N_m = 8$; age range = 24 – 60, $M = 32.81$, $SD = 11.44$ years).

Materials

We selected a subset of instructions defined by participants in the pilot study (see subsection 5.3.5). In this study, each participant was presented with 12 scenarios and asked to write instructions for implementing the corresponding automation. Two independent researchers evaluated each instruction and categorized them as follows:

- *Non-ambiguous*, when the instruction provided clear references to conditions and actions; *ambiguous*, when the instruction's structure, terminology, or details were unclear, potentially leading to multiple interpretations of the outcome.
- *Incomplete*, when one or multiple elements (state, event, action) depicted in the video were left implicit; *complete*, when the instruction explicitly included all elements.

Of the 199 instructions collected, only the 34 that both researchers classified as *ambiguous* and *complete* were included in this study.

Procedure

For each rule, we provided ChatGPT-4 with specific instructions, a list of sensors and smart devices involved in the scenario (named *context* in the prompt, similarly to Ghiani et al., 2017), and we questioned it by asking:

- To generate Python code ready for deployment. The prompt was:

"Given the following rule and the following context, make a Python code ready to be deployed. Just output the code without any further comments".

- To identify any errors or ambiguities in the rule written by the user. The prompt was:

"Given the following rule and the context, identify if there are possible ambiguities or errors in the way the instructions were written".

See Figure 6.9 for a visual representation of the procedure.

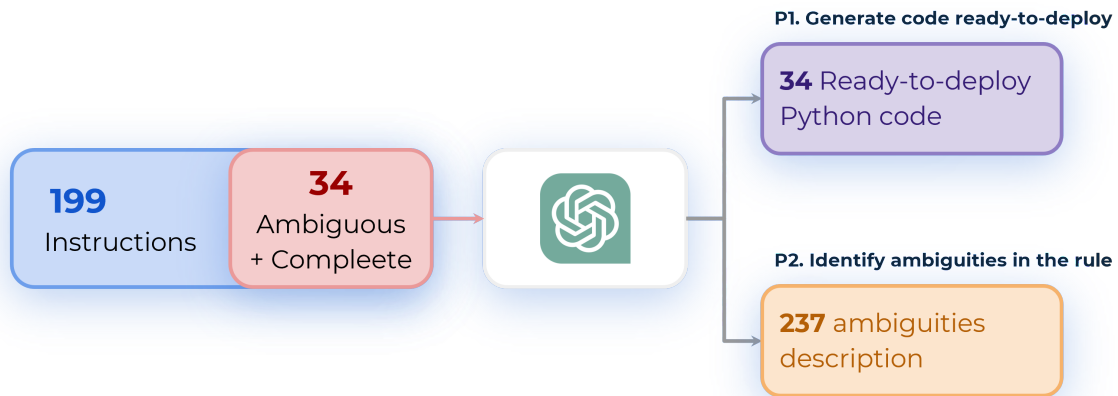


Figure 6.9: The image provides a graphical representation of the process, illustrating the transition from the 199 instructions collected in the pilot study (subsection 5.3.5) to the outputs generated by ChatGPT-4 using the two prompts.

Results

The 34 Python code snippets generated by ChatGPT-4 were evaluated for correctness by two independent researchers, resulting in an accuracy rate of 94.00%. Only two of the generated codes were found to be incorrect. The first lacked a condition ("*as during the day*") explicitly specified in the user instruction ("*If at night the temperature remains, **as during the day**, below 10°C, turn on the heating*"). The second failed to account for an adverb ("*quickly*") indicating the necessity of a time interval for the rule to execute correctly ("*When the temperature **quickly rises** to 26 degrees and smoke comes out, the water spray is activated at the same time*").

When asked to identify ambiguities in the 34 natural language instructions, ChatGPT-4 detected a total of 237 ambiguities, with a minimum of 3 and a maximum of 10 per instruction ($M = 6.97$, $SD = 1.45$). Two researchers independently coded the ambiguity descriptions and then analyzed whether and how ChatGPT-4 had autonomously

addressed these ambiguities in the corresponding ready-to-deploy Python code snippets. Four main themes emerged from the ambiguities' descriptions.

Theme 1: Ambiguities related to the outcome of the rule and its variables (n = 83). Eleven (11) instances involved uncertainties regarding the end of the action, for example, one ambiguity related to the rule "At night and when a presence is detected around the Dog's kennel, **the light must be turned on**" was:

- **End of Action:** *"The rule does not state **how long the light should remain on** after being triggered. Should it stay on for a set period, until no more movement is detected, or until daylight is sufficient according to the light sensor?"*.

Nine (9) ambiguities were related to temporal aspects as the rule did not clearly state the sequence/simultaneity of occurrences to trigger the action, and 2 involved suggestions for additional sensors or devices that, if integrated, can ensure the needed outcome. For example:

- **Temporal Aspects** [Rule: "the house must water the plants **only if the sun is present and a temperature of 25 degrees Celsius is reached**"]: *"The rule states that both conditions (sun presence and temperature) must be met, but it does not specify if they need to occur simultaneously"*.
- **Additional Devices** [Rule: "make sure that **my dog can enter the house** at night through the small door in the courtyard."]: *"The rule assumes the system can identify the dog. If there are other animals in the area, the system needs a way to ensure that only the dog can use the door (e.g., **using a collar with an RFID chip**)"*.

Other 39 involved possible ambiguities in error handling and, specifically, in handling possible or imaginary conflicts, false negatives or positive triggers, or possible safety hazards. Finally, 22 described possible undesirable outcomes (e.g., automatically turning on a fireplace based on temperature and motion) associated with potentially harmful outcomes for safety, security risks, damages, and high/unusual energy consumption. For example:

- **Error handling** [Rule: "If it is night and the dog is outside the house, you must **open the dog's door** to allow it to enter the house and then close it once it has entered"]: *"The rule does not account for what should happen if the door fails to open or close."*

- **Undesirable outcomes** [Rule: "If the television is turned on during the day, the house must **close the curtains** to prevent light from entering."]: *"Energy efficiency – The rule could lead to unnecessary use of the motorized curtains if the TV is turned on and off frequently throughout the day"*.

Looking at the Python code snippets, 13 of these ambiguities were fully resolved by ChatGPT-4, 4 were partially solved, and 66 were not addressed in the generated code (Figure 6.10).

Theme 2: Ambiguities related to the language (n = 55). Twenty-one (21) ambiguities involved word usage in formulating instructions with some words being overly specific, and others being too generic or unconventional. For instance:

- **Word Precision** [Rule: "In case of fire, activate the **evacuation plan**. If you detect movements inside, automatically open the door"]: *"The rule mentions activating an evacuation plan but does **not provide details** on what the plan entails."*

Twenty-four (24) were related to the use of generic expressions, employing words instead of defining specific values:

- **Figurative Value** [Rule: "If the television is turned on **during the day**, the house must close the curtains to prevent light from entering"]: *"The rule does not specify what constitutes 'day'. Is it based on **specific hours**?"*.

In only 1 instance, the ambiguity was related to an ambiguous typo in a rule. In 3 cases, the ambiguity was related to the use of conjunctions, particularly the expression "and/or". In 3, the ambiguity was related to a rule composed of multiple sentences while, in other 3, it was related to the language used in general. Some examples of these ambiguities are listed below:

- **Conjunctions** [Rule: "If an increase in internal temperature **and/or** smoke is detected, turn on the fire safety system"]: *"The use of **and/or** could lead to confusion about whether the system should activate with just smoke, just an increase in temperature, or if both conditions must be present"*.
- **Sentence division** [Rule: "**Day mode:** if the TV is on, keep the curtains closed. **Night mode:** if the TV is on, open the curtains; with a temperature above 22°C and no rain, also open the window."]: *"The rule does not specify whether this condition is only applicable **at night or both during the day and at night**. This could lead to confusion about when to apply this condition"*.

- **Language used** [Rule: "After a certain time (around 20-21), if the temperature is below 10 degrees, light the fireplace"]: *"If the smart home system's programming interface is in English or another language, the rule should be translated and formatted according to the system's requirements"*.

In this group, 31 ambiguities were fully resolved in the code, 3 were only partially resolved, and 21 had not been addressed (see Figure 6.10).

Theme 3: Ambiguities related to the necessity of content extension (n = 55). This included: 30 cases of suggestions on specifying alternative states or conditions to avoid ambiguities; 9 cases of determining when/how often to monitor the state; 6 cases for providing details of the action, and 10 cases concerning the need of confirmation/alert feedback. Some examples:

- **Specifications on States** [Rule: "If the temperature is below 10 degrees Celsius and there is a family member in the house, then **turn on the fireplace**"]: *"The rule doesn't specify what to do if the fireplace is already on"*.
- **Monitoring State** [Rule: "Home Automation System, close the curtains when it is **daytime** and I am **watching TV**"]: *"The rule does not specify whether the system should continuously monitor for the conditions or if it should only check at certain intervals"*.
- **Details on Action** [Rule: "If the television is turned on during the day, the house must **close the curtains** to prevent light from entering"]: *"The rule does not specify how much the curtains should close"*.
- **Feedback to Users** [Rule: "**close the window** when it rains and electronic devices are on"]: *"The rule does not include any provisions for alerting the occupants of the home that the window will be closed. This could be important for awareness and safety"*.

Of these ambiguities, 14 were fully resolved in the generated code, 4 partially, while 37 were not addressed (see Figure 6.10).

Theme 4: Ambiguities related to the system's functioning (n = 44). Among this category, 27 were related to suggestions/comments on sensor localization and integration within a broader system; 17 accuracy/ sensitivities of sensors that can lead to false positive or negative triggers, as:

- **Sensor list** [Rule: "if at night the **temperature** remains, as during the day, below 10°C, turn on the heating"]: *"The rule does not specify where the temperature should*

be measured. If the temperature sensor is in the living room, it may not accurately reflect the temperature in other parts of the house”.

- **False Positive or Negative** [Rule: “In case of fire/open flames/smoke, if someone is **present** in the room, unlock and open the door”]: “It mentions a movement sensor, but movement sensors can sometimes give *false positives false positives (e.g., movement of pets) or false negatives (e.g., if a person is immobile)*”.

Of these instances, 19 ambiguities were fully resolved, 1 was partially resolved, and 24 were not addressed.

ChatGPT-4 fully addresses 66 (27.84%) of the ambiguities, partially resolved 12 (5.10%), while 159 (67.10%) were not addressed (see Fig 6.10).

Overall the analysis reveals ChatGPT-4’s proficiency in addressing ambiguities related to the second theme *language*, successfully addressing 31 out of 55 cases. This is frequently attributed to its skill in interpreting incorrect verbal aspects or unconventional temporal and grammatical structures. However, ChatGPT-4 shows less expertise in dealing with ambiguities associated with the first theme *outcome*, as it replicated the same ambiguities found in the prompt in approximately 66 out of 80 cases. Notably, ChatGPT-4 tends to overlook two specific categories: error handling and ambiguities that might lead to unintended effects. Additionally, there were instances in each theme where ChatGPT-4 only *partially resolved ambiguities*. For example, in a case involving feedback to house occupants, the code included code debugging-level warning messages but not direct messages to the occupants. Another case involved the figurative interpretation of values, like defining “darkness” in a sentence. ChatGPT-4 identified the ambiguity but inadequately addressed it in the generated code. The code incorporated a logic to read data from the light sensor as a trigger, but it did not give the user a structure to set a threshold for the sensor data, offering only boolean true/false options for action triggering.

Discussion

The results of this first study provide evidence of the potential of LLMs, such as ChatGPT-4, in interpreting ambiguous natural language instructions and translating them into functional code. Despite the ambiguities in the rules formulation, as well as the simplicity of the prompts and the provided context, ChatGPT-4 was able to generate syntactically accurate and complete Python code in 94.00% of cases. Moreover, ChatGPT-4

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

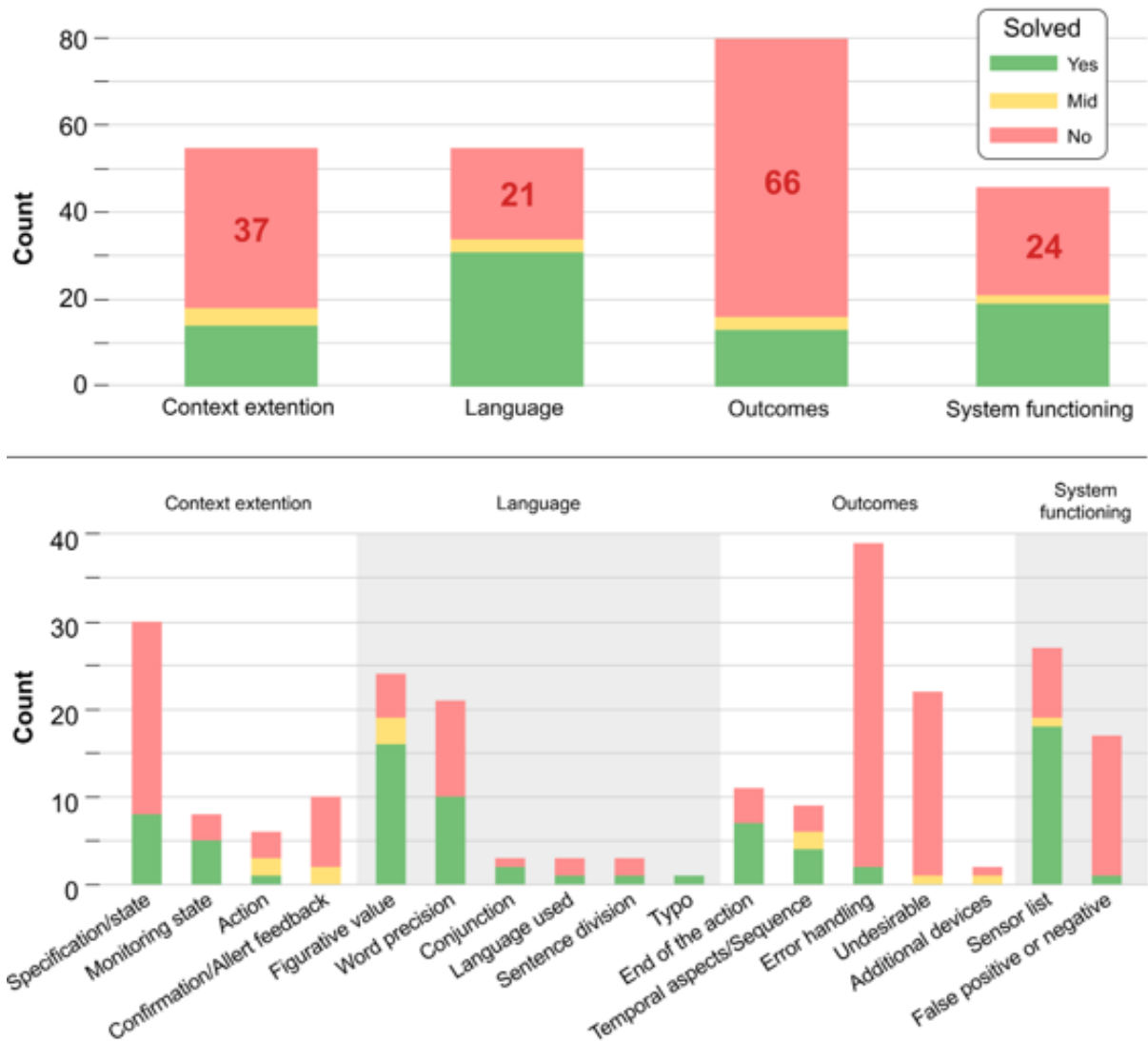


Figure 6.10: Distribution of solved ambiguities by theme (above) and identification code (below).

was capable of detecting many ambiguities in users' expressions. However, a significant portion of ambiguities (67.10%) remained unresolved in the generated code, potentially impacting safety and security aspects (e.g., Breve et al., 2023).

On the other hand, a key concern is that ChatGPT-4's handling of ambiguities – while capable of producing functional code – relies on its own assumptions rather than the explicit intent of the user. We observed that ChatGPT-4 arbitrarily selects how to address these ambiguities without explicit user input, raising an important debate about whether its approach aligns with users' actual intentions.

This issue is particularly critical when users are non-programmers, as they may struggle to interpret Python code and identify potential issues. It becomes even more concerning in cases where users have cognitive impairments, as their ability to detect and correct unintended output in aa code could be further reduced.

From our findings, we argue that effective interaction for non- programmers should not rely solely on direct code generation. Instead, a TAP LLM-based system should first focus on identifying ambiguities, followed by an iterative resolution process involving negotiation with the user to ensure the accurate generation of code (King et al., 2024; Rivkin et al., 2023).

Limitations. We acknowledge some limitations in our study, starting from the limited number of participants that can affect the generalization of the emerged results. Additionally, ChatGPT-4’s ability to address ambiguities could be mitigated by future model releases, emphasizing the importance of continuing research in this area. Nevertheless, we believe that this first study on automatically generated code from end users’ natural language instructions may help in future research, such as on the design of effective interaction paradigms that facilitate the negotiation process to mitigate ambiguities. This will require an extended research-through-design approach that addresses the multiple aspects emerging from the study.

6.2.4 Study 7: Evaluating LLMs’ Ability to Interpret Ambiguous Instructions from Individuals with Intellectual Disabilities

The second study evaluated the performance of ChatGPT-4o⁸ in interpreting intents associated with home automation instructions written in natural language by individuals with intellectual disabilities, as well as its ability to translate these instructions into pseudo-code⁹. The study described in this section refers to the one reported in Morra et al. (In submission) and was conducted between June and July 2024 following approval from the ethics committee of *Polytechnic University of Milan*.

Since individuals with intellectual disabilities may face challenges in language production and expressive communication (more details in subsection 6.2.1), they might encounter difficulties in clearly conveying their needs and intentions through natural language. Such challenges could, in turn, introduce higher walls for LLMs – and in this case for ChatGPT – in accurately extracting individuals’ intended meaning.

⁸The most recent version available in June and July 2024.

⁹We decided to query for pseudo-code generation (vs. Python code or other specific programming languages) to allow more researchers in our team to understand and analyze the generated code.

By examining the performance and limitations of LLM in automatically identifying user intent and generating corresponding pseudo-codes, we aimed to define some guidelines for designing effective EUD tools that empower individuals with intellectual disabilities in adapting smart home environments to their needs. The study focused on **RQ6** and **RQ7** by investigating:

- The correctness of the generated pseudo-code, both formally and practically, when instructions provided by people with intellectual disabilities are ambiguous or incomplete;
- ChatGPT's ability in interpreting user intentions when given ambiguous instructions by people with intellectual disabilities;
- ChatGPT ability to recognize ambiguities in user input and generate clarification questions in line with the detected ambiguities.

The study was structured in two distinct phases:

1. **The User Study: *Collection of participants instructions*.** As a first step, we collected natural language instructions to achieve home automation scenarios from participants with intellectual disabilities.
2. **The Study with ChatGPT: *Analysis of ChatGPT performance*.** A selection of the gathered automation instructions was employed to prompt ChatGPT, similar to the methodology used in Study 6.

Participants

The study included 14 participants aged between 23 and 56 years ($M = 35.86$, $SD = 9.67$), including six females ($M_f = 38.67$, $SD_f = 9.67$ years) and eight males ($M_m = 33.75$, $SD_m = 9.75$ years). All participants had a diagnosed intellectual disability and were assisted by the same social cooperative, *Fraternità e Amicizia*, located in Milan, northern Italy (see Table 6.12).

Among the participants, three had a diagnosis of borderline intellectual functioning, with an Intelligence Quotient (IQ) ranging from 70 to 85, with difficulties in adaptive functioning across conceptual, social, and practical domains. Two participants had a diagnosis of mild intellectual disability, with an IQ between 51 and 70 and slight difficulties in adaptive functioning. The remaining nine participants were diagnosed with moderate intellectual disability, characterized by an IQ between 35 and 51 and

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.12: The table reports data on participants’ demographics, their diagnosis, speaking and typing skills, and familiarity with home automation systems.

id	age	gen.	diagnosis	speaking & typing ability	smart-home
P1	32	F	Mild Intellectual Disability	Struggles with writing on paper but types easily on digital keyboards. Speaks fluently with complex sentences and no phonological issues	Rarely
P2	28	F	Moderate Intellectual Disability, Obsessive-Compulsive Disorder, Behavioral Disorders	No difficulties in speaking or typing	Often
P3	38	F	Moderate Intellectual Disability	No difficulties in speaking or typing	Sometimes
P4	41	F	Moderate Intellectual Disability	No difficulties in speaking or typing	Never
P5	38	F	Moderate Intellectual Disability	Simple but effective language, initiates conversations	Sometimes
P6	56	F	Borderline Intellectual Functioning	No difficulties in speaking or typing	Never
P7	41	M	Mild Intellectual Disability	Texts and speaks effectively. Works as a receptionist, interacts with strangers fluently	Often
P8	24	M	Moderate Intellectual Disability, Drug-Resistant Epilepsy, Congenital Encephalopathy	No difficulties in speaking or typing	Sometimes
P9	53	M	Moderate Intellectual Disability, Developmental Coordination Disorder, Encephalopathy	Articulates complex sentences but has phonological difficulties that impact comprehension	Never
P10	27	M	Moderate Intellectual Disability, Psychotic Symptomatology, Emotional Regulation Difficulties in Childhood, Epilepsy	No difficulties in speaking or typing	Sometimes
P11	28	M	Kabuki Syndrome	Good conversational skills but phonological articulatory issues may interfere with understanding	Never
P12	34	M	Moderate Intellectual Disability, Spastic Tetraparesis, Hyperkinesia in the Upper Limbs, Post-Anoxic Encephalopathy, Sensorineural Deafness	Typing and speaking are slow but accurate. Severe phonological articulatory difficulties hinder comprehension	Sometimes
P13	29	M	Borderline Intellectual Functioning, Autism Spectrum Disorder Level 1, Unspecified Chromosomal Condition	No difficulties in speaking or typing	Sometimes
P14	37	M	Moderate Intellectual Disability, Behavioral Disorders, Epilepsy	No typing issues. Minor phonological articulatory problems, no impact comprehension	Never

more pronounced difficulties in adaptive functioning. Regarding prior experiences with smart-home technologies, nine participants reported having used voice assistants (1 = “every day”, 6 = “sometimes”, and 2 = “rarely”). Among them, three owned a smart-home voice assistant, though only two had customized their functionalities (one “every day” and one “sometimes”).

Phase 1 - The User Study

In the first phase, we involved the participants for collecting home automation instructions expressed in natural language. We asked participants to watch eight videos depicting home automation scenarios (described in the paragraph **Materials**) and we asked them to provide the instructions to achieve that scenario (adapting the methodology used in Study 3, see section 5.3). Then, instructions were analyzed by three re-



Figure 6.11: Example frames extracted for the *s1* scenario “a person enters the bathroom during the night and the light turns on”. The top-left frame (1) represents the event “a person enters” while the state “during the night” is false (i.e., “during the day”); the top-right frame (2) represents that the action is not performed (i.e., nothing changes); the bottom-left frame (3) represents the event “a person enters” while the state “during the night” is true; the bottom-right frame (4) represents the execution of the action “the light turns on”.

searchers who classified them according to specific parameters (see **Instruction Analysis** paragraph). The instructions classified as “rules” were then evaluated by three external judges to understand the difficulty of interpreting the rule intent (see paragraph **Rules Intent Analysis**).

Materials. We designed and employed eight short, silent videos – each lasting between 13 and 16 seconds ($M = 15.00$, $SD = 1.41$ s) – depicting various home-automation scenarios (see Figure 6.11 and Table 6.13).

To create these eight videos, we started with the 12 videos developed and used in Study 3 (section 5.3) and discussed them with expert educators from the social cooperative *Fraternità e Amicizia*. The educators recommended maintaining the illustrative visual style for the video scenarios, emphasizing the importance of using simple, clear illustrations while avoiding abstraction. This visual style ensures that the scenarios

remain engaging for participants, facilitating comprehension without overwhelming details and fostering engagement and accessibility. As a result of these discussions with experienced educators, the following decisions were made:

- The number of videos was reduced, and educators agreed that eight videos were sufficient for the study.
- From the initial set of 12, four videos were included in the new pool. Specifically, we selected two scenarios focusing on *Personal Safety* (s8, s11 of Study 3) and two on *Temperature Management* (s5, s7).
- We created four new videos on themes closer to individuals with intellectual disabilities' needs: *Bathroom Management* and *Home Security*. These themes were observed by previous research conducted by one of the authors (Morra et al., 2024) on accessibility and home automation systems and confirmed by expert educators.
- In certain cases, it was considered important to represent a human performing the action instead of a dog, as this would aid participant identification with the scenarios, helping them stay focused and understand the context more effectively. To avoid introducing gender or cultural biases, a human character was designed without gender-specific features. The dog character was retained in one scenario involving fire-related danger as we did not want to have the risk of any emotional impact on participants.

Identical to the videos used in Study 3, each of the eight videos illustrated a state-event-action sequence to demonstrate the functioning of interconnected smart home devices. The first part of each video depicted a scenario where the initial state was false and no action was triggered, followed by the part in which the state was true and the action performed. The eight home automation scenarios – listed in Table 6.13 – were equally distributed across four themes:

1. ***Bathroom Management*** – Previous research (Morra et al., 2024) highlighted the relevance of this theme for individuals with intellectual disabilities. An example scenario within this theme is: "A person is inside the bathroom. When the door handle is lowered, a red light turns on". This scenario was designed considering shared living environments where informing others about bathroom occupancy is important.

Table 6.13: The scenarios illustrated in the videos, with the respective themes.

id	state	event	action	theme
s1	night	person enters	light on	<i>bathroom</i>
s2	person inside	handle is lowered	red light on	<i>bathroom</i>
s3	during the night	door opens	red light on	<i>home security</i>
s4	during the night	window opens	red light on	<i>home security</i>
s5	it is cold	person enters	fireplace on	<i>temperature</i>
s6	it is cold	it becomes night	fireplace on	<i>temperature</i>
s7	dog inside	smoke	door opens	<i>personal safety</i>
s8	smoke	temperature rises	sprinklers on	<i>personal safety</i>

2. **Home Security** – Participants in the same study by Morra et al. (2024) emphasized the importance of home security. An example scenario is: “A person opens a window at night, and a red light turns on”. In this case, the red light serves to alert residents to a potential intruder.
3. **Personal Safety** – Two videos (originally s8 and s11) from Study 3 were selected in consultation with expert educators. These videos depicted fire-related dangerous situations. In this theme, the dog character was maintained.
4. **Temperature Management** – Other two videos (originally s5 and s7) from Study 3 were selected with expert educators. These scenarios depicted situations requiring heating adjustments in the home. Unlike the fire-related scenarios, the dog character was replaced with a human figure, considering the non-threatening nature of these situations.

As in Study 3, an alternative mirrored version of each video was created to mitigate potential biases associated with spatial positioning. In these mirrored versions, elements appearing on the left side of the original video were displayed on the right. Each participant was shown half of the videos in the original version and half in the mirrored version to control directional/spatial bias.

Task and Procedure. To provide a familiar environment for participants, the study was conducted within the daycare center of *Fraternità e Amicizia*, which they regularly attended. The sessions were conducted in a quiet room equipped with a laptop placed on a table with three chairs in front of it, where the participant, an educator, and a researcher were seated. The educator’s role was to ensure that instructions were ac-

cessible and comprehensible based on the participant’s knowledge, intervening only when necessary – such as when participants encountered difficulties in understanding the task or formulating specific phrases or concepts clearly. Before the study, the legal guardians of all participants provided informed consent, and assent was also obtained from each participant. Data was collected through the online platform *Qualtrics*. At the beginning of each session, the researcher introduced the study, providing a general overview of the activity while avoiding details about the specific tasks. The procedure and task were adapted from Study 3 while maintaining the core methodology: each participant watched the videos individually and verbalized the corresponding instructions to achieve the scenarios depicted. The instructions provided at the beginning of the study were as follows:

“This study aims to investigate how people express instructions to manage smart devices. For each video, you have to express the instructions to achieve the scenario depicted as if you were communicating them to a smart home system. The smart home system understands natural language but has no prior knowledge or experience with the scenario you will describe to it”.

Participants were given the choice to either read the instructions independently or have them read aloud by the researcher or the educator. They were informed that no response was considered right or wrong and that they should provide spontaneous answers. Additionally, there was no time constraint, allowing participants to take as long as necessary. They were also permitted to replay each video as many times as needed.

Once participants confirmed their understanding of the instructions and the task’s objective, they were required to press a start button to begin. Upon initiation, a video and an empty text field appeared on the screen. Participants could press the “play” button at their convenience to watch the silent video, which played from start to finish without the option to pause. After the video ended, they could either replay it or start typing their response in the text field (see Figure 6.12).

During the task, participants could choose to type their responses or dictate them to the researcher, who would transcribe them into the text field. This option was offered to accommodate those with motor difficulties that made keyboard use particularly challenging. The final choice was always left to the participants¹⁰. For participants who opted for researcher-assisted transcription, responses were carefully recorded verba-

¹⁰For instance, one participant with fine-motor difficulties still preferred to type responses independently.

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

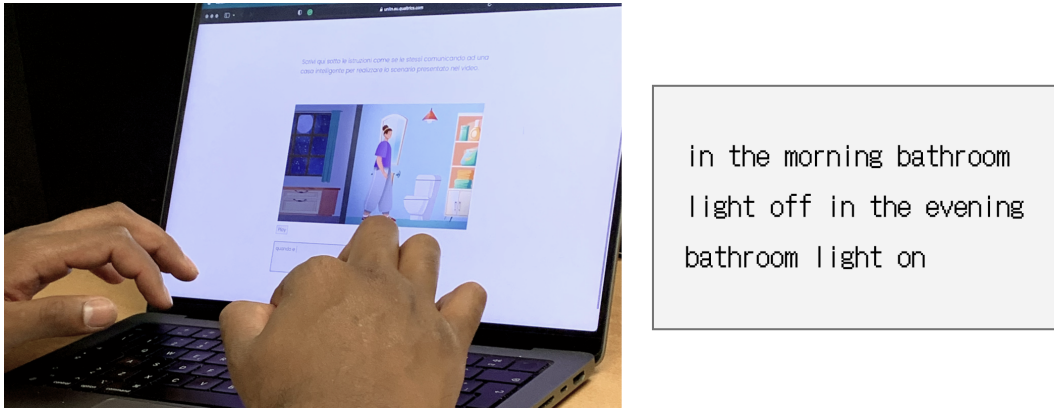


Figure 6.12: The figure represents a participant writing a rule after observing a video scenario during the study. On the right, an example of a generated rule.

tim without modifications or interpretations, ensuring that the original verbal structure and any grammatical errors were preserved. Once satisfied with their response, participants could press a "continue" button to proceed to the next scenario, repeating this task for all eight videos.

To minimize potential stress due to the study's duration (approximately 30 minutes), after the fourth video, participants were informed that they had reached the halfway point and could take a break if desired. They could resume the activity at any time by pressing the "continue" button.

To ensure the counterbalancing of video themes within each session, the videos were divided into two blocks of four, with each block containing one scenario from each category. The order of these blocks was counterbalanced across participants, and within each block, the sequence of the four videos was randomized to prevent potential biases related to viewing order. Additionally, each participant viewed half of the videos in their original version and the other half in a mirrored version. The presentation order of the original and mirrored versions was counterbalanced across participants. The sessions were audio recorded from the researcher's laptop.

Instruction Analysis. The instructions expressed by participants in natural language were further analyzed. These served as the primary dataset for classification (partially similar to the one described in Study 3 and Study 6), which was conducted independently by three researchers. Each instruction was classified based on the following criteria:

1. *The type of instruction.* Instructions were categorized into three distinct types:

- "*Rules*" when the instructions outlined a general guideline or condition, often introduced by conjunctions such as "*if*", "*when*", or "*while*", or structured as a sequence of conditions and events (e.g., "*during the night, close all the windows*"). Additionally, statements with a structured rule formulation were included in this category (e.g., "*a person enters the house, the lights turn on*").
- "*Command*" when instructions requested an immediate action without conditions (e.g., "*turn on the light*").
- "*Descriptions*" when instructions only described the scene shown in the video (e.g., "*a person enters the kitchen in this scene*").

2. *Its clarity.* Instructions were classified as "*non-ambiguous*" when they accurately specified both conditions and actions. Conversely, they were classified as "*ambiguous*" if they contained unclear phrasing, vague terminology, or lacked necessary details. Specifically, we employ the four themes related to ambiguities that emerged from Study 6 to guide the classification. An instruction is classified as ambiguous if it fits into at least one of the ambiguity categories related to "language", "outcomes", "content extension", and "system functionality".

3. *Its completeness.* Instructions were considered "*complete*" if they included all three elements of the scenario – state, event, and action. If any of these elements were missing or only implied, the instruction was labeled as "*incomplete*".

Further analysis was conducted exclusively on instructions classified as "*rule*" by at least two of the three researchers¹¹.

Rules Intent Analysis. To analyze the rules' intent, three independent and external judges were instructed to determine the intended meaning of each rule.

The judges recruited for this evaluation came from diverse academic backgrounds: one was a computer science student, another a law student, and the third an architect. None of them had prior experience with projects related to intellectual disability or specific knowledge of the study to ensure an unbiased perspective. Additionally, they were unaware of the origin of the rules and had no information about the characteristics of the videos.

¹¹The rationale is the one exposed in Study 3 and Study 6.

This external evaluation aimed to establish a baseline to assess the difficulty of interpreting the rules provided by participants¹².

Each judge received an example rule along with its corresponding intent description (specifically created to differ from the actual scenarios) and the following instructions:

"These are rules written by individuals to define the behavior of a home-automation system. We ask you to rewrite each rule in your own words to reflect its intended meaning."

During individual online sessions with a researcher, each judge evaluated the rules in a pseudo-randomized order, ensuring that rules from the same scenario were never presented consecutively. The intents described by the judges for each rule were then compared to assess agreement among them and to determine whether their interpretations aligned with the original scenario depicted in the video (see Table 6.14).

Two researchers independently analyzed the three intents described by the judges for each rule. The level of difficulty in interpreting the instructions was determined based on the judges' interpretations. A rule was classified as *"easy"* to interpret when all three judges provided similar interpretations. If two out of the three judges agreed on the intent, the rule was considered *"moderately difficult"*. Finally, if each judge provided a different interpretation, the rule was categorized as *"difficult"* to interpret, indicating a high level of ambiguity.

In addition, by comparing each intent provided by the judges with the *rule prototype*¹³, two researchers verified instances of *"perfect-intent-match"* in which at least one judge provided an interpretation that completely aligned with the original structure of the corresponding video scenario (i.e., including state, event, and action).

If researchers had conflicting assessments at any point of the analysis, discussions were held until reaching an agreement.

Phase 2 - The Study with ChatGPT

Using the instruction classified as *"rule"* as input, ChatGPT was tasked with three main prompts (see Table 6.15):

1. Generate pseudo-code based on the natural language rule and the context provided;

¹²External judges were recruited to avoid potential bias, as researchers involved in the study might have been influenced by their familiarity with the video design and the data collection.

¹³A *rule prototype* is a correct and complete rule that includes all the elements presented in the video.

Table 6.14: The table presents examples of intent descriptions provided by judges (*J1*, *J2*, *J3*) for participant-generated rules ("*rule*"). The "*comp.*" column indicates whether the rule was complete ("*yes*") or incomplete ("*no*"). The "*intent*" column specifies the interpretation difficulty level. Two scenarios ("*sc.*") were selected to illustrate perfect-intent-match ("*pim*") with rule prototype: *s4* ("*night, window opens, red light*"); *s1* ("*night, person enters bathroom, light*"). In **bold** are rules/description with a pim.

id	sc.	rule	comp.	J1	J2	J3	intent	pim
P4	<i>s4</i>	<i>opened the window and there was the alarm</i>	no	if alarm is on and window is opened, it should trigger	wants the alarm to sound when the window opens	wants the alarm to be triggered when someone opens the window	easy	0
P8	<i>s4</i>	<i>the girl opens the window when it is light the alarm did not flash when it is dark the alarm was triggered</i>	yes	in daytime, alarm is off, allowing windows to be opened. in evening, alarm is on, so if someone opens the windows, the alarm will be triggered	wants the alarm to sound if one tries to open the window in the dark	wants to activate alarm at sunset and deactivate it at sunrise	mod. diff.	2/3
P10	<i>s1</i>	<i>first it was morning and was in the bathroom then turned on the bathroom light</i>	no	when residents wake up in the morning, the lights must be on	wants bathroom light to turn on in the evening when entering the bathroom	wants the bathroom light to stay on during night and to turn off in the morning	diff.	1/3
P6	<i>s1</i>	<i>we would need emergency lights when it's all dark. Because [name] sometimes falls, once I found her here on my legs.</i>	yes	when it is dark, user would like to have some lights on	wants to have emergency lights on at night with the lights off	wants emergency lights to be on during the night	easy	0

Table 6.15: The table lists the three prompts used to query ChatGPT. The context table refers to the list of apartment characteristics shown in Table 6.16.

prompt id	prompt text
Code-from-rule	Given the following rule {row['rule']} and the following context {table}, make a pseudo-code. Just output the code without any further comments.
Check-rule	Given the following rule {row['rule']}, context {table}, and definitions {definitions}, identify if there are possible ambiguities or errors in the way the rule was written. Just output the list without any further comments.
From-rule-quest	Given the following rule {row['rule']} make a list of what clarification questions you would ask to reduce the ambiguity level of the received instruction. Just output the list without any further comments.

2. Identify possible ambiguities or errors within the instruction;
3. Generate a series of questions to disambiguate the provided instruction.

ChatGPT was also provided with contextual information, including a detailed list of smart devices, their specific locations across different rooms, and a basic floor plan. This method, adopted from Ghiani et al., 2017 and King et al., 2024, was employed to simulate a real-world scenario for customizing home automation environments. Details regarding the simulated house setting are presented in Table 6.16.

ChatGPT's Responses Analysis. The responses generated by ChatGPT were structured and assessed by two independent researchers. This evaluation involved verifying the logical and practical consistency of the code while ensuring its alignment with the users' prompts (prompt id *Code-from-rule* in Table 6.15). The generated pseudo-code snippets were analyzed based on the following criteria:

1. **Practical correctness** – their functionality within a practical environment. For example, when devices not present in the provided context are involved in the code or its deployment leads to security or safety issues.
2. **Formal correctness** – their adherence to standard programming structures. For example, the proper use of conditional statements or the presence of formal errors such as infinite loops.

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.16: The table shows the context provided to ChatGPT, including room elements, installed sensors, and smart devices.

room	elements	installed sensors	smart devices
Living room / kitchen	1 entrance, 2 doors, 1 window	Temperature/Humidity, Light, Noise, Entrance (doors/windows), Presence (movement), Smoke/fire	Door, TV, Window, Coffee Machine, Fridge, Ceiling lamp, Oven, Floor lamp, Thermostat, Smart plugs, Heater, Curtains
Bedroom	1 door, 1 window	Temperature/Humidity, Light, Noise, Entrance (doors/windows), Presence (movement)	Door, Smart plugs, Window, Curtains, Heater, Desk light, TV, Alarm, Ceiling lamp
Hallway	2 doors	Temperature/Humidity, Entrance (doors/windows), Presence (movement)	Door, Ceiling lamp, Window, Heater
Bathroom	1 door, 1 window	Temperature/Humidity, Light, Entrance (doors/windows), Presence (movement)	Door, Smart plug, Window, Toilet, Heater, Mirror, Ceiling lamp, Curtains

3. **Intent Recognition** – whether the code accurately reflected the user’s intent as determined by the human judges. When it did not match the intent, researchers annotated the reason (e.g., “condition missing” when an explicitly expressed element in the rule was not considered in the code).

Additionally, the two independent researchers examined the ambiguities and errors identified by ChatGPT in the user-generated instructions, along with the clarification questions proposed by ChatGPT to resolve them (respectively prompt id *Check-rule* and *From-rule-quest* in Table 6.15). A thematic analysis (Braun and Clarke, 2006) was conducted to identify emerging themes in ChatGPT outputs. We analyzed whether the questions that ChatGPT generated aimed to address the detected ambiguities or elements that did not allow for intent recognition. Two researchers independently annotated the questions indicating whether they were related to ambiguities detected or intent-related elements.

Results: Phase 1

The following paragraphs discuss the results of the first phase. For a summary of the procedure and results, see Figure 6.13.

Instruction Analysis. We collected 127 instructions from the 14 participants with intellectual disabilities. These included 112 written instructions and 15 transcribed from audio recordings. In some instances, participants provided only minimal written re-

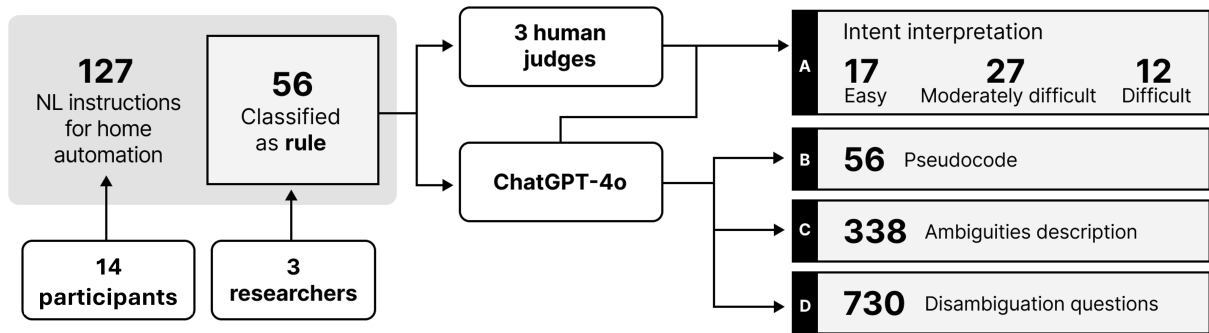


Figure 6.13: The figure visually summarizes the results of each study phase.

sponses, occasionally limited to a single word. Consequently, any verbal explanation given before or after writing the instruction – which could serve as an alternative clarification – was considered a second version of the instruction. For instance, participant P13, who wrote “*Occupied bathroom*” for scenario *s1*, verbally elaborated:

- P13 [referring to *s1*]: “*Basically, there was a guy who went from the living room to the bathroom, but the other one can’t go in because there’s already someone else inside. So, the bathroom is occupied*”.

Similarly, when presented with scenario *s8*, P13 initially wrote: “*Extinguish the fire with the fire protection system*”. However, the verbal explanation provided additional details:

- P13 [referring to *s8*]: “*At first, the temperature was low, but as it rose, what was in the pot overheated, causing a small fire, and the fire alarm extinguished it. If something is burning, I’d say to put out the fire using electricity*”.

In such cases, both textual and verbal versions were included in the dataset as separate instructions.

Among the 127 instructions provided, 56 were categorized as “rules”, 14 as “commands”, and 57 as “descriptions” (see Table 6.17). For the aim of the study, all subsequent analyses considered only those classified as “rules”. Regarding completeness, 18 of the 56 rules contained all the elements (state, event, action) of the video scenario and were classified as “complete”. The remaining 38 were categorized as “incomplete”. All 56 rules were classified as “ambiguous” by the researchers, as each contained at least one unclear structure, terminology, or missing detail.

Table 6.17: The table shows the results of the instruction analysis performed by three researchers.

id	production		completeness		type of instruction		
	textual	vocal	complete	incomplete	command	description	rule
P1	8	0	1	7	4	0	4
P2	8	0	0	8	0	8	0
P3	8	0	0	8	4	3	1
P4	8	2	0	10	1	5	4
P5	8	0	5	3	0	5	3
P6	8	5	0	13	2	4	7
P7	8	0	2	6	0	3	5
P8	8	0	2	6	0	3	5
P9	8	0	1	7	0	5	3
P10	8	1	2	7	0	7	2
P11	8	0	0	8	1	1	6
P12	8	0	4	4	0	1	7
P13	8	7	5	10	2	6	7
P14	8	0	2	6	0	6	2
<i>total</i>	112	15	22	103	14	57	56

Intent Analysis. Since all rules were classified as “**ambiguous**”, the intent categorization provided by the three independent judges was used to establish a more consistent assessment of the clarity of participant-generated rules. Based on the level of agreement among judges:

- 17 rules (30.36%) were classified as “**easy**” to interpret, as all three judges agreed on the same intent (3/3 agreement).
- 27 rules (48.21%) were considered “**moderately difficult**”, meaning that one judge’s interpretation differed from the other two (2/3 agreement).
- 12 rules (21.43%) were categorized as “**difficult**” to interpret, as each judge provided a different interpretation (1/3 agreement).

By cross-referencing clarity and completeness (see Table 6.18), we found that among the “incomplete” rules ($n = 38$), 8 were classified as “difficult” to interpret, 19 as “moderately difficult”, and 11 as “easy”. Among the “complete” rules ($n = 18$), 4 were considered “difficult”, 8 “moderately difficult”, and 6 “easy”. A **perfect-intent-match** occurred in 26 cases where at least one judge provided an interpretation that fully aligned with the original structure of the corresponding video scenario (i.e., including state, event, and action). Interestingly, in 10 of these cases, at least one judge

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.18: The table shows the results of the intent analysis of the 56 rules, divided between complete and incomplete. In the lower part of the table we reported the data regarding *perfect-intent-match* cases divided by level of difficulty of intent interpretation reporting the ratio of judges who had a *perfect-intent-match* over the total.

<i>intent</i>	complete ($n = 18$)	incomplete ($n = 38$)
easy	6	11
moderately difficult	8	19
difficult	4	8
<i>perfect-intent-match ratio</i>	($n = 16$)	($n = 10$)
easy – 3/3	4	0
moderately difficult – 2/3	6	2
moderately difficult – 1/3	2	5
difficult – 1/3	4	3

had a perfect-intent-match even though the rule was categorized as “**incomplete**” (e.g., P10’s rule in Table 6.14). This suggests that the judge inferred at least one element that was not explicitly stated in the participant’s rule. Further analysis of these 10 cases revealed:

- In 2 instances, two judges agreed on the perfect-intent-match (“moderately difficult”; 2/3 agreement on perfect-intent-match).
- In 5 cases, only one judge achieved a perfect-intent-match, while the other two agreed on an incomplete intent (“moderately difficult”; 1/3 agreement on perfect-intent-match).
- In 3 cases, the rule was classified as “**difficult**” to interpret, indicating that no consensus was reached on the rule’s intent (1/3 agreement on perfect-intent-match).

For the remaining 16 cases, the rules were classified as “complete” (i.e., participants included all scenario elements). The analysis revealed:

- In 4 cases, all three judges reached a perfect-intent-match (“easy”), and these instances were exclusively observed among “**complete**” rules (3/3 agreement on perfect-intent-match).
- In 6 cases, two judges agreed on perfect-intent-match interpretation (“moderately difficult”; 2/3 agreement on perfect-intent-match).

- In 2 cases, only one judge identified a perfect-intent-match, while the other two agreed on an incomplete version ("moderately difficult"; 1/3 agreement on perfect-intent-match).
- In 4 cases, the rule was classified as "difficult" to interpret, as only one judge provided a perfect-intent-match (1/3 agreement on perfect-intent-match).

Notably, even if these 16 rules were "**complete**", in 6 instances, two judges did not recognize all the elements of the original scenario.

Results Phase 2

The following paragraphs discuss the results of the second phase. For a summary of the procedure and results, see Figure 6.13.

Correctness of ChatGPT-Generated Code. The 56 code instances generated by ChatGPT for controlling home-automation scenarios, based on user instructions (prompt id: *Code-from-rule* in Table 6.15), were evaluated according to:

1. Their **practical correctness**, which assesses whether the generated code would function as intended in a real-world setting. Among the generated code instances, 37 were deemed practically correct, whereas 19 contained errors, resulting in a functional success rate of 66.07%.
2. Their **formal correctness**, which evaluates adherence to standard programming structures, including the proper use of conditional statements. Only 25 instances met the required structural standards, while 31 were classified as formally incorrect (55.36%). A frequent formal error in trigger-action logic was the absence of an *end-of-action* condition. For example, in some cases, a device (e.g., a heater) remained on indefinitely, even after the initial activation conditions were no longer met.

The results highlighted a discrepancy between practical and formal correctness (see Table 6.19). In 20 instances, these two assessments conflicted:

- 16 cases were **practically correct but formally incorrect**, indicating that functionally sound code often lacked proper structure;
- 4 cases were **formally correct but practically incorrect**, meaning that structurally well-formed code would fail to execute correctly in real-world conditions.

Table 6.19: The table shows the frequencies with which the codes ($N = 56$) were evaluated for formal and practical correctness and user intent recognition. Specifically, we reported how many times ChatGPT recognized intent, how many times it had a perfect-intent-match (*pim*) with the original video, and how many of these times it outperformed the human judges. We reported the total counts and divided them for completeness and the level of difficulty in interpreting the rule.

rules	correctness		intent		
	<i>practical</i>	<i>formal</i>	<i>recognized</i>	<i>pim</i>	<i>outperform</i>
Total ($N = 56$)	37 (66.07%)	25 (44.64%)	27 (48.21%)	7 (12.50%)	5 (8.93%)
complete ($n = 18$)	10	9	4	4	2
incomplete ($n = 38$)	27	16	23	0	3
easy ($n = 17$)	11	9	6	1	1
mod. difficult ($n = 27$)	19	12	17	2	2
difficult ($n = 12$)	7	4	4	4	2

While analyzing the practical correctness of pseudo-codes, we observed some cases where the code would lead to real-life security risks. We will discuss these further in a dedicated paragraph **Safety and Practical Limitations**.

Intent Recognition in ChatGPT-Generated Code. Out of the 56 evaluated codes, ChatGPT successfully recognized the intent in 27 instances, corresponding to 48.21% of the total rules (see Table 6.19). Regarding rule completeness, ChatGPT correctly identified the intent in 4 out of the 18 "**complete**" rules. Among the 38 "**incomplete**" rules, it successfully recognized the intent in 23 cases. Performance was further analyzed based on interpretation difficulty:

- For "**easy**" rules ($n = 17$), ChatGPT correctly understood the intent in 6 cases.
- For "**moderately difficult**" rules ($n = 27$), intent was recognized in 17 cases.
- For "**difficult**" rules ($n = 12$), intent was identified in 4 cases.

ChatGPT had a **perfect-intent-match** in 7 instances. In 5 of these, ChatGPT **outperformed** human judges by detecting conditions or elements that the majority of evaluators overlooked. Specifically:

- In 2 cases, ChatGPT's interpretation aligned with one judge's intent description in "difficult" rules, thereby validating its accuracy (in these cases, ChatGPT's performance was categorized as perfect-intent-match and intent recognized).

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.20: The table reports frequencies for practical and formal correctness of the codes ($N = 56$) when the intent was (or was not) recognized by ChatGPT (respectively, $n_{yes} = 27$ and $n_{no} = 29$), along with χ^2 test results are also reported.

correctness	intent	n	(%)	χ^2	df	p
practical	no	16	(55.17%)	3.19	1	0.07
	yes	21	(77.79%)			
formal	no	11	(37.93%)	1.10	1	0.30
	yes	14	(51.85%)			

- In 2 other cases, ChatGPT’s interpretation matched one judge’s understanding of “moderately difficult” rules (in these cases, the intent was not recognized but there was a perfect-intent-match). In yet another 2 moderately difficult rules, ChatGPT had a perfect-intent-match aligning with the majority of judges.
- In 1 instance, ChatGPT outperformed all three judges for a rule classified as “easy” to interpret.

We explored if there were differences between the features of rules when ChatGPT fully captured the original intent (perfect-intent-match, $n = 7$) and when it did not ($n = 49$)¹⁴. We observed that the average length of the prompt was 28.14 words ($SD = 12.25$) when ChatGPT achieved a perfect-intent-match while it was 19.16 words ($SD = 11.84$) when ChatGPT failed.

Cross-referencing ChatGPT’s intent recognition frequencies with code correctness revealed some trends (see Table 6.20). Regarding “practical correctness”, the generated code was correct in 55.17% of cases where ChatGPT did **not** recognize the intent. However, when the intent was correctly identified, practical correctness increased to 77.78%. A similar trend emerged for “formal correctness”: when ChatGPT failed to recognize the intent, formal correctness was achieved in 37.93% of cases, whereas when the intent was recognized, formal correctness increased to 51.85%. These findings suggest that intent misinterpretation led to a higher likelihood of errors. However, when performed χ^2 tests, these tendencies were not significant [both $\chi^2_s(1) < 3.19$, $p_s > 0.07$; see Table 6.20].

We then analyzed whether there was a relationship between the number of ambiguities detected (prompt id: *Check-rule*; Table 6.15) or the number of questions produced (prompt id: *From-rule-quest*) and the recognition of the intent by ChatGPT. Since the

¹⁴Aware of the large difference between the samples, we did not perform any statistical tests.

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.21: The table shows the averages (M), standard deviations (SD), minimum (min), and maximum (max) of ambiguities ("*ambiguity_n*") and questions ("*question_n*") per rule resulted from ChatGPT interactions. The results of *Mann-Whitney U tests* – performed to test for differences between the number of items produced by ChatGPT and the recognition of rule intent – are also reported.

	intent	M	SD	min	max	U	p
ambiguity_n	no	6.14	1.64	4.00	11.00	372.50	0.76
	yes	5.93	1.66	3.00	11.00		
question_n	no	12.52	3.10	10.00	20.00	500.00	0.07
	yes	13.52	2.49	10.00	20.00		

data did not follow a normal distribution (see Appendix K), to evaluate these differences, we employed the *Mann-Whitney U test* (see Table 6.21).

For ambiguity detection, a slightly higher number of ambiguities ($M = 6.14$, $SD = 1.64$ items) was detected when ChatGPT failed to recognize the intent compared to when the intent was recognized ($M = 5.93$, $SD = 1.62$ items). The *Mann-Whitney U test* showed no significant difference ($U = 372.50$, $p = 0.76$), suggesting that intent recognition had minimal influence on ambiguity detection. Similarly, the number of generated questions was slightly higher when the intent was recognized ($M = 13.51$, $SD = 2.41$ items) compared to when it was not ($M = 12.52$, $SD = 3.10$ items). The *Mann-Whitney U test* showed a marginally non-significant result ($U = 500.00$, $p = 0.07$).

Thematic Analysis on ChatGPT-Detected Ambiguities and Errors. The second prompt used to query ChatGPT (prompt id: *Check-rule*; Table 6.15) aimed to obtain a list of ambiguities and errors identified by the model in the user-generated instructions. ChatGPT returned 338 items, consisting of 178 ambiguities and 160 errors. The number of errors and ambiguities identified per instruction ranged from 3 to 11, with most rules presenting between 5 and 7 issues. Six (6) results were excluded from the following thematic classification due to their recognition as model hallucinations, such as: '*The terminology varies slightly (e.g., "Ceiling lamp" versus "Ceiling lamp" without capitalization), which, although minor, might confuse automated systems parsing this data*' (note that the two terms are exactly the same).

All identified ambiguities and errors fall into the four categories identified in Study 6 without revealing any new categories of ambiguity. Of the 332 results, 138 occurrences (41.57%) fell under the category "*Ambiguities related to the necessity of content extension*", where all instances were identified as ambiguities related to lack of

"Specifications on States" ($n = 138$ occurrences), indicating the absence of conditions or alternative states within the rule, thus obscuring the intended behavior. For example, for the rule "In the morning bathroom light off in the evening bathroom light on", ChatGPT highlights that the information is incomplete, stating:

- **Context Extension – Specifications on States:** "The rule specifies the state of the bathroom light during the morning and evening **but does not specify its state during the afternoon or night**. This leaves a gap in understanding how the system should behave during those times".

One-hundred-thirteen (113) occurrences (34.04%) pertained to "Language". Among these, the most prominent ambiguity category was related to "Language Used" ($n = 33$ occurrences), showing a lack of clarity. An example is the rule expressed by participant P1 [referring to s8] "you leave something on the stove, a fire start, and the burglar alarm extinguished the fire" where the "burglar alarm" is mistakenly used instead of referring to the "fire prevention system" as highlighted by ChatGPT:

- **Language – Language Used:** "The 'burglar alarm extinguished the fire' implies that an anti-theft device (typically used for burglar protection) is responsible for fire suppression, which is unconventional."

This was followed by "Word Precision" ($n = 28$ occurrences) category, related to overly specific or overly generic terms, and "Figurative Value" ($n = 27$ occurrences), associated with generic expressions (e.g., "night") rather than defining specific values (e.g., "from 8 PM to 6 AM"). Similarly, state descriptions like "the light was on" are open to varied interpretations, as they fail to specify details such as brightness levels, timing, or specific conditions. Moreover, temperature-related expressions like "when it's cold" are highly subjective, as they provide no quantitative definition, resulting in different interpretations depending on individual perceptions.

Furthermore, 42 occurrences (12.65%) were classified as "Ambiguities related to the outcome of the rule and its variables", with 24 occurrences under "Error Handling", reflecting the lack of specifications to ensure proper management of failures or conflicts in logic. Lastly, 39 occurrences (11.75%) were categorized as "Ambiguities related to the system's functioning", with 33 occurrences classified as "Sensor List", associated with sensor localization and integration within a broader system.

Thematic Analysis on ChatGPT-Generated Disambiguation Questions. With the third prompt (prompt id: *From-rule-quest*; Table 6.15), ChatGPT generated a total of

730 questions to clarify the 56 rules. The number of questions for rules whose intent was recognized ($n = 365$, $M = 13.52$, $SD = 2.41$ items) did not differ from those where the intent was not recognized ($N = 363$, $M = 12.52$, $SD = 3.10$ items), as reported in paragraph **Intent Recognition in ChatGPT-Generated Code**. We focused our thematic analysis on the 363 questions related to the 29 rules for which ChatGPT failed to identify the intent. Our goal was to examine the most common issues addressed by ChatGPT when the code generated did not align with the participants' intentions. From this analysis, we identified four main themes.

Theme 1: Questions related to the necessity of content extension (186 instances). The theme included 77 questions related to "*Specifications on States*", 43 questions focused on requests for more "*Details on Actions*" to be performed. Furthermore, 30 questions addressed possible "*Exceptions and Alternative Cases*", 23 on "*Specifications on Sensors*" to be considered, and 13 questioned if there should be a "*Delay on Actions*" to be taken. Here are some examples of questions generated by ChatGPT that fall within this theme.

- **Specification on States:** "*Does the rule apply every morning or just under certain conditions?*" [Rule: First it was morning and was in the bathroom then turned on the bathroom light].
- **Details on Actions:** "*How bright should the emergency lights be?*" [Rule: We would need emergency lights when it's all dark. Because *name* sometimes falls, once I found her here on my legs].
- **Exceptions and Alternative Cases:** "*Are there any exceptions to when the window should not be opened or the alarm not sounded?*" [Rule: in the morning a girl opens the window and at night she opens the window and the alarm sounds].
- **Specification on Sensors:** "*What kinds of sensors or inputs are available to determine bathroom occupancy?*" [Rule: To say when there is someone in the bathroom not to open the door. with a voice say it is busy].
- **Delay on Actions:** "*How quickly should the system respond once an attempt to open the door is detected?*" [Rule: To say when there is someone in the bathroom not to open the door. with a voice say it is busy].

Theme 2: Questions related to the outcome of the rule and its variables (95 instances). On 43 occasions, ChatGPT asked to provide more details on the general "*Motivations*

and Intentions” of users in defining the rule. In 14 questions, ChatGPT asked information about the “End of the Action” and in 13 about “Error Handling”. Eleven (11) questions were related to “Safety and Security” aspects. Eight (8) questions regarding the presence of “Automation” questioning if the action is supposed to be manual or automated; 6 questions were related to specification on “Temporal Aspects”, for example, the sequentially of events. The following are examples of questions related to each sub-theme.

- **Motivations and Intentions:** “What is the purpose of opening the window in both the morning and at night?” [Rule: in the morning a girl opens the window and at night she opens the window and the alarm sounds].
- **End of the Action:** “Should the light be turned off at any point afterward?” [Rule: he goes to the bathroom to do his needs, the light turns on].
- **Error Handling:** “In case of power failure, how should the system behave?” [Rule: To say when there is someone in the bathroom not to open the door. with a voice say it is busy].
- **Safety and Security:** “Are there any safety precautions to be followed while opening the window or sounding the alarm?” [Rule: in the morning a girl opens the window and at night she opens the window and the alarm sounds].
- **Automation:** “Who is supposed to open the window, Alexa or the user?” [Rule: Alexa open the window gosh it doesn’t work I’ll try without it doesn’t work then I’ll open it . I’ll try again now that dark ALEXA but it doesn’t work then attention, and alarm went off].
- **Temporal Aspects:** “Is there any specific sequence to the actions taken at night (opening the window first then sounding the alarm)?” [Rule: in the morning a girl opens the window and at night she opens the window and the alarm sounds].

Theme 3: Questions related to the language used (55 instances). Twenty-nine (29) questions were related to an explanation of the “Structure” of the instructions and “Words” used, and 26 requested to specify values when “Figurative Values” were used.

- **Structure and Words Used:** “What does ‘had to go to the toilet’ precisely mean in terms of actions to be taken?” [Rule: The girl had to go to the toilet first the door opened. The second time the light was triggered because there was another person inside].

- **Figurative Value:** "What specific time in the morning does the girl open the window?" [Rule: in the morning a girl opens the window and at night she opens the window and the alarm sounds].

Theme 4: Questions related to specifications on system functioning (27 instances). These questions related to monitoring the system, the necessity of installing or connecting additional devices, and details on the system functioning in general.

- **System Functioning:** "Who or what entity is responsible for following or enforcing this rule?" [Rule: opened the window and there was the 'alarm].

We analyzed whether the questions generated aimed to address detected ambiguities or elements that did not allow intent recognition by ChatGPT.

Questions Addressing Ambiguities. The themes that arose from the qualitative analysis of questions generated are similar to those that emerged during the analysis of the detected ambiguities. In fact, 91 questions (25.07%) were intended to answer the ambiguities detected by ChatGPT. For each rule, at least 1 (with a maximum of 7) questions addressed an ambiguity ($M = 3.14$, $SD = 1.55$ questions).

By analyzing the type of theme of the questions addressing ambiguities, we saw that the most common theme was about the need to extend the content (*Theme 1*) and specifically about *Specification on States* for both "difficult" rules ($n = 22$ questions addressing ambiguities detected; $n = 6$ related to *Specification on States*) and "moderately difficult" rules ($n = 29$ questions addressing ambiguities detected; $n = 8$ related to *Specification on States*). In contrast, for "easy" questions addressing ambiguities ($n = 41$ questions addressing ambiguities detected), the most common theme of questions was to disambiguate the language used by the user (*Theme 3*; $n = 11$ related to the language used).

Questions Addressing Intent Recognition. Overall, 56 questions (15.43%) were identified as being related to clarifying what ChatGPT missed or misinterpreted in specific instructions interpretation. On average, each rule had at least one question related to the element for which ChatGPT did not reflect intent ($M = 1.93$, $SD = 1.62$ questions) with a minimum of 0 questions (in 4 cases) and a maximum of 6 (in 2 cases). Of these questions related to intent recognition, 15 questions also address ambiguities detected. The most common themes in the questions related to the recognition of intent were that of the extent of content (*Theme 1*) and precisely *Specification on States* ($n = 17$) and that related to outcome (*Theme 2*) specifically the reasons underlying the rules, i.e., *Motivations and Intentions*.

Quantitative Analysis on Ambiguities and Questions. We analyzed the number of

6.2. EXPLORING HOW LARGE LANGUAGE MODEL-BASED SYSTEMS ADDRESSES AMBIGUITIES IN LANGUAGE

Table 6.22: The table reports descriptive statistics and non-parametric tests performed on ambiguities detected and questions generated by ChtGPT considering rules’ completeness and clarity of intent.

	rule	M	SD	min	max	test		p
ambiguity_n	easy	6.18	1.59	4.00	11.00	F	4.77	0.02
	moderate	6.41	1.65	3.00	11.00			
	difficult	5.00	1.21	3.00	7.00			
	complete	6.67	1.82	3.00	11.00	F	3.63	0.07
	incomplete	5.78	1.45	3.00	11.00			
question_n	easy	13.06	2.33	10.00	18.00	F	1.60	0.23
	moderate	12.33	2.24	10.00	16.00			
	difficult	14.48	4.06	10.00	20.00			
	complete	12.17	2.57	10.00	18.00	U	428.50	0.12
	incomplete	13.40	2.86	10.00	20.00			

ambiguities and questions generated in relation to rule classification. As the data showed no normality of distribution, to assess the differences between the number of ambiguities/questions per “complete” and “incomplete” rules, we used a non-parametric *Mann-Whitney U test*. To measure the differences between “easy”, “moderately difficult”, and “difficult” rules, we adopted *Welch’s ANOVA* since the test of homogeneity of variances indicated an inhomogeneous variance. All assumption checks are reported in Appendix K.

Ambiguities. Considering rules’ completeness and intent clarity, we notice a very slight increase in the average number of ambiguity detected for “complete” rules ($M = 6.67$, $SD = 1.82$ items) compared to “incomplete” rules ($M = 5.78$, $SD = 1.45$ items) and for “moderate” and “easy” rules (respectively, $M_{mod} = 6.41$, $SD_{mod} = 1.65$ and $M_{easy} = 6.18$, $SD_{easy} = 1.59$ items) compared to “difficult” ones ($M_{diff} = 5.00$, $SD_{diff} = 1.23$ items). As reported in Table 6.22, the difference between rules with different completeness was not significant. Interestingly, the *Welch-ANOVA* showed a significant difference between the number of ambiguities detected and the clarity of intent ($F = 3.63$, $p = 0.02$). *Bonferroni post-hoc test* (reported in Appendix L) confirmed that the number of ambiguities detected for rules that were “difficult” to interpret was significantly lower than that for “moderately difficult” rules ($F = -2.62$, $p_{bonf} = 0.03$).

Questions. Considering rules’ completeness and intent clarity, we notice a very slight

increase in the average number of question generated for "incomplete" rules ($M = 13.40$, $SD = 2.86$ items) compared to "complete" rules ($M = 12.17$, $SD = 2.57$ items) and for "difficult" rules ($M = 14.48$, $SD = 4.06$ items) compared to the other two categories (see Table 6.22). However, in both cases, the differences were not statistically significant (results are reported in table Table 6.22).

Relation of Ambiguities and Questions. To evaluate the relationship between the number of questions generated ($M = 13.00$, $SD = 2.81$ items) and the number of ambiguities detected ($M = 6.04$, $SD = 1.62$ items) in the 56 rules, we opted for a non-parametric test, as the data showed no normality of distribution (see Appendix K). The *Spearman's rank correlation coefficient* suggests a weak and statistically non-significant correlation between the number of ambiguities identified and the number of questions generated ($\rho = -0.16$, $p = 0.24$).

Safety and Practical Limitations While analyzing the code correctness of pseudocodes, we highlight instances of the emergence of potentially unsafe suggestions from ChatGPT. In some instances ($n = 6$), the pseudo-code included actions that could pose safety risks. For example, 5 times the code generated by ChatGPT triggered an evacuation procedure – after the detection of a fire or smoke – that consists of opening all doors/windows and turning on ceiling lights. These actions could pose a significant danger in a real fire emergency. As an example, we report one of the pseudo-code snippets [Rule: you have to warn if the house is on fire] highlighting unsafe actions.

```
FUNCTION checkFireAlerts()
  rooms = ["Living room/Open kitchen", "Bedroom", "Hallway", "Bathroom"]
  FOR each room IN rooms
    IF smokeSensorDetectedFire(room) THEN
      notifyUser("Warning: Fire detected in " + room)
      activateEmergencyDevices(room)
    END IF
  END FOR
END FUNCTION

FUNCTION activateEmergencyDevices(room)
  // This function activates emergency devices in the specified room
  devices = getSmartDevices(room)
  FOR each device IN devices
    IF device EQUALS "Door" THEN
      unlockDoor(room)
    ELSE IF device EQUALS "Window" THEN
      openWindow(room)
    
```

```
ELSE IF device EQUALS "Ceiling lamp" THEN
    turnOnCeilingLamp(room)
ELSE IF device EQUALS "Heater" THEN
    turnOffHeater(room)
END IF
END FOR
END FUNCTION
```

In the other case ($n = 1$), the code called for locking the bathroom door until movement is detected inside, which poses the risk of getting stuck inside the bathroom. However, the analysis of ChatGPT's questions reveals that the system often incorporates considerations related to safety or unintended consequences based on how the prompt was phrased. For instance, ChatGPT suggests questions such as:

- *"Are there any specific safety precautions that need to be taken before attempting to put out the fire?"* [**Rule:** If the house is on fire, you have to put out the fire with water].
- *"Are there additional safety measures that should accompany the activation of the alarm?"* [**Rule:** Turn on the burglar alarm when you leave something burning on the fire].

These questions require the user to enter crucial information for handling errors or dangerous situations or to prevent the system from deducing the safest line of action.

Discussion

We collected 127 home instructions from 14 individuals with intellectual disability. Fifty-six (56) were classified as "rules" as they contained, to some extent, the definition of a trigger and an action. All rules were classified as ambiguous, fitting one or more ambiguity categories identified in Study 6.

Understanding individuals' intentions can be challenging, especially when language is ambiguous and words may not accurately reflect true intentions. This complexity persists even if we manipulate the final goal, such as defining the home automation scenario presented in the video. To address this issue, we involved three human judges and asked them to perform an intent-recognition task on each rule, aiming to understand the difficulty level in intent interpretation. We classified three levels of difficulty based on the judges' consensus. However, it is essential to consider that while, in some cases, multiple evaluators interpreted the instructions similarly, suggesting lower ambiguity, this agreement does not necessarily imply that the identified intent aligned with what the user intended after watching the initial video scenario. This further highlights the complexity of interpreting and generating code to match the true intent of users, specifically those with intellectual disability.

Similarly to Study 6, we prompted ChatGPT to generate pseudo-code, detect ambiguities in the rule, and define disambiguation questions.

We observed that pseudo-codes generated were not highly correct both practically (66.07%) and formally (55.36%), sometimes leading to concerning implications on **safety**. This result can be related to LLM's unpredictability, and no consistent patterns in its code-generation performance. These results are aligned with previous studies that have examined ChatGPT's code-generation tasks (Ouyang et al., 2023). However, the results could also be related to the high ambiguity of rules generated by our participants with intellectual disability, who often have challenges in language-based communication, as observed by (Coppens-Hofman et al., 2017).

ChatGPT's ability to interpret intentions with highly ambiguous instructions was less than (48.21%). As we don't actually know the real intent of rules, we based our classification of intents on human judges' agreement. Sometimes ($n = 5$) humans overlooked some elements that instead ChatGPT recognized and addressed in the code.

This may be connected to the extensive data that ChatGPT is trained on, which can make its information about specific domains more comprehensive than human judges.

While this could be beneficial, there is a possibility that the information provided may be inaccurate. Since ChatGPT's default responses convey certainty instead of highlighting areas of ambiguity, it fills in those gaps with autonomously chosen information. This could lead to potential security and safety issues, as we observed.

In the same direction, the quantitative analysis showed a significant relationship between the number of ambiguities detected by ChatGPT and the level of difficulty in interpreting the intent, indicating that fewer ambiguities were detected for "difficult" rules than for "moderately difficult" ones.

Conversely, our results indicated no statistical significance in the relationship between ChatGPT performance in interpreting the rule's intent and code quality or rule features. The only insight we had, which requires further investigation, is that the length of the prompts may be related to ChatGPT's strong performance.

This is interesting since linguistic ambiguities increase as the number of words in the prompt increases, but since the longer the sentences are, the greater the margin of interpretation that exists, which increases the accuracy probabilities of ChatGPT's performance.

From a thematic analysis of the ambiguities detected in the user-generated rules, we observed that all the ambiguities and errors fitted within the same themes emerged in Study 6, suggesting that they are probably the ones identified by the model regardless

of sample characteristics.

From the thematic analysis of the generated questions, we observed that they fell into similar categories as those that emerged for ambiguities. However, of the 363 clarification questions generated for rules in which intent was not recognized by Cht-GPT, only 25.07% of the cases were related to the ambiguities detected, and 15.43% were related elements that caused ChatGPT failure in recognizing the rule's intent. From the quantitative analysis, we found no significant relationship between the number of questions and rule features. Our results suggest that the number of unhelpful questions was high/excessive, but it encloses possibilities in using ChatGPT-generated disambiguation questions (King et al., 2024) to guide user interaction to refine trigger-action rules that are less ambiguous and, hopefully, closer to the user's intent.

Limitations. We acknowledge several limitations of this study. First, the results are based on instructions generated by a small sample, which may influence the outcomes. Future studies involving a larger number of participants are necessary to support these findings.

Also, defining instructions from video scenarios could be more challenging for some participants than for others, depending on specific cognitive abilities. While this was a study design choice to have more control over individual intents, and this aspect has been addressed partially through the educator's assistance and the possibility of expressing the instructions verbally, it could have impacted the answers of some participants. Furthermore, the 55.90% of the instructions generated by the participants were not rules but commands or descriptions. By excluding those instructions, we lost a part of participants' expressiveness that would be interesting to investigate with the same method.

Another limitation is related to the intent recognition task which was performed just by three human judges. It could be important to expand the number of judges involved to have a more accurate assessment of the level of difficulty of the rules' interpretation. Lastly, while the latest version of the ChatGPT model was employed in this study, LLM's capability is rapidly evolving, and results related to intent understanding and code generation may require re-evaluation with future model versions. Nevertheless, it is essential to note that technological advancements do not automatically lead to improvements in accessibility, thus this study could represent a baseline to benchmark future versions.

Chapter 7

Toward New Interaction Approaches

This final Chapter describes two streams of research and development we are starting to explore. In the first section (7.1), it is presented *RuleCraft* (Andrao et al., 2024a). This is an online module that can be used by third-party software to define trigger-action rules, and their context, with a **form-based compositional interface**. As *RuleCraft* is still under development, here we present just its first application in the educational setting. In the second section (7.2), it is presented an initial investigation of the use of LLMs to support trigger-action definition in an iterative approach. We present a pilot study, conducted in collaboration with the *University Polytechnic of Milan*, in which we had a first exploration of how individuals with intellectual disability interact with a **LLMs-based prototype** for creating home automation adapting their environment to their needs and the implications in improving individuals autonomy.

7.1 RuleCraft: an End-User Development Hub for Education

As discussed in subsection 6.1.3, in school environments, the use of smart devices, including IoT, is gaining popularity for their potential to improve various aspects of teaching and learning (Ramlowat and Pattanayak, 2019). Another way to increase student engagement and motivation in the learning process is gamification (Majuri et al., 2018; Oliveira et al., 2023), defined as the use of game elements in non-game contexts (Deterding et al., 2011). As gamification aims to support people’s involvement and engagement in complex tasks, it has often been employed in education (Manzano-León et al., 2021; Oliveira et al., 2023).

We connected *SMARTER* with a gamified interface named *SmartGame* (see Gini et al., 2023; Figure 7.1) to enhance the children’s learning experience. *SmartGame* was



Figure 7.1: Three screenshots of the *SmartGame* web interface and a scenario in which *SmartGame* is connected to two *SMARTERs* modules are shown in the figure.

designed to enable both the use of a single *SMARTER* or multiple modules to allow cooperative learning, joining more *SMARTERs*. We conducted a first usability evaluation during a workshop with 21 children aged between 9 and 11 ($M = 10.05$, $SD = 0.38$ years). The assessment revealed good usability and a positive attitude toward the system, as described in Gini et al. (2024).

However, enabling teachers and educators to customize smart tools and the functioning of gameful systems may contribute to their effectiveness in educational settings (Koivisto and Hamari, 2019; Oliveira et al., 2023; Wiedenbeck, 2005).

Thus, we designed a new EUD form-based compositional platform, named *RuleCraft*, to enable teachers to define the functioning of one or more educational tools, in this specific case of *SMARTER* and *SmartGame*, across various school subjects.

Additional, the system supports the upload of multiple *vocabularies*, allowing other domain experts to create and upload material to extend the EUD solution to different topics, subjects, and tools.

RuleCraft was presented in Andrao et al. (2024a) (and partially in Andrao et al., 2023c).

Linguistic Clues and Primitives in *RuleCraft*

Based on our findings regarding language and mental models, described in Chapter chapter 5, we designed the front-end of the user interface to explicitly separate states and events, allowing users to create trigger-action rules in the form of "As soon as <event> While <state(s)>, <action(s)>" (see Figure 7.2).

We proposed a preliminary attempt of vocabulary for game elements (Andrao et al., 2023c). We first explore the taxonomy of game elements proposed by Toda et al. (2019),

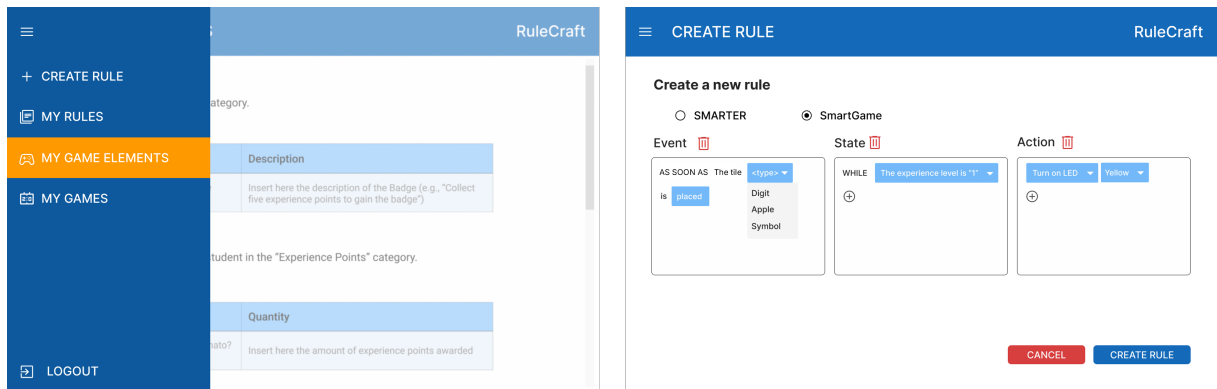


Figure 7.2: In the figure, two screens of *RuleCraft* are presented. On the left is visible the lateral menu while on the right the trigger-action rule creation page.

Table 7.1: Examples of game element-based language primitives for events, states, and actions

game element	[AS SOON AS] events	[WHILE] states	action
Acknowledgment: e.g., badge	"User gains <N> acknowledgment(s)"	"User has <N> acknowledgment(s)"	"Assign <N> acknowledgment(s)"
Level: e.g., profile level	"User reaches the level <N>"	"User is at level <N>"; "User is at a level lower[higher] than <N>"	"Level up"
Points: e.g., experience points	"User gains points"	"User has <N> points"; "User has more[less] than <N> points"	"Assign <N> points"

in order to select and implement verbal primitives (see Table 7.1 for examples of the game elements and the related language primitives).

Considering the results presented in Chapter 6, we have implemented both abstract and concrete versions of the primitives, which will be evaluated with users in the future.

7.1.1 The System Architecture

RuleCraft (see Figure 7.3) is an existing platform for domain experts to craft and refine *vocabularies* tailored to specific learning domains (e.g., math), including motivational aspects (i.e., game elements). Domain experts can be represented for example by researchers, developers, or educators who are interested in developing a new set of *verbal primitives* for innovative EUD educational tools. By verbal primitives, we in-

tend basic and understandable commands written in natural language. In our system, verbal primitives can be coded as events, states, and actions (Brackenbury et al., 2019; Huang and Cakmak, 2015). The system is composed of a web app for the upload of vocabularies and the following creation of trigger-action rules with a form-based compositional interaction strategy. The created rules can then be used to customize the behavior of one or more interfaces. In our case scenario, the interfaces are represented by *SMARTER* (Andrao et al., 2023b) and *SmartGame* (Gini et al., 2023).

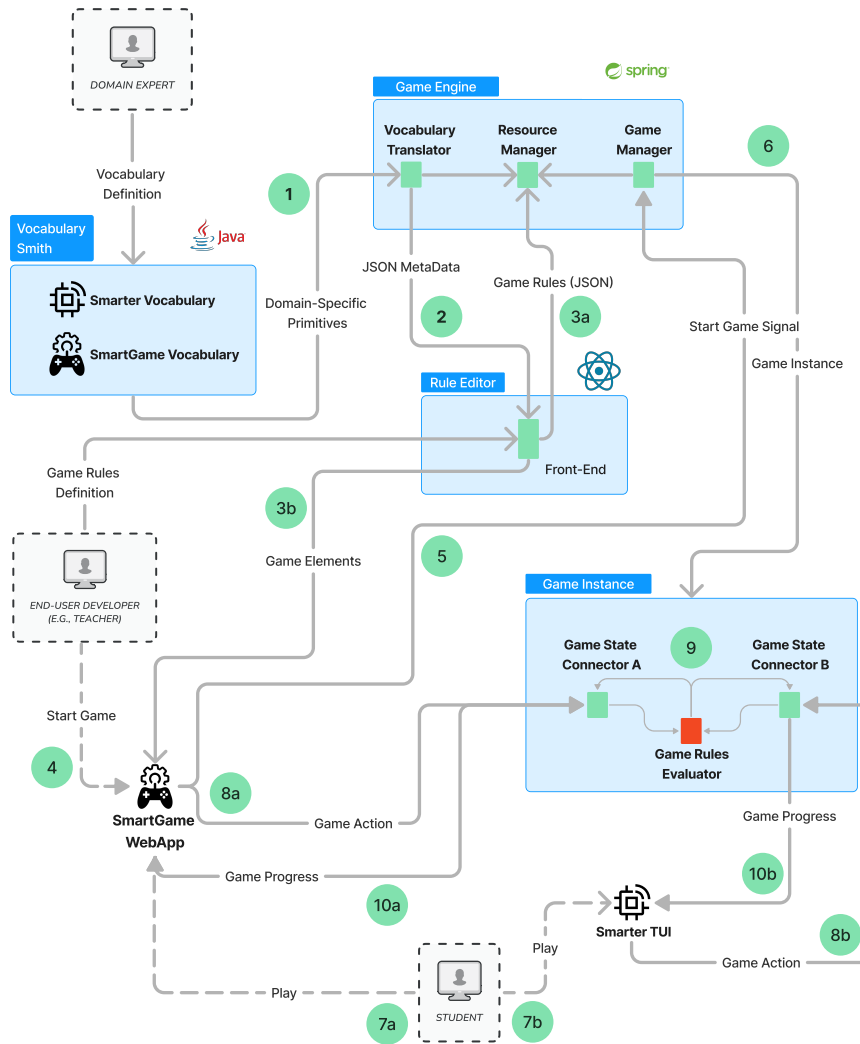


Figure 7.3: RuleCraft's Architecture and Flow.

The **Vocabulary Smith** component, built in Java, is the cornerstone of the process for defining and refining vocabularies. Beyond the standard functionalities of a simple vocabulary editor, it translates these vocabularies into a set of verbal primitives (1), which are required to define the rules and game elements for specific learning tasks. These verbal primitives then find their way to the heart of the system, the Java-based **Game Engine** component, built using the *Spring* framework. Here, the primitives are transformed into JSON metadata format by the Vocabulary Translator (2). This is a crucial step, as it paves the way for the subsequent stages of rule definition and editing. This formatted metadata serves as the backbone for the Rule Editor component, an intuitive frontend interface for end-user developers (i.e., teachers) implemented using *React*. There, users can define rules (3a) using predefined syntax forms and constraints exploiting the available primitives and the respective game elements visualized in the final application (3b – which is the *SmartGame* web app, implemented using *React*). Once the rules are carefully defined, they find their home in the **Resource Manager** component. At the same time, the corresponding game elements are provided to the application, ready to be used by students.

Teachers leverage the *SmartGame* web app to initiate and configure learning tasks, defining the classes and tangible devices (*SMARTER*) to be utilized. They can signal the *start of the game* to the **Game Manager** (4) and (5) bringing a new *game instance* to life (6), inviting students to engage actively. Whether through the *SmartGame* web app interface (7a), or directly interacting with *SMARTER* (7b), students delve into the learning experience. *Game actions*, whether initiated through the web app (8a) or the tangible devices (8b), are swiftly intercepted by dedicated **Game State Connectors**. These connectors play a vital role in ensuring adherence to game rules checked by the **Games Rule Evaluator** component while facilitating player status progression. Upon rule evaluation (9), this provides real-time feedback signals, such as illuminating green or red lights (10b), to indicate student progress or game advancement in the web app (10a).

7.2 LLM-Based Iterative Disambiguation Process: A Pilot Study

Based on the results outlined in Chapter 6, we focused on the idea of creating an iterative process based on generated questions for disambiguating natural language ambiguities to explore how an LLM – in this specific case, ChatGPT-4o¹ – can guide the def-

¹The latest available version in January 2025.

Table 7.2: Participant demographics and diagnostic information, including intellectual disability ("ID") level and verbal production ("verbal") capacity.

id	age	gen.	diagnosis	ID	IQ	verbal
P1	23	M	Partial fetal alcohol syndrome, symptoms of ADHD	mild	< 60	good
P2	24	M	Autism spectrum (level 2)	mild	< 61	good
P3	19	F	Down syndrome, oppositional defiant behaviors	severe	< 35	low
P4	18	F	Partial trisomy of chromosome 18	severe	< 35	moderate

initiation of trigger-action rules by participants. We developed a prototype LLM-based system to conduct a preliminary study involving caregiver-adult pairs with intellectual disabilities.

This pilot study explored the advantages and limitations of an LLM-based interface in supporting people with intellectual disability in defining instructions for home automation. This study was conducted in January 2025 and received approval from the ethics committee of *Polytechnic University of Milan*.

7.2.1 Participants

We recruited four couples composed of one care recipient with intellectual disability and one caregiver. The recruiting occurred through snowball sampling. The four participants with intellectual disability – two males and two females – were aged between 18 and 24 ($M = 21.00$, $SD = 2.94$ years).

All participants had a diagnosis for a specific genetic or medical syndrome associated with intellectual disability (see Table 7.2). Two had a mild intellectual disability – associated with an IQ between 51 and 70 and slight difficulties in adaptive functioning – and two had severe intellectual disability – IQ between 20 and 35 and severe difficulties in adaptive functioning. One of the participants had no experience with home automation systems, while three of them owned some devices related to home automation (voice assistants, lights, intercom) and used them to some extent (1 = "almost never", 1 = "sometimes", 1 = "daily"). They were all from the surrounding area of Padova, in the northern of Italy. The caregivers involved were the parents of the care recipients (in the text, we refer to them as C1, C2, C3, and C4).

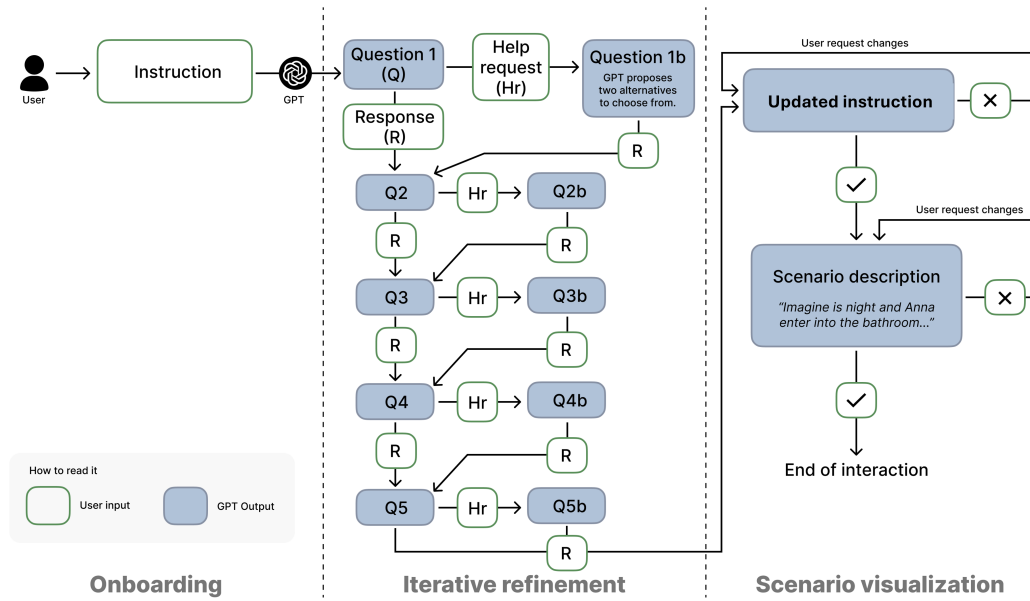


Figure 7.4: Interaction flow of a round of interaction with the interface.

7.2.2 Materials

Prototype

This study used a prototype chat-based interface designed to guide users in creating trigger-action-like instructions for control home automation system. The prototype interface for this study was developed using *OpenAI* GPT functionalities, enabling rapid customization for specific purposes. The custom GPT underwent iterative testing by two researchers to ensure consistent adherence to a three-step interaction process (see Figure 7.4):

1. **Step 1: Onboarding.** In this step, the interface awaits the user's **input** of an instruction. If the input is unrelated to home automation, the interface specifies that it is necessary to write an instruction, even if it is incomplete.
2. **Step 2: Iterative refinement.** Upon receiving an initial instruction or an idea for a home automation task, the interface identifies ambiguities and poses up to **five sequential questions** to refine the instruction. Questions are presented one at a time to avoid overwhelming the user. After refinement, the system offers an updated instruction and seeks user confirmation on its adequacy.
3. **Step 3: Scenario visualization.** Once the user confirms that the instruction is

satisfactory, the interface generates a **hypothetical scenario** illustrating how the instruction would function in practice, presented as storytelling. For example,

"Imagine it's a cold winter day. The temperature in the house drops to 17C. Without anyone having to do anything, the smart fireplace automatically turns on, warming up the room. After a while, the temperature rises and exceeds 22C. At this point, the fireplace turns off by itself, preventing house from overheating".

After describing the scenario, the interface awaits the user's feedback. If the user is satisfied, the round of interaction with the interface concludes. Otherwise, the interface prompts the user to specify modifications they would like to make.

User-Experience Questionnaire - Short Version (UEQ-S)

UEQ-S is a standardized questionnaire to assess the user experience (Schrepp et al., 2017). This short form consists of eight items, each represented by pairs of opposing adjectives (e.g., *"confusing"* – *"clear"*), positioned at the extremes of a seven-point scale ranging from *"-3"* to *"+3"*. The questionnaire comprises two scales that measure the following aspects:

1. The **Pragmatic Quality Scale** evaluates the effectiveness and efficiency of the product (four items, e.g., *"useful"* – *"useless"*).
2. The **Hedonic Quality Scale** assesses the appeal and attractiveness of the design (four items, e.g., *"engaging"* – *"boring"*).

In addition to providing an *"Overall"* user experience score, the questionnaire allows for separate evaluations of the individual scales that compose it.

7.2.3 Task and Procedure

The study was organized as semi-structured interviews conducted remotely over two weeks using the interface via *Zoom*. In addition to a caregiver-care recipient couple, two researchers participated in each session: one acting as a moderator and the other as a supervisor responsible for data collection. Each session lasted approximately 90 minutes, with participants given the freedom to take necessary breaks.

Before starting, the procedure was carefully explained to the participants, who gave their consent to take part in the study. Participants were informed that their participation was totally voluntary and that they could interrupt at any time by withdrawing their consent.

Interacting with the Prototype

The moderator read the instructions that detailed the process and tasks to be performed by the participants. Participants were asked to create a home automation instruction following a similar methodology to the one proposed in Study 7: after watching a short video depicting a home automation scenario, participants were asked to compose an instruction for a home automation system to execute the scenario presented. The first interaction for everyone was based on the *s1* video-scenario described in Study 7.

Since the sessions took place remotely, the moderator shared the screen to display the interface prototype. Participants were given a choice to either type their instruction in the chat – which the moderator would then copy and paste into the interface – or dictate the instruction aloud for the moderator to transcribe verbatim into the input field of the interface.

After submitting the instructions, participants were invited to continue interacting with the interface, providing any additional prompts or responses required by the interface. Upon reaching the final step of the interaction, participants were given the option to start another interaction round by watching another video scenario (among those used and described in Study 7), self-defining an instruction to input into the interface, or ending the study. After completing each interaction round, participants were asked to provide feedback on the interaction, responding to three pre-defined questions concerning the usefulness, clarity, and amount of questions generated by the interface.

User's and Caregiver's Questionnaires

At the end of the entire interaction session, participants were asked to answer some questions about anagraphic information, experience with home automation systems, and programming languages. Answers were collected through *Qualtrics*. Following, we asked caregivers to fill out the UEQ-S (Schrepp et al., 2017), implemented in *Qualtrics*, and to answer some open-ended questions to gather caregiver feedback about the prototypal interface. Questions covered both positive and negative aspects of the interface, as well as future expectations about LLM-based systems and their possible impact in home automation settings and in supporting people with intellectual disabilities.

7.2.4 Data Analysis

In addition to user-generated chat transcription from the prototype, the sessions were audio-recorded. Two researchers independently analyzed the transcribed recordings with an inductive thematic content analysis (Braun and Clarke, 2006). For analyzing the user-generated chat, a deductive approach was employed to classify the questions starting from our previous results Study 6 and Study 7. The caregiver responses to the UEQ-S were analyzed using descriptive statistics and benchmark comparisons (Schrepp et al., 2017).

Engagement levels were assessed based on the number of rounds of interactions participants performed with the interface after the first video scenario.

7.2.5 Results

LLM-based interface performance

All participants completed at least 1 round (maximum 4 rounds), producing an improved version of the original instruction. Due to technical problems, round 4 of P1 was not completed and was excluded from the analysis. All initial and revised instructions are provided in Appendix M. Among all participants, we analyzed the performance of the LLM-based prototype in 11 rounds of interaction. For each interaction round, the interface generated a minimum of 2 to a maximum of 5 clarification questions ($M = 3.90$, $SD = 1.29$) for a total of 39 questions (see Table 7.3).

Table 7.3: The frequency table shows the 39 questions generated distribution among participants and interaction rounds.

id	round 1	round 2	round 3	round 4
P1	2	3	5	-
P2	2	5	5	5
P3	5	3	-	-
P4	4	-	-	-

Following the classification emerged from our prior studies (Study 6 and Study 7), among the 39 questions analyzed, 24 fall within the *Content Extension* category. These questions asked the user to specify the "Sensor" to be used, the "Action" to perform, or "Delay" times. Ten (10) questions were categorized under *Outcome* ambiguity, where

the system seeks clarification on "*End of Actions*" to define if the action should be repeated indefinitely or interrupted when specific conditions are met. Four (4) questions were related to *Language* ambiguity, where the interface requests clarification of "*Figurative values*". Finally, 1 question falls into the *System specification* category, with the interface asking about the specific brand of smart plug available in the household. No new ambiguity-related categories emerged from any generated question.

User Engagement and Feedback

Engagement levels were assessed based on the number of rounds of interactions participants performed. Different levels of engagement with the interface were observed among participants. Both P1 and P2 engaged fully with the interface without hesitation after completing their first round of interaction. P1 proposed self-defined instructions from the second round, inspired by personal experience. For example, "*When I get up to go to the center, Alexa should warm the milk*". This idea, discussed with the caregiver, prompted P1 to reflect on how the system could optimize daily routines and address personal needs.

In cases of P3 and P4, they decided to interrupt the interaction after the second and the first rounds, respectively. In the case of P3, the participant expressed a lack of interest in continuing the activity as she was tired from the task saying that we could have proceeded with other instructions the following day. P4 was not interested in the interaction, so immediately after the first round, she preferred to interrupt the activity.

After the first interaction round, participants were asked about their opinions regarding the interface's questions. P1, P2, and P3 found the questions understandable and valuable. P2 noted the questions were very easy and expected a higher amount of questions. P4 preferred not to answer.

Questions Understanding. Analyzing the verbal reports, we observed different levels of understanding of clarification questions generated by the interface. P1, P2, and P4, while requiring active support from caregivers to understand some of the questions fully, generally grasped the overall requests posed by the interface.

In contrast, P3 required substantial assistance from the caregiver to comprehend the questions and formulate responses. These responses often consisted of single words or concepts that closely echoed the caregiver's input. The caregiver later explained that tasks that encompass the definition of time duration are challenging for P3 and re-

ported that she then typically responds to questions by choosing the last of the options provided.

Caregivers Feedback and Responses

Caregivers played an active, supportive role during activities, often reading the interface's questions aloud to prompt participant reflection. An exception was observed with participant P4, where C4 took a more substitutional role, directly providing answers to the interface on the participant's behalf or even pushing the participant to change the response (e.g., "*No, no, let's do the first one it suggested instead*"). It was observed that all caregivers used *concrete examples* related to the participants' personal experiences to encourage reflection or simplify the understanding of the tasks to be performed with the interface. In particular, caregivers leveraged participants' prior experiences with systems like *Alexa*, referencing these interactions to help them understand the type of instructions expected by the interface.

UEQ-S results. Analyses of results from the UEQ-S questionnaire show an average user experience score of 1.41 ($SD = 1.53$), placing it in the "good" range² (see Appendix N for details on UEQ-S scores). The Pragmatic Quality Scales fall in the "below average" range of the UEQ benchmark ($M = 0.81$, $SD = 2.18$), highlighting some issues in the practical usability of the prototype, specifically related to efficiency (*item 3*) and ease of use (*item 2*). On the other hand, the Hedonic Quality Scale scored in the "excellent" range ($M = 2.00$, $SD = 0.94$). Specifically, the level of interest (*item 6*) and novelty (*item 8*) attributed to the system were both 2.50 on average.

Preliminary Themes Emerged. From qualitative analysis of caregivers' responses to open-ended questions, insights into the interface's interaction design and user experience emerged.

- *Automation vs. Autonomy.* The LLM-based system has been recognized as a promising tool for improving autonomy, particularly in co-living scenarios or if caregivers can no longer provide direct care. Despite its potential, caregivers have expressed concern that excessive automation may undermine the development of essential problem-solving skills. C2 brought the example of using a calculator in school to support students, which occurs only after they have developed basic nu-

²Above +1.50 is considered "excellent", between +0.80 and +1.50 "good", between 0.00 and +0.80 "below average", between 0.00 and -0.80 "poor", and below -0.08 "very bad" (Schrepp et al., 2017).

meracy skills. They emphasize the need for a balanced approach that empowers users without fostering over-reliance on technology.

- *Individuals Differences.* Caregivers of participants with mild intellectual disabilities confirmed that the questions, vocabulary, and system interactions were accessible and valid despite some difficulties. Caregivers of participants with severe intellectual disabilities (P3 and P4) reported that the tasks and questions were too complex for independent use. They suggested redesigning the system as a collaborative tool in which caregivers and participants work together or as a moderate setup in which caregivers assist in defining instructions, while the LLM-based system supports participants in performing daily routines awith voice prompts for a guided but autonomous experience.
- *Visual and Concrete Clues.* C3 and C4 emphasized the importance of visual aids and limited concrete options to support decision-making. For example, C4 noted that open-ended questions posed significant challenges while offering two simple alternatives for decision-making was more manageable and understandable. C2, C3, and C4 also pointed out that incorporating visual cues alongside the questions could greatly improve the accessibility of the system, enhance comprehension, and support independent responses.
- *Input-output Modalily.* Users' interactions with the interface could be conducted by writing instructions or expressing them vocally. It was observed that, in all cases, participants chose to interact verbally. This preference was driven by the convenience of speaking, as noted by C4 who stated "*It is easier to do it verbally*". C2 stated that for P2, interaction through writing is normally easier than verbal interaction. Unfortunately, the setting of the online study posed some difficulties. Although participants could type their responses, P2 struggled to type them because he was used to a traditional keyboard and, in contrast, they were using a tablet during the study.

7.2.6 Discussion

Joint interviews with caregivers and users, together with observations of their interaction with the prototype, show that an iterative process, guided by LLM-generated disambiguation questions can be effective in supporting users with intellectual disability in generating home automation instructions. The questions generated by the

LLM can be categorized under specific ambiguities previously identified in Study 6 and Study 7, suggesting that the LLM can detect some ambiguities in the user prompt.

For what concerns the LLM responses, the iterative process resulted in refined prompts that were improved compared to the initial one, often overly abstract and ambiguous. At the end of each round of interaction, users confirmed that they were satisfied with the refined instruction.

Nevertheless, our observations and direct feedback from the participants identify some aspects of the LLM-based interface that can inform and guide the design of future tools for home automation.

For example, the individual differences and multi-modality approach, currently under investigation by other related studies (Artizzu et al., 2024; Barricelli et al., 2022; Calò and De Russis, 2024; Monge Roffarello and De Russis, 2023), could represent an important research avenue for future studies in this field. Also, using more concrete content (e.g., providing examples) can help individuals during the interaction, in line with what we found in Study 5 and related to the *EUDability* constructs (Barricelli et al., 2023; Barricelli et al., 2024).

Limitation

This preliminary pilot study has several limitations. First, the findings are derived from a small sample of only four caregiver–care recipient pairs, limiting the generalizability of the results. Additionally, a significant constraint is the online setting of the study, which considerably restricted the possibility of observing users’ natural interactions with the system in real-world settings.

Chapter 8

General Discussions

In this Chapter, we pull together the threads of what we found with our studies (presented in Chapter 5 and Chapter 6) and how we answered the research questions we addressed (section 8.1). In the second part of the Chapter (section 8.2), we highlight the implications for the design of TAP-based EUD systems.

8.1 Retrospective on Objectives

The three studies on language and mental models described in Chapter 5 (Study 1, Study 2, and Study 3) highlighted that some conjunctions, verbal structures, and some orders of the elements in the rule can better support non-programmers in mentally representing and identifying temporal differences between states and events in trigger-action rules.

The four studies described in Chapter 6 highlighted some practical guidelines to support non-programmers in dealing with TAP through two different approaches: form-based compositional (Study 4, Study 5) and LLM-based (Study 6, Study 7).

In this dissertation, we addressed seven research questions that guided the design of our studies. In the following subsections, we will analyze each research question.

8.1.1 Addressing RQ1: Language for Distinguishing States and Events

RQ1: *Do specific language features facilitate the creation of effective and distinct mental models for states and events in non-programmer users during the composition and interpretation of trigger-action rules?*

Regarding our first research question, we found that certain language aspects can ei-

ther facilitate or hinder non-programmers in mentally representing states and events.

Specifically, we observed that some conjunctions and verb forms are more naturally associated with either events or states during the composition of trigger-action rules (Study 1). Also, we observed that some of these linguistic features are more effective than others in supporting the identification of states and events (Study 2).

These results align with the findings of (Huang and Cakmak, 2015) and echo the efforts observed in the literature (e.g., Brackenbury et al., 2019; Manca et al., 2019; Zancanaro et al., 2022) and in commercial tools (e.g., *SharpTools*; *SmartRules*) to distinguish between states and events at the front-end and back-end levels.

8.1.2 Addressing RQ2: Conjunctions and Verbal Structures

RQ2. *Which specific temporal conjunctions and verbal structures are naturally associated with the definition of states and events?*

In Study 1 and Study 2, we found that temporally specific conjunctions, such as “*as soon as*” for events and “*while*” or “*as long as*” for states, facilitated more accurate mental representations of states and events. Additionally, for events, participants preferred a redundant verb form emphasizing the instantaneousness of the action. Conversely, the conjunctions “*if*” and “*when*” – although naturally preferred by participants – were found to ambiguously describe both states and events, in particular when presented with the present tense.

These results partially confirm previous findings (e.g., Desolda et al., 2021a; Ghiani et al., 2017; Zancanaro et al., 2022) on the role of conjunctions in non-programmers’ mental representation of states and events, as well as on the potential impact of the verb form used (Manca et al., 2019). However, our studies reveal that using “*if*” and “*when*” does not support non-programmers in accurately identifying and representing states and events. Consequently, employing these conjunctions could lead to greater ambiguity and misunderstanding during EUD tasks by non-programmers.

8.1.3 Addressing RQ3: Order of Elements

RQ3. *Is there an elements’ order naturally used by individuals that can facilitate them in composing and mentally representing trigger-action rules?*

From Study 1 and Study 3, we observed that non-programmers generally prefer to place the trigger part at the beginning of the rule when composing trigger-action rules. This result can be interpreted as a manifestation of the *Iconicity assumption*, whereby

individuals are facilitated in mentally representing elements in their actual temporal sequential order (Givón, 1992). The trigger-first order has been widely used in the literature (e.g., Bellucci et al., 2019; Brich et al., 2017; Dey et al., 2006) and in commercial products (e.g., *AutomateIt*; *IFTTT*; *SmartThings*), and our results strongly support the adoption of this order. For complex rules – where the trigger involves both an event and a state – our findings show that the naturally preferred sequence is typically *event-state-action*. Although this order violates the *iconicity assumption*, it could be explained by the *Figure-Ground theory* (De Vega et al., 2007). This order has also been employed in solutions proposed in literature (e.g., Ardito et al., 2018; Brackenbury et al., 2019; García-Herranz et al., 2010; Zancanaro et al., 2022) and is implemented in some commercial solutions (e.g., *Assistant*; *SharpTools*; *SmartRules*).

8.1.4 Addressing RQ4: Mental Models during Interaction

RQ4. *What are the characteristics of users' mental models when interacting with a "constrained" TAP-based EUD system?*

We addressed our fourth research question by studying a specific use case in the field of education. In Study 4 and Study 5, we examined teachers' mental models while interacting with a form-based compositional visual interface to define trigger-action rules.

Regarding working mental models, we observed that our participants relied on common patterns for mentally representing concepts, allowing us to identify three distinct metaphors, some more effective than others. The most common approach was based on a representation of trigger-action programming as step-by-step (the "command" metaphor), which we observed to be the least effective. Additionally, we observed that, in some cases, drawing on personal expertise to conceptualize events and states (the "instruct-child" metaphor) helped some participants to adopt TAP logic ("react" metaphor). Furthermore, we identified two conceptual models of the system (e.g., "smart" vs. "mindless" computer), which influenced participants' reasoning.

These findings contribute to the literature on mental models, aligning with prior findings on mental models (Fischer, 1991; Fischer, 2001; Norman, 1986; Norman, 1983) and on the role of expertise-related scenarios in shaping strategies for EUD tasks (Wiedenbeck, 2005).

8.1.5 Addressing RQ5: Abstract and Concrete Descriptions

RQ5. *Do the linguistic features (abstract vs. concrete descriptions) implemented in a compositional TAP-based interface influence the assumption of different reasoning strategies leading to more effective mental models during EUD tasks?*

In Study 5, we investigated our fifth research question by analyzing whether abstract vs. concrete descriptions implemented in a TAP-based platform, impact teachers' reasoning strategies and mental models' assumptions.

We observed that the abstract version was harder to use but encouraged broader strategy exploration, improving performance in later tasks (supporting participants to assume working mental models that support TAP logic). In contrast, starting with the concrete version was easier (Barricelli et al., 2023) – as it aligned more with working mental models assumed by our participants – but led to a rigid reasoning strategy and mental model adoption, making it harder to transition to others that might be more appropriate. These results align with studies on procedural learning and mental models (Eriksson and Jansson, 2016; Johnson-Laird, 1983; Johnson-Laird, 2010; Xie et al., 2019).

8.1.6 Addressing RQ6: Natural Language Ambiguities

RQ6. *Which language ambiguities are identified and addressed by an LLM while interpreting trigger-action rules written in natural language?*

The results from Study 6 and Study 7 establish a baseline for ambiguity categories identified by LLMs, such as ChatGPT, when interpreting natural language trigger-action rules. One of the primary sources of ambiguity stems from language vagueness, as natural language lacks the precision of code, which operationalizes abstract concepts (Monge Roffarello and De Russis, 2023; Scoccia, 2023). The other is related to ambiguity in trigger descriptions, specifically on states, which often lack details. Despite these ambiguities, in Study 6, ChatGPT-4 generated syntactically accurate and complete code. However, our results from Study 7 suggested that the correctness of code could decrease when the level of ambiguity rises ¹ potentially affecting safety and security (Breve et al., 2023).

An important consideration emerged from the fact ChatGPT-4 arbitrarily resolves ambiguities in codes without explicit user input, raising concerns about its alignment with users' intentions.

¹We did not perform any comparison between Study 6 and Study 7 as the study designs have clear differences.

8.1.7 Addressing RQ7: Intent Recognition and Disambiguating Questions

RQ7. *Which disambiguation questions can an LLM system generate to align with user intent?*

In Study 7 (and in the pilot study presented in section 7.2), we examined the question-generation capabilities of ChatGPT envisioning an iterative disambiguation process with the user. This strategy could enhance the reliability and safety of the generated code while ensuring more alignment with user intent (King et al., 2024; Rivkin et al., 2023).

We observed that the generated questions often reflected similar themes to those identified for ambiguities. The majority of these questions aimed to clarify details regarding the state, while just a low percentage was related to language usage. Indeed, language-related ambiguities were frequently addressed arbitrarily by ChatGPT, as noted in Study 6.

Moreover, most of the generated questions were useless, not aiming at resolving the identified ambiguities nor addressing the discrepancies between the generated code and the participant’s intention.

8.2 Implications for the Design of End-User Development Tools

At a practical level, the findings of this dissertation suggest some design guidelines for developing TAP-based systems that effectively support non-programmers in handling trigger-action rules by avoiding common bugs related to language, such as those derived from temporal ambiguity of occurrences (Brackenbury et al., 2019; Huang and Cakmak, 2015) and the level of abstractness (Barricelli et al., 2023; Barricelli et al., 2024).

8.2.1 Mapping Linguistic Cues in Systems’ Ontology

The choice between event and state eventually depends on the type of sensor used: for example, a proximity sensor on a door leads more naturally to a state (e.g., the door is closed if the sensor is activated), while an accelerometer may lead more naturally to an event (e.g., the door is moving so it is easier to derive that it is opening or closing). Nevertheless, it is usually possible to process the sensor signals to derive the sequence of “starting events” and “closing events” for each state to turn a state-preferred sensor into an event-based structure, vice-versa, an event-preferred sensor can be turned into a state-based structure by keeping track of the state duration between the start and the

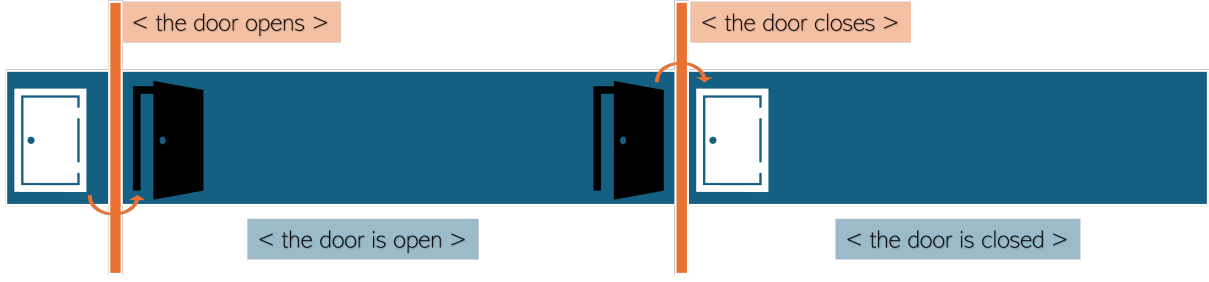


Figure 8.1: The figure graphically represents the distinction and the relation between **states** ("the door is open" and "the door is closed") and **events** ("the door opens" and "the door closes").

end events (see Figure 8.1).

As mentioned in previous work (Debnath et al., 2022; Desolda et al., 2021a; Gallo and Paterno, 2022; Gallo et al., 2024; Manca et al., 2019), aligning the system's ontology (the representation of the domain) to fully and properly represent the distinction between events and states is important. Our results go further by suggesting the link to the final user: as we have seen, explicitly mapping the system representations to lexical aspects could effectively support end-users in trigger-action programming by disambiguating the temporal aspects of triggers. Specifically, in the internal representation of the system, **conjunctions** and the **type of verb** can support the distinct representation of states and events, as well as the identification of **ambiguous triggers** to be notified or negotiated with users.

Although the suggestion to use specific linguistic clues has already been discussed in the literature (e.g., Huang and Cakmak, 2015; Manca et al., 2019; Zancanaro et al., 2022), we can provide more articulated guidelines by suggesting to use the "best" conjunction with respect to the verb used to support users in distinguish occurrences with different temporal features. Our results suggest considering three possible ways to map specific linguistic aspects to system events and states derived from sensors (see Table 8.1 and Figure 8.2):

I. Linguistic expressions identified as an **event** description, which describes instantaneous occurrences. The system should identify the phrase as an event when:

- the conjunction is *event-specific* (e.g., "as soon as") regardless of the verb (which can be either generic or event-specific);
- the conjunction is *generic* (supporting both states and events, e.g., "if" or "when") or not specified (*NA*), but the verb is *event-specific* (e.g., "it starts to rain").

Table 8.1: Summary of the most adequate linguistic expression to identify triggers that represent events or states.

occurrence	conjunction	verb
event	<i>as soon as</i>	it starts to/ present tense
	<i>when / if / NA</i>	it starts to
state	<i>while / as long as</i>	progressive form/ present tense
	<i>when / if / NA</i>	progressive form
ambiguous	<i>as soon as</i>	progressive form
	<i>while / as long as</i>	it starts to
	<i>when / if / NA</i>	present tense

The former is given by the fact that phrases introduced by "*as soon as*" effectively supported our non-programmers in accurately identifying events (more than 90% of the time; see Chapter 5). The latter derives from our results which highlight individuals' preference for generic conjunctions (e.g., "*when*") for describing events despite these conjunctions – associated with a generic verb – do not support people in identifying events (less than 50% of the time).

II. Linguistic expression identified as a **state**, which describes prolonged occurrences. The system should identify the phrase as a state when:

- the conjunction is *state-specific* (e.g., "*while*" or "*as long as*") regardless of the verb (which can be either generic or state-specific);
- the conjunction is *generic* (supporting both states and events, e.g., "*if*" or "*when*") or is not specified (*NA*), but the verb is *state-specific* (e.g., "*it is raining*").

The former is given by the fact that phrases introduced by "*while*" and "*as long as*" effectively supported our individuals in accurately identifying states (more than 90% of the time; see Chapter 5). The latter derives from our results which highlight individuals' interpretation of phrases introduced by "*when*" and "*if*" as descriptions of both states and events when associated with a generic verb.

III. Linguistic expressions identified as **ambiguous**, which is general or with lexical inconsistencies. The system should identify the phrase as ambiguous when:

- the conjunction is generic (e.g., "*if*" or "*when*") or it is not specified and the verb form is generic (i.e., present tense);

- the conjunction is *occurrence-specific* and the verb specific but not congruent (e.g., the event-specific conjunction "*as soon as*" with the state-specific verb form "*it is raining*").

8.2.2 Form-Based Compositional Visual Interface to Guide Users in Composing Trigger-Action Rules

Linguistic Primitives

From the point of view of design, one approach is to guide people to use the most appropriate conjunction and verb form to effectively communicate triggers. In this respect, **form-based compositional visual interfaces** (e.g., *IFTTT*) may constrain the user action to **avoid the possibility of composing a temporally ambiguous trigger**. In this respect, the use of generic conjunctions (e.g., "*if*") should be avoided, as neither preferred nor supportive for accurate occurrence identification. Also the use of "*when*" should be discouraged as it may not adequately represent the temporal features of the triggering occurrence, leading to ambiguity and errors in rules definitions. It would be preferable to use "*as soon as*" for introducing events and "*while*" for introducing states, as they effectively help users in the accurate identification of occurrences' temporal features (as we did in *RuleCraft*; section 7.1). Also, systems design should consider providing redundant verbal forms specific to describe states and events, as they are perceived as a natural association by users and can highlight their distinction.

Alternatively, if the EUD tool does not apply strict constraints in rule composition at the interface level (e.g., it does not explicitly separate states and events), it is important to notify the user of the possible presence of ambiguous interpretation in the rules, supporting them in the debugging phase (as suggested by other relevant work in the field, e.g., De Russis and Monge Roffarello, 2018; Manca et al., 2019).

Considering the **order of elements** by which users should be guided in composition we have seen that **trigger-first** is more natural (Study 1; Study 3). Even if supporting users to think about states seems to be a good approach to foster effective strategies of rule composition (Study 4), it is preferable to employ a flow of composition in which users can compose the first the event, then the state(s), and finally the action(s). However we saw some differences in composition vs. interpretation preference. Suggesting that future works should deeply investigate possible differences.

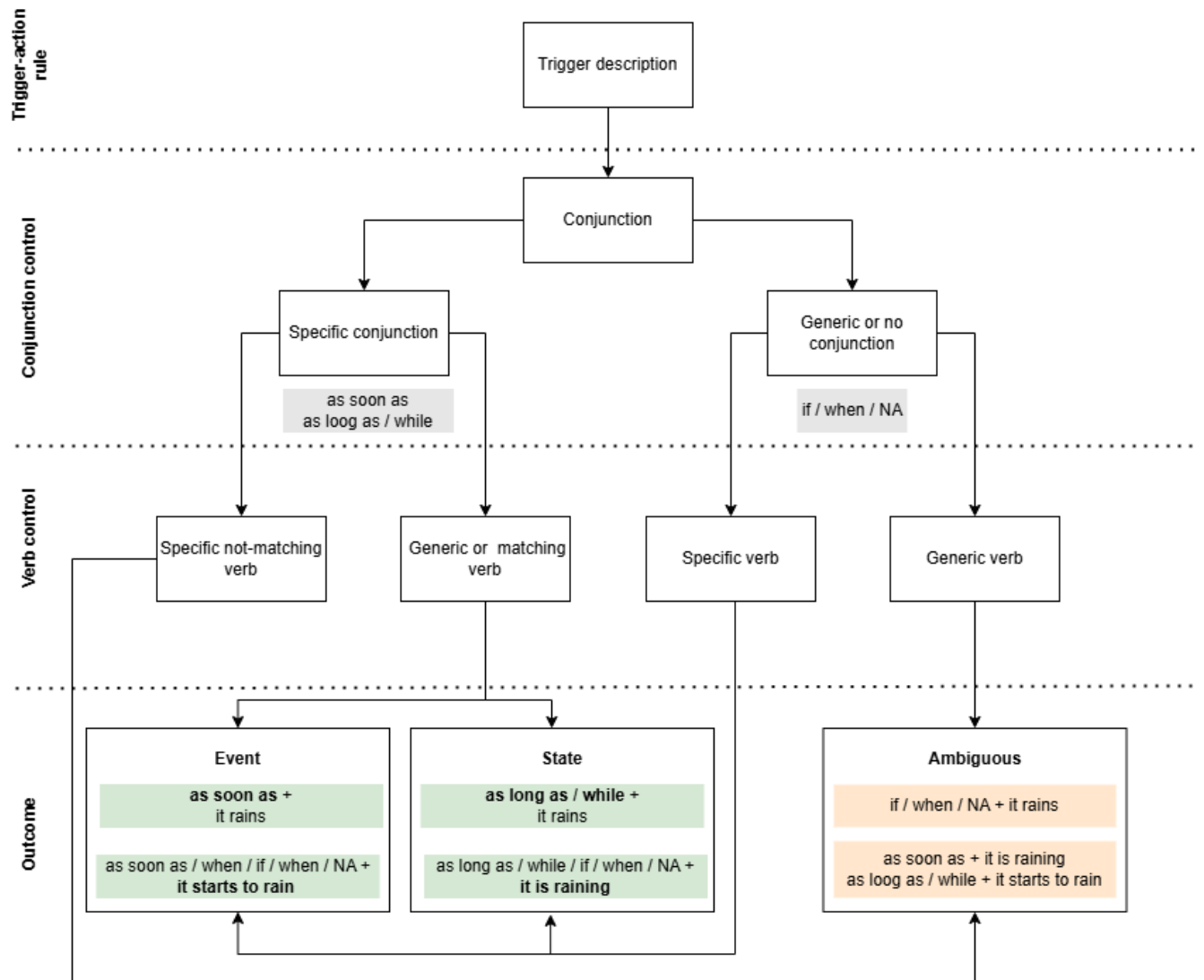


Figure 8.2: The figure represents how linguistic clues can be exploited and mapped into the system ontology.

Concreteness and Abstractness of Primitives

We observed that the level of abstraction of primitives can be a problem in compositional visual interfaces (Chapter 6). Verbal primitives overly abstract are difficult and distant from users' mental models, making it preferable to have more concrete examples (Barricelli et al., 2023; Barricelli et al., 2024). On the other hand, abstract content can support users in questioning their approach getting closer to trigger-action logic (aligning with findings in human reasoning Markovits, 1988).

A solution is to design effective tutorials so that the conceptual models of designers and the mental models of users can be more aligned in the meaning of abstract concepts (Fischer, 2017; Norman, 1986). We propose the following lessons learned to ground effective training for TAP-based EUD:

1. Offering concrete primitives may make the first impact with EUD easier, yet it would probably not facilitate further adoption of an abstract reasoning style; since concrete primitives better support the "command" metaphor (at least in teachers), they would not even facilitate the acquisition of an event-based logic.
2. The "command" metaphor is predominant and needs to be addressed as it might constitute a wrong mental model for TAP.
3. While the "react" metaphor is the closest to event-driven programming, a comparison with concrete-real life action (e.g., for teachers, the way children are instructed on educational tasks – the "instruct" metaphor) might be more effective for domain-expert to facilitate the adoption of a more optimal mental model;
4. Attention should be paid during complex exercises to avoid reverting to ineffective metaphors (specifically the "command" metaphor): it is worth remembering that, although concrete primitives may inappropriately foster the "command" metaphor, the abstract primitives, being possibly too complex, may inadvertently support the command metaphor too;
5. It could be useful to teach users what the system can and cannot do in advance so that they can switch their focus from "learning how to use the system" to "learning how to use the primitives".

One solution is to have systems that, at first, offer concrete examples but allow abstractions at some point in the interaction. Future studies should study other combinations and levels of abstract and concrete primitives, adopting the *EUDability* frame-

work (Barricelli et al., 2023; Barricelli et al., 2024), to understand how to guide users from reasoning based on concrete examples to more abstract ones.

8.2.3 LLM-based System for Handling Ambiguities in Trigger-Action Rules

An alternative to compositional EUD systems is the use of **LLMs to guide users** in the definition of trigger-action rules (e.g., Gallo and Paterno, 2022; Gallo et al., 2024).

Controlling Temporal Ambiguities

These systems should control conjunction and verb features for extracting temporal indications from the rules created and communicate to the users the presence of ambiguous phrases to negotiate with them and undercover the temporal aspects of the occurrences described. In the internal representation of the system, starting from the trigger description (e.g., *"when it starts to rain"*), we suggest including two layers to reach the system's outcome definition (see Figure 8.2):

- **Conjunction control:** identification of the conjunction, which can be "occurrence-specific" (e.g., event-specific, such as *"as soon as"*, or state-specific, like *"while"*) or "generic" (not specifically associated with a class of occurrences, e.g. *"if"*).
- **Verb control:** processing of the verb form, subsequent to the first step. When:
 - an occurrence-specific conjunction is associated with a matching specific verbal form (e.g., *"as soon as it starts to rain"* or *"while it is raining"*) or a generic verbal form (e.g., *"as soon as it rains"* or *"as long as it rains"*), then the phrase is identified with an event or a state;
 - an occurrence-specific conjunction is associated with a non- matching specific-verb form (e.g., *"as soon as it is raining"* or *"as long as it starts to rain"*) then the phrase is identified as ambiguous and it requires notification or negotiation with the user (according to the type of system compositive vs. generative) to undercover temporal aspects of the trigger;
 - a generic conjunction (or no conjunction) is associated with an occurrence-specific verb form, then the phrase is identified as an event or a state depending on the verbal form (e.g., *"if it starts to rain"* or *"when it is raining"*);
 - a generic conjunction (or no conjunction) is associated with a generic verb (e.g., *"if it rains"*) then the phrase is identified as ambiguous and requires no-

tification or negotiation with the user (according to the type of system component vs. generative) to uncover temporal aspects of the trigger.

A long-term goal is to create a system whereby rules written in natural language (thus increasing the expressiveness power of users) are interpreted and returned as trigger-action rules to users (e.g., similar to the system presented by Gallo et al., 2024) using the linguistic cues that in our studies we have seen to be most effective for creating a mental distinct representation of states and events.

Detecting and Disambiguate Ambiguities

The results of Study 6 and Study 7 suggested some clues to design LLM-based EUD systems that, through an iterative process, disambiguate the trigger-action rules to align more closely with the user’s intent. From a practical point of view, as we started doing in the Pilot Study described in section 7.2, we propose a three-step structure that could be an effective approach to maximize model performance and address current limitations through an iterative and guided process:

1. The first phase could consist of an **onboarding** process to assist users in formulating the basic prompt. This phase could support multi-modal input (Artizzu et al., 2024; Barricelli et al., 2022; Calò and De Russis, 2024; Monge Roffarello and De Russis, 2023) aligned to user preferences and capabilities, such as text or speech input, and provide concrete examples and hints to support prompt generation. Such design choices can help reduce cognitive load, ensuring a smoother initial interaction with the system. This phase may address the issue that individuals often generate instructions as commands or descriptions instead of rules (as observed in Study 3 and Study 7).
2. The second phase, **ambiguity identification**, could involve the system detecting ambiguities in the user’s instruction. These ambiguities may be related to the temporal characteristics of the trigger part and/or to categories classified in Study 6 and Study 7. The system would analyze the input to identify unclear or imprecise elements that could lead to misinterpretation/errors in code generation.
3. The third phase, the **disambiguation** process, could focus on addressing the identified ambiguities. The system could generate a structured set of clarification questions, prioritized based on their importance for rule functioning (e.g., related to safety and security first). This iterative process would help align the system’s

understanding with the user's intent, minimizing arbitrary or incorrect interpretations and enhancing rule accuracy.

From a preliminary investigation (Pilot Study described in section 7.2), we derived some suggestions for improving the questions generated by the LLM-based system:

- To ensure accessibility the system should generate **closed-ended questions** with limited options (ideally numbered or labeled) from which the user can choose an answer, such as: *"After John enters the house, do you want: A. the door to close, B. the door to remain open"*. This approach would simplify decision-making and reduce cognitive load (Finlay and Antaki, 2012).
- Avoid abstract concepts and terminology in question generation (Barricelli et al., 2023; Barricelli et al., 2024). It is important to use **concrete examples** (Hollomotz, 2018), for instance, time-related questions should be phrased as *"how many minutes"* instead of *"how much time"*. Complex or uncommon terms like "sensor", "home automation", or "smart" should be avoided or accompanied by straightforward explanations.
- Each question should remain **concise and simple**. Complex questions should be split into simpler ones, using the scheme of an introductory question and a follow-up confirmation. For example, instead of a single ambiguous question about action duration, the system could first ask if the user wants a specific duration and then follow up with whether it should be longer or shorter than a given time frame.
- For what concerns output, the value of **visual aids** to simplify complex concepts like time sequences (before, during, after). Visual cues, combined with system-generated questions, could improve user comprehension and facilitate iterative refinement. Integrating model-generated images, consistent with prior studies (Calò and De Russis, 2024), could further enhance the system's accessibility.
- In general, **multi-modality** is an essential aspect of interface design based on LLM (Barricelli et al., 2022; Monge Roffarello and De Russis, 2023). The system should include a number of customizable features as interaction modes since individual differences significantly influence preferences.
- Ideally, the system should generate **concrete and experience-related questions** using real-life, context-specific examples from the user's personal or domain experience. For example, in the case of individuals with intellectual disability, the

system should reference familiar names or scenarios instead of a generic term such as "someone" (e.g., *"Do you want to open the door to John immediately after he rings the doorbell or after a few minutes?"*). Similarly, for domain-experts, the questions should be related to their domain expertise. For example, taking into consideration teachers, as we saw that the "instruct-child" metaphor is good/appropriate for guiding the interaction with a TAP-based system (Study 5), a generated question could prompt the user to imagine the classroom setting (e.g., *"Do you want to give the positive feedback to the student immediately after the positioning of the correct tile or after a few minutes?"*).

Chapter 9

Conclusions

This dissertation explored the role of language and mental models in guiding non-programmers in the definition of trigger-action rules for smart device automation. Through three studies, we examined how lexical choices influence mental representations of states and events during trigger-action programming. Our findings highlight the impact of language on users' working mental models, with specific conjunctions and verb forms either facilitating or hindering accurate representation of temporal occurrences.

Our results suggest that improving TAP requires lexically precise interfaces that align with the system's internal representation of states and events. Specifically, systems should explicitly differentiate between instantaneous events and prolonged states, even when both are recognized by the same sensor (e.g., the state "*it is raining*" vs. the event "*it starts to rain*"). Mapping this ontological distinction onto lexical choices can help reduce ambiguity, fostering more accurate mental models of rule-based automation. While designing user interfaces leveraging TAP, designers should consider:

- temporal conjunctions such as "*as soon as*" for events and "*while*" or "*as long as*" for states facilitated more precise mental representations;
- "*if*" and "*when*", despite being naturally preferred by individuals, were found to ambiguously describe both states and events, particularly when used with the present tense.
- non-programmers generally favored a *trigger-first* sequence followed by the action and, for complex rules, where the trigger involved both an event and a state, the preferred sequence was *event-state-action*.

These findings have implications for the design of EUD systems that support non-programmers in defining trigger-action rules, in at least two approaches:

- **Form-Based Compositional Visual Interfaces:** Implementing linguistic constraints can guide users toward clearer language structures, reducing ambiguity in trigger specification and enhancing comprehension.
- **LLM-Based Systems:** In natural language interfaces, word choice variability increases ambiguity. Our findings suggest that mechanisms for analyzing and controlling conjunctions and verb forms in user-generated rules can improve system interpretability.

Through four studies, we explored these two approaches: constraining mental models through structured primitives in a form-based compositional visual interface and allowing users to freely express trigger-action rules in natural language with an LLM-based system.

As a result of studies on interaction with form-based compositional visual interfaces, we highlight the following findings:

- While concrete descriptions were easier to grasp, they sometimes reinforced rigid reasoning patterns and working mental models that were not aligned with trigger-action logic.
- Abstract descriptions initially posed challenges but encouraged mental models explorations, often leading to more effective rule composition strategies.

From studies on LLM-based systems, we observed the following:

- User-generated rules exhibited significant ambiguity, often due to overly abstract formulations that made interpretation by an LLM-based system difficult.
- Clarification questions generated for disambiguation can prevent the LLM-based system from arbitrarily addressing these ambiguities. Questions should be concrete and specific to refine rules to better align with user intent.
- Non-programmers who focused on defining states tended to adopt more functional reasoning strategies. LLM-based systems could effectively assist users in this aspect, as the generated questions often considered state configurations.

9.1 Limitations

The work presented in this thesis has several limitations, some of which are discussed in the corresponding subsections of each study in Chapters 5 and 6.

A common limitation across the studies is that the results are based only on the Italian language. Although the linguistic cues examined have semantically correspondences in English and other European languages, further research is needed to validate these findings across different linguistic contexts. Our initial hypotheses were informed by prior studies conducted in other languages, such as English (Broccias et al., 2022), Spanish (De Vega et al., 2007; García-Herranz et al., 2010), and German (De Vega et al., 2007). The consistency of our results with these previous findings (e.g., issues with *"when"*) gives us confidence that our work contributes meaningfully to the literature. While our research is constrained to Italian, we believe that these results can serve as a foundation for investigating additional languages and exploring similarities among neo-latin languages (i.e., those derived from Latin).

Furthermore, the studies on user interaction with TAP-based systems (presented in Chapter 6) focused on specific use cases, namely teachers and accessibility. This targeted approach allowed us to engage motivated participants who could identify with the scenarios presented. However, it also limits the generalizability of our findings to broader populations.

Finally, another limitation is the choice to examine specific systems without comparing them to alternative solutions (e.g., another form-based compositional interface or a different LLM-based system). Future research should address this gap by conducting comparative studies.

9.2 Future Directions

The limitations identified in this work, along with our findings, suggest several promising directions for future research on language and mental models, as well as on user interaction with TAP-based systems.

9.2.1 Linguistic Cues and Mental Models

Listed below are some of the research lines that may arise from our studies.

- Expanding the investigation to other languages to assess the generalizability of linguistic cues across neo-latin languages employing the same paradigm (e.g.,

Broccias et al., 2022; De Vega et al., 2007).

- Broadening the set of language-related cues. Beyond exploring additional conjunctions and verb forms, a new direction could be to examine the influence of verb aspect or actionality (i.e., dynamicity, duration, telicity – the presence or absence of a point at which the event necessarily ends) on the use of conjunctions and tense (Vendler, 1957).
- Investigating a more diverse set of scenarios, increasing the number and the topic of textual descriptions or video-based scenarios beyond to determine whether intrinsic factors influence linguistic preferences.
- Examining whether there are differences between the preferred linguistic clues and order of elements in **interpretation** (how users understand rules) and in **composition** (how users structure rules).
- While our study systematically explored elements' order in written phrases, future research should examine the impact of vertical vs. horizontal rule elements presentation in composition and interpretation (e.g., as suggested by studies on the *SNARC* effect; Wood et al., 2008).

9.2.2 User Interaction with TAP-Based Systems

Following are reported some potential research directions that may emerge from our studies.

- Investigating strategies to nudge teachers toward more effective working mental models, building on our findings. This could be done experimentally by manipulating instructions (as in Zancanaro et al., 2022) or in real-world settings by designing training programs or tutorials.
- Conducting comparative studies with different domain-expert groups to assess the generalizability of our findings on teachers' mental models.
- Exploring different levels of abstraction and concreteness of TAP-based solutions, using the *EUDability* framework (Barricelli et al., 2023) as a baseline for comparison.
- Conducting a more systematic investigation of the conceptual mental models of computers assumed by non-programmers and how these models influence user performance in defining trigger-action rules.

- Implementing and evaluating multi-modal approaches in LLM-based systems (Monge Roffarello and De Russis, 2023), particularly in iterative disambiguation processes. This should involve expert educators, caregivers, and individuals with intellectual disabilities to enhance accessibility.

Bibliography

- Abdollahinami, S., Ducceschi, L., and Zancanaro, M. (2022). End-user development and closed-reading: an initial investigation. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI '22*, pages 1–5, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3531073.3531128.
- Abreu-Mendoza, R. A., Coulanges, L., Ali, K., Powell, A. B., and Rosenberg-Lee, M. (2021). From non-symbolic to symbolic proportions and back: a cuisenaire rod proportional reasoning intervention enhances continuous proportional reasoning skills. *Frontiers in Psychology*, 12:633077. doi: 10.3389/fpsyg.2021.633077.
- Ahmed, E., Islam, A., Sarker, F., Huda, M. N., and Abdullah-Al-Mamun, K. (2016). A road to independent living with smart homes for people with disabilities. In *2016 5th International conference on informatics, electronics and vision (ICIEV)*, pages 472–477. IEEE. doi: 10.1109/ICIEV.2016.7760048.
- Al-Emran, M., Malik, S. I., and Al-Kabi, M. N. (2020). A survey of internet of things (iot) in education: Opportunities and challenges. In *Toward social internet of things (SIoT): Enabling technologies, architectures and applications: Emerging technologies for connected and smart social objects*, pages 197–209. Springer, Cham. doi: 10.1007/978-3-030-24513-9_12.
- Al-Karawi, K. A. and Mohammed, D. Y. (2021). Improving short utterance speaker verification by combining mfcc and entropy in noisy conditions. *Multimedia Tools and Applications*, 80(14):22231–22249. <https://doi.org/10.1007/s11042-021-10767-6>.
- Almukadi, W. and Stephane, A. L. (2015). Blackblocks: tangible interactive system for children to learn 3-letter words and basic math. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces, ITS '15*, pages 421–424, New York, NY, USA. Association for Computing Machinery. November 15 - 18, 2015. doi: 10.1145/2817721.2823482.
- Almusaed, A., Yitmen, I., and Almssad, A. (2023). Enhancing smart home design with ai models: A case study of living spaces implementation review. *Energies*, 16(6):2636. <https://doi.org/10.3390/en16062636>.
- Andersen, R., Mørch, A. I., and Litherland, K. T. (2021). Learning domain knowledge using block-based programming: Design-based collaborative learning. In Fogli, D., Tetteroo, D., Barricelli, B. R., Borsci, S., Markopoulos, P., and Papadopoulos, G. A., editors, *End-User Development: 8th*

- International Symposium on End User Development, IS-EUD 2021*, pages 119–135, Cham. Springer International Publishing. July 6–8, 2021. doi: 10.1007/978-3-030-79840-6_8.
- Andrao, M., Balducci, F., Breve, B., Cena, F., Desolda, G., Deufemia, V., Gena, C., Matera, M., Mattioli, A., Paternò, F., et al. (2023a). Understanding concepts, methods and tools for end-user control of automations in ecosystems of smart objects and services. In Spano, L. D., Schmidt, A., Santoro, C., and Stumpf, S., editors, *End-User Development: 9th International Symposium on End User Development, IS-EUD 2023*, Lecture Notes in Computer Science (LNCS, volume 13917), pages 104–124, Cham. Springer, Springer Nature Switzerland. June 6–8, 2023. https://doi.org/10.1007/978-3-031-34433-6_7.
- Andrao, M., Cappelletti, A., Desolda, G., Greco, F., Treccani, B., and Zancanaro, M. (2023b). Smarter than before: Empowering teachers to program a modular iot educational device. In Bellucci, A., De Russis, L., Díaz, P., Mørch, A., Fogli, D., and Paternò, F., editors, *Workshops, Work in Progress, Demos and Doctoral Consortium at IS-EUD*, volume 3408. <https://ceur-ws.org/Vol-3408/short-s0-01.pdf>.
- Andrao, M., Desolda, G., Greco, F., Manfredi, R., Treccani, B., and Zancanaro, M. (2022a). End-user programming and math teachers: An initial study. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI 2022*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3531073.3534493.
- Andrao, M., Desolda, G., Greco, F., Manfredi, R., Treccani, B., and Zancanaro, M. (2022b). Smarter: an iot learning game to teach math. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI '22*, pages 1–3, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3531073.3534467.
- Andrao, M., Gini, F., Bucchiarone, A., Marconi, A., Treccani, B., and Zancanaro, M. (2023c). Enhance gamification design through end-user development: a proposal. In Bellucci, A., De Russis, L., Díaz, P., Mørch, A., Fogli, D., and Paternò, F., editors, *Workshops, Work in Progress, Demos and Doctoral Consortium at IS-EUD*, volume 3408. <https://ceur-ws.org/Vol-3408/short-s4-02.pdf>.
- Andrao, M., Gini, F., Frageri, D., Bucchiarone, A., Cappelletti, A., Treccani, B., and Zancanaro, M. (2024a). Rulecraft: an end-user development hub for education. In *Proceedings of the 2024 International Conference on Advanced Visual Interfaces*, pages 1–3. <https://doi.org/10.1145/3656650.3656741>.
- Andrao, M., Gini, F., Greco, F., Cappelletti, A., Desolda, G., Treccani, B., and Zancanaro, M. (2025). "react", "command", or "instruct"? teachers' mental models on end-user development. In *Proceedings of CHI Conference on Human Factors in Computing Systems, CHI '25*, New York, NY, USA. Association for Computing Machinery. April 28 - May 1, 2025. <https://doi.org/10.1145/3706598.3713234>.
- Andrao, M., Morra, D., Paccosi, T., Matera, M., Treccani, B., and Zancanaro, M. (2024b). "this sounds unclear": Evaluating chatgpt capability in translating end-user prompts into ready-to-deploy python

- code. In *Proceedings of the 2024 International Conference on Advanced Visual Interfaces, AVI '24*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3656650.3656693.
- Andrao, M., Treccani, B., and Zancanaro, M. (2023d). Language and temporal aspects: A qualitative study on trigger interpretation in trigger-action rules. In Spano, L. D., Schmidt, A., Santoro, C., and Stumpf, S., editors, *End-User Development: 9th International Symposium on End User Development, IS-EUD 2023*, Lecture Notes in Computer Science (LNCS, volume 13917), pages 84–103, Cham. Springer, Springer Nature Switzerland. June 6–8, 2023. Cagliari, Italy. https://doi.org/10.1007/978-3-031-34433-6_6.
- Andrao, M., Treccani, B., and Zancanaro, M. (In-submission). Making trigger-action rules more comprehensible: Investigating which linguistic clues effectively guide non-programmers. *International Journal of Human-Computer Studies*. In-submission.
- APA (2013). *Diagnostic and Statistical Manual of Mental Disorders (DSM-5®)*. American Psychiatric Publishing. <https://books.google.it/books?id=-JivBAAAQBAJ>.
- APA (n.d.). *Apa dictionary of psychology*. <https://dictionary.apa.org/>.
- Ardito, C., Buono, P., Desolda, G., and Matera, M. (2018). From smart objects to smart experiences: An end-user development approach. *International Journal of Human-Computer Studies*, 114:51–68. *Advanced User Interfaces for Cultural Heritage*. <https://doi.org/10.1016/j.ijhcs.2017.12.002>.
- Arduino (n.d.). *Arduino*. <https://www.arduino.cc/>.
- Artizzu, V., Carcangiu, A., Manca, M., Mattioli, A., Mereu, J., Paternò, F., Santoro, C., Simeoli, L., and Spano, L. D. (2024). End-user development for extended reality using a multimodal intelligent conversational agent. In *Proceedings of the Workshop on Prototyping and Developing Real-World Applications of Extended Reality at the 17th International Conference on Advanced Visual Interfaces (Genoa, Italy)*. <https://ceur-ws.org/Vol-3704/paper5.pdf>.
- Assistant, H. (n.d.). *Home assistant*. <https://www.home-assistant.io/>.
- Atooma (n.d.). *Atooma. resonance — predicting human behaviours*. <https://atooma.com/>.
- Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805. <https://doi.org/10.1016/j.comnet.2010.05.010>.
- AutomateIt (n.d.). *Automateit - automate tasks on your android*. <https://automateitapp.com/>.
- Bacchin, D., Pluchino, P., Grippaldi, A. Z., Mapelli, D., Spagnolli, A., Zanella, A., and Gamberini, L. (2021). Smart co-housing for people with disabilities: A preliminary assessment of caregivers' interaction with the domho system. *Frontiers in psychology*, 12:734180. <https://doi.org/10.3389/fpsyg.2021.734180>.

- Baddeley, A. D. and Hitch, G. (1974). Working memory. In Bower, G. H., editor, *Psychology of Learning and Motivation*, volume 8, pages 47–89. Academic Press. [https://doi.org/10.1016/S0079-7421\(08\)60452-1](https://doi.org/10.1016/S0079-7421(08)60452-1).
- Barkhuus, L. and Vallgård, A. (2003). Smart home in your pocket. In *Adjunct Proceedings of UbiComp*, pages 165–166. Citeseer. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2376102ffcb8f93d29cad1918d7d22a9694de4b5>.
- Barricelli, B. R., Cassano, F., Fogli, D., and Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137. <https://doi.org/10.1016/j.jss.2018.11.041>.
- Barricelli, B. R., Fogli, D., Gargioni, L., Locoro, A., and Valtolina, S. (2024). Towards the unification of computational thinking and eudability: Two cases from healthcare. In *Proceedings of the 2024 International Conference on Advanced Visual Interfaces, AVI '24*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3656650.3656671.
- Barricelli, B. R., Fogli, D., Iemmolo, L., and Locoro, A. (2022). A multi-modal approach to creating routines for smart speakers. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI '22*, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3531073.3531168>.
- Barricelli, B. R., Fogli, D., and Locoro, A. (2023). Eudability: A new construct at the intersection of end-user development and computational thinking. *Journal of Systems and Software*, 195:111516. <https://doi.org/10.1016/j.jss.2022.111516>.
- Barricelli, B. R. and Valtolina, S. (2017). A visual language and interactive system for end-user development of internet of things ecosystems. *Journal of Visual Languages & Computing*, 40:1–19. Semiotics, Human-Computer Interaction and End-User Development. <https://doi.org/10.1016/j.jvvlc.2017.01.004>.
- Bellucci, A., Vianello, A., Florack, Y., Micallef, L., and Jacucci, G. (2019). Augmenting objects at home through programmable sensor tokens: A design journey. *International Journal of Human-Computer Studies*, 122:211–231. <https://doi.org/10.1016/j.ijhcs.2018.09.002>.
- Benavides-Varela, S., Callegher, C. Z., Fagiolini, B., Leo, I., Altoè, G., and Lucangeli, D. (2020). Effectiveness of digital-based interventions for children with mathematical learning difficulties: A meta-analysis. *Computers & Education*, 157:103953. <https://doi.org/10.1016/j.compedu.2020.103953>.
- Binet, M. A. (1894). Reverse illusions of orientation. *Psychological Review*, 1(4):337. <https://doi.org/10.1037/h0067333>.
- Blackwell, A. (2002). First steps in programming: a rationale for attention investment models. In *Proceedings of the IEEE 2002 Symposium on Human Centric Computing Languages and Environments (HCC'02)*, HCC '02, page 2, USA. IEEE Computer Society. doi: 10.1109/HCC.2002.1046334.

- Bonar, J. and Soloway, E. (1983). Uncovering principles of novice programming. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '83, pages 10–13, New York, NY, USA. Association for Computing Machinery. January 24 - 26, 1983. doi: 10.1145/567067.567069.
- Bonar, J. and Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human-Computer Interaction*, 1(2):133–161. doi: 10.1207/s15327051hci0102_3.
- Boren, T. and Ramey, J. (2000). Thinking aloud: Reconciling theory and practice. *IEEE transactions on professional communication*, 43(3):261–278. doi: 10.1109/47.867942.
- Boxes.py (n.d.). Boxes.py. <https://boxes.hackerspace-bamberg.de/>.
- Brackenbury, W., Deora, A., Ritchey, J., Vallee, J., He, W., Wang, G., Littman, M. L., and Ur, B. (2019). How users interpret bugs in trigger-action programming. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, CHI '19, pages 1–12, New York, NY, USA. Association for Computing Machinery. May 4 - 9, 2019. doi: 10.1145/3290605.3300782.
- Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101. doi: 10.1191/1478088706qp063oa.
- Breve, B., Desolda, G., Greco, F., and Deufemia, V. (2023). Democratizing cybersecurity in smart environments: Investigating the mental models of novices and experts. In Spano, L. D., Schmidt, A., Santoro, C., and Stumpf, S., editors, *End-User Development: 9th International Symposium on End User Development, IS-EUD 2023*, pages 145–161, Cham. Springer Nature Switzerland. June 6–8, 2023. doi: 10.1007/978-3-031-34433-6_9.
- Brich, J., Walch, M., Rietzler, M., Weber, M., and Schaub, F. (2017). Exploring end user programming needs in home automation. *ACM Trans. Comput.-Hum. Interact.*, 24(2). <https://doi.org/10.1145/3057858>.
- Bridge, C., Zmudzki, F., Huang, T., Owen, C., and Faulkner, D. (2021). Impacts of new and emerging assistive technologies for ageing and disabled housing. *AHURI Final Report*. doi: 10.18408/ahuri7122501.
- Broccias, C., Baicchi, A., and Bagli, M. (2022). Contrasting english and italian simultaneity subordinators as/come, while/mentre, when/quando: A preliminary investigation. *Cognition and Contrast: Festschrift for/für Prof. Dr. Sabine De Knop*, page 177. doi: 10.4000/books.pusl.27908.
- Burnett, M. M. and Scaffidi, C. (2013). End-user development. In *The Encyclopedia of Human-Computer Interaction*, 2nd Ed., chapter 10. The Interaction Design Foundation. <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/end-user-development>.

- Calò, T. and De Russis, L. (2024). Enhancing smart home interaction through multimodal command disambiguation. *Personal and Ubiquitous Computing*, pages 1–16. <https://doi.org/10.1007/s00779-024-01827-3>.
- Cañas, J. J., Antolí, A., and Quesada, J. F. (2001). The role of working memory on measuring mental models of physical systems. *Psicológica*, 22(1):25–42. <https://www.redalyc.org/pdf/169/16917746003.pdf>.
- Card, S. (1983). *The Psychology of Human-Computer Interaction (1st Ed.* CRC Press., Florida, US. <https://doi.org/10.1201/9780203736166>.
- Casati, F., Castano, S., Fugini, M., Mirbel, I., and Pernici, B. (2000). Using patterns to design rules in workflows. *IEEE Transactions on Software Engineering*, 26(8):760–785. doi: 10.1109/32.879813.
- Clark, M., Dutta, P., and Newman, M. W. (2016). Towards a natural language programming interface for smart homes. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, UbiComp '16*, page 49–52, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/2968219.2971443.
- Clarke, V. and Braun, V. (2017). Thematic analysis. *The journal of positive psychology*, 12(3):297–298. doi: 10.1080/17439760.2016.1262613.
- Cleland, J., Hutchinson, C., Williams, P. A., Manuel, K., and Laver, K. (2024a). The experience of using home automation by individuals with disability. *Disability and Rehabilitation: assistive technology*, 19(6):2389–2396. doi: 10.1080/17483107.2023.2288391.
- Cleland, J., Hutchinson, C., Williams, P. A., Manuel, K., and Laver, K. (2024b). A scoping review to explore the health, social and economic outcomes of home automation for people with disability. *Disability and Rehabilitation: Assistive Technology*, 19(4):1446–1453. doi: 10.1080/17483107.2023.2196308.
- Collins, A. and Gentner, D. (1987). How people construct mental models. In *Cultural models in language and thought*, chapter 10, pages 243–265. Cambridge University Press, Cambridge, UK. <https://doi.org/10.1017/CBO9780511607660.011>.
- Cooper, W. E. and Klouda, G. V. (1995). The psychological basis of syntactic iconicity. In *Syntactic iconicity and linguistic freezes: The human dimension*, chapter 9, page 331. Walter de Gruyter. <https://doi.org/10.1515/9783110882926>.
- Coppens-Hofman, M. C., Terband, H., Snik, A. F., and Maassen, B. A. (2017). Speech characteristics and intelligibility in adults with mild and moderate intellectual disabilities. *Folia Phoniatrica et Logopaedica*, 68(4):175–182. <https://doi.org/10.1159/000450548>.
- Corno, F., De Russis, L., and Monge Roffarello, A. (2019a). Empowering end users in debugging trigger-action rules. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19*, page 1–13, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3290605.3300618.

- Corno, F., De Russis, L., and Monge Roffarello, A. (2020). Heytap: Bridging the gaps between users' needs and technology in if-then rules via conversation. In *Proceedings of the 2020 International Conference on Advanced Visual Interfaces, AVI '20*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3399715.3399905.
- Corno, F., De Russis, L., and Roffarello, A. M. (2017). A semantic web approach to simplifying trigger-action programming in the iot. *Computer*, 50(11):18–24. doi: 10.1109/MC.2017.4041355.
- Corno, F., De Russis, L., and Roffarello, A. M. (2019b). A high-level semantic approach to end-user development in the internet of things. *International Journal of Human-Computer Studies*, 125:41–54. <https://doi.org/10.1016/j.ijhcs.2018.12.008>.
- Craik, K. J. W. (1944). *The nature of explanation*, volume 445. University Press, Macmillan, Oxford, England. doi: 10.2307/2181361.
- Dahl, Y. and Svendsen, R.-M. (2011). End-user composition interfaces for smart environments: A preliminary study of usability factors. In Marcus, A., editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, pages 118–127, Berlin, Heidelberg. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-21708-1.14.
- De Kleer, J. (1990). Multiple representations of knowledge in a mechanics problem-solver. In *Readings in qualitative reasoning about physical systems*, pages 40–45. Elsevier. <https://doi.org/10.1016/B978-1-4832-1447-4.50009-2>.
- De Kleer, J. and Brown, J. S. (1983). Assumptions and ambiguities in mechanistic mental models. In *Mental Models, 1st Ed.*, chapter 8, pages 163–198. Psychology Press, New York, US. <https://doi.org/10.4324/9781315802725>.
- De Russis, L. and Corno, F. (2015). Homerules: A tangible end-user programming interface for smart homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15*, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/2702613.2732795>.
- De Russis, L. and Monge Roffarello, A. (2018). A debugging approach for trigger-action programming. In *Extended abstracts of the 2018 CHI conference on human factors in computing systems, CHI EA '18*, pages 1–6, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3170427.3188641.
- De Vega, M., Rinck, M., Díaz, J. M., and León, I. (2007). Figure and ground in temporal sentences: The role of the adverbs when and while. *Discourse Processes*, 43(1):1–23. <https://doi.org/10.1080/01638530709336891>.
- De Vega, M., Robertson, D. A., Glenberg, A. M., Kaschak, M. P., and Rinck, M. (2004). On doing two things at once: Temporal constraints on actions in language comprehension. *Memory & cognition*, 32(7):1033–1043. doi: 10.3758/BF03196879.

- Debnath, N. C., Banerjee, S., Van, G. U., Quang, P. T., and Thanh, D. (2022). An ontology based approach towards end user development of iot. In Gupta, B., Bandi, A., and Hossain, M., editors, *Proceedings of 37th International Conference on Computers and Their Applications, CATA2022*, volume 82 of *EPIC Sre. Computing*, pages 1–10. 21-23 March 2022. <https://easychair.org/publications/volume/CATA2022>.
- Delgado, A. J., Wardlow, L., McKnight, K., and O'Malley, K. (2015). Educational technology: A review of the integration, resources, and effectiveness of technology in k-12 classrooms. *Journal of Information Technology Education*, 14. doi: 10.28945/2298.
- Desolda, G., Ardito, C., and Matera, M. (2017). Empowering end users to customize their smart environments: model, composition paradigms, and domain-specific tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–52. doi: 10.1145/3057859.
- Desolda, G., Greco, F., Guarnieri, F., Mariz, N., and Zancanaro, M. (2021a). Sensation: an authoring tool to support event-state paradigm in end-user development. In Ardito, C., Lanzilotti, R., Malizia, A., Petrie, H., Piccinno, A., Desolda, G., and Inkpen, K., editors, *Human-Computer Interaction-INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part II 18*, pages 373–382. Springer. https://doi.org/10.1007/978-3-030-85616-8_22.
- Desolda, G., Greco, F., Zancanaro, M., and Costabile, M. F. (2021b). Extending task automation systems with event-state-condition-action capabilities. In *CEUR WORKSHOP PROCEEDINGS*, volume 3053, pages 48–51. CEUR-WS. 30 August 2021. https://ceur-ws.org/Vol-3053/paper_10.pdf.
- Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: defining “gamification”. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. doi: 10.1145/2181037.2181040.
- Dewsbury, G., Clarke, K., Rouncefield, M., Sommerville, I., Taylor, B., and Edge, M. (2003). Designing acceptable ‘smart’home technology to support people in the home. *Technology and Disability*, 15(3):191–199. doi: 10.3233/TAD-2003-15305.
- Dey, A. K., Sohn, T., Streng, S., and Kodama, J. (2006). icap: Interactive prototyping of context-aware applications. In Fishkin, K. P., Schiele, B., Nixon, P., and Quigley, A., editors, *Pervasive Computing: 4th International Conference, PERVASIVE 2006, Dublin, Ireland, May 7-10, 2006. Proceedings 4*, pages 254–271, Berlin, Heidelberg. Springer. https://doi.org/10.1007/11748625_16.
- Dobkin, B. and Dorsch, A. (2017). The evolution of personalized behavioral intervention technology: Will it change how we measure or deliver rehabilitation? *Stroke*, 48:2329–2334. doi: 10.1161/STROKEAHA.117.016620.
- Downing, L. J. and Stiebels, B. (2012). Iconicity. *The morphology and phonology of exponence*, pages 379–426. doi: 10.1093/acprof:oso/9780199573721.003.0012.

- Erfurt, G., Hornecker, E., Ehlers, J., and Plaschkies, S. (2019). Hands-on math: A training system for children with dyscalculia. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, pages 1–6, New York, NY, USA. Association for Computing Machinery. May 4 - 9, 2019. doi: 10.1145/3290607.3313012.
- Eriksson, K. and Jansson, F. (2016). Procedural priming of a numerical cognitive illusion. *Judgment and decision making*, 11(3):205–212. doi: 10.1017/S1930297500003053.
- Ezeamuzie, N. O., Leung, J. S., and Ting, F. S. (2022). Unleashing the potential of abstraction from cloud of computational thinking: A systematic review of literature. *Journal of Educational Computing Research*, 60(4):877–905. <https://doi.org/10.1177/07356331211055379>.
- Fan, M., Antle, A. N., and Cramer, E. S. (2016). Design rationale: opportunities and recommendations for tangible reading systems for children. In *Proceedings of the the 15th international conference on interaction design and children*, IDC '16, pages 101–112, New York, NY, USA. Association for Computing Machinery. June 21 - 24, 2016. doi: 10.1145/2930674.2930690.
- Fan, M., Antle, A. N., Hoskyn, M., Neustaedter, C., and Cramer, E. S. (2017). Why tangibility matters: A design case study of at-risk children learning to read and spell. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, CHI '17, pages 1805–1816, New York, NY, USA. Association for Computing Machinery. May 6 - 11, 2017. doi: 10.1145/3025453.3026048.
- Finlay, W. M. and Antaki, C. (2012). How staff pursue questions to adults with intellectual disabilities. *Journal of Intellectual Disability Research*, 56(4):361–370. <https://doi.org/10.1111/j.1365-2788.2011.01478.x>.
- Fischer, G. (1991). The importance of models in making complex systems comprehensible. In Tauber, M. and Ackermann, D., editors, *Human Factors in Information Technology*, volume 2 of *Human Factors in Information Technology*, pages 3–36. North-Holland Elsevier. <https://doi.org/10.1016/B978-0-444-88602-6.50005-0>.
- Fischer, G. (2001). User modeling in human–computer interaction. *User modeling and user-adapted interaction*, 11:65–86. doi: 10.1023/A:1011145532042.
- Fischer, G. (2017). Exploring richer ecologies between designers and users. In *Conversations Around Semiotic Engineering*, pages 21–29. Springer International Publishing, Cham. doi: 10.1007/978-3-319-56291-9_4.
- Fischer, G. and Giaccardi, E. (2006). Meta-design: A framework for the future of end-user development. *End user development*, pages 427–457. https://doi.org/10.1007/1-4020-5386-X_19.
- Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., and Mehandjiev, N. (2004). Meta-design: a manifesto for end-user development. *Commun. ACM*, 47(9):33–37. doi: 10.1145/1015864.1015884.
- Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2):93–116. doi: 10.1080/08993400802114508.

- Fitzgerald, S., McCauley, R., Hanks, B., Murphy, L., Simon, B., and Zander, C. (2010). Debugging from the student perspective. *IEEE Transactions on Education*, 53(3):390–396. doi: 10.1109/TE.2009.2025266.
- Fogli, D., Peroni, M., and Stefini, C. (2017). Imathome: Making trigger-action programming easy and fun. *Journal of Visual Languages & Computing*, 42:60–75. <https://doi.org/10.1016/j.jvlc.2017.08.003>.
- Fogli, D. and Tetteroo, D. (2022). End-user development for democratising artificial intelligence. *Behaviour & Information Technology*, 41(9):1809–1810. doi: 10.1080/0144929X.2022.2100974.
- Forman, G. and Pufall, P. B. (1988). *Constructivism in the computer age*. Jean Piaget Symposia. Psychology Press. <https://api.taylorfrancis.com/content/books/mono/download?identifierName=doi&identifierValue=10.4324/9780203771242&type=googlepdf>.
- Fronza, I. and Pahl, C. (2018). End-user software engineering in k-12 by leveraging existing curricular activities. In *Proceedings of the 13th International Conference on Software Technologies - Volume 1: ICSOFT*, pages 283–289, Setubal, Portugal. SciTePress. doi: 10.5220/0006846702830289.
- Gallitto, G., Treccani, B., Zancanaro, M., et al. (2020). If when is better than if (and while might help): on the importance of influencing mental models in eud (a pilot study). In *CEUR WORKSHOP PROCEEDINGS*, volume 2702, pages 7–11. CEUR-WS. 29 September 2020. https://ceur-ws.org/Vol-2702/EMPATHY_2020_paper_2.pdf.
- Gallo, S., Manca, M., Mattioli, A., Paternò, F., and Santoro, C. (2021). Comparative analysis of composition paradigms for personalization rules in iot settings. In Fogli, D., Tetteroo, D., Barricelli, B. R., Borsci, S., Markopoulos, P., and Papadopoulos, G. A., editors, *End-User Development*, pages 53–70, Cham. Springer International Publishing. doi: 10.1007/978-3-030-79840-6_4.
- Gallo, S. and Paterno, F. (2022). A conversational agent for creating flexible daily automation. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI '22*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3531073.3531090.
- Gallo, S., Paternò, F., and Malizia, A. (2024). A conversational agent for creating automations exploiting large language models. *Personal and Ubiquitous Computing*, pages 1–16. doi: 0.1007/s00779-024-01825-5.
- Galotti, K. M. and Ganong, W. F. (1985). What non-programmers know about programming: Natural language procedure specification. *International Journal of Man-Machine Studies*, 22(1):1–10. [https://doi.org/10.1016/S0020-7373\(85\)80073-0](https://doi.org/10.1016/S0020-7373(85)80073-0).
- García-Herranz, M., Haya, P. A., and Alamán, X. (2010). Towards a ubiquitous end-user programming system for smart spaces. *J. Univers. Comput. Sci.*, 16(12):1633–1649. doi: 10.3217/jucs-016-12-1633.

- Garzotto, F. and Gonella, R. (2011). An open-ended tangible environment for disabled children's learning. In *Proceedings of the 10th International Conference on Interaction Design and Children, IDC '11*, pages 52–61, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/1999030.1999037.
- Gennari, R., Matera, M., Melonio, A., Mores, M., Morra, D., and Rizvi, M. (2024). A rapid-prototyping toolkit for people with intellectual disabilities. *International Journal of Human-Computer Studies*, 192:103347. <https://doi.org/10.1016/j.ijhcs.2024.103347>.
- Gennari, R., Matera, M., Morra, D., Melonio, A., and Rizvi, M. (2023). Design for social digital well-being with young generations: Engage them and make them reflect. *International Journal of Human-Computer Studies*, 173:103006. <https://doi.org/10.1016/j.ijhcs.2023.103006>.
- Gennari, R., Matera, M., Morra, D., and Rizvi, M. (2022). A phygital toolkit for rapidly designing smart things at school. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces, AVI '22*, pages 1–5, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3531073.3531119.
- Gentner, D. and Stevens, A. (1983). *Mental Models, 1st Ed.* Psychology Press, New York, US. <https://doi.org/10.4324/9781315802725>.
- Gentry, T. (2009). Smart homes for people with neurological disability: State of the art. *NeuroRehabilitation*, 25(3):209–217. doi: 10.3233/NRE-2009-0517.
- Gervais, M., Bouchard, B., Labreque, L., Tremblay, V., Bouchard, J., Chouinard, M.-C., Dionne, C., Bouchard, K., and Gaboury, S. (2023). Technological tools for assisting people with autism spectrum disorder (asd), intellectual development disorder (idd) and physical disabilities (pd): Developing a technological tool that assists people with different disabilities or disorders to help their personal autonomy. In *Proceedings of the 16th International Conference on Pervasive Technologies Related to Assistive Environments, PETRA '23*, page 171–176, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3594806.3594857>.
- Ghiani, G., Manca, M., Paternò, F., and Santoro, C. (2017). Personalization of context-dependent applications through trigger-action rules. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–33. doi: 10.1145/3057861.
- Gini, F., Andrao, M., Bagna, M., Bucchiarone, A., Treccani, B., Zancanaro, M., and Marconi, A. (2024). Gamified tangible iot for education: Exploring usability in smartgame. In *Proceedings of the 2024 International Conference on Advanced Visual Interfaces, AVI '24*, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3656650.3656713>.
- Gini, F., Andrao, M., and Marconi, A. (2023). The smartgame: mixing digital and tangible to foster math education and social interaction. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference, IDC '23*, pages 588–592, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3585088.3593866.

- Girouard, A., Solovey, E. T., Hirshfield, L. M., Ecott, S., Shaer, O., and Jacob, R. J. (2007). Smart blocks: a tangible mathematical manipulative. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 183–186, New York, NY, USA. Association for Computing Machinery. February 15 - 17, 2007. doi: 10.1145/1226969.1227007.
- Givón, T. (1985). Language, function and typology. *Journal of Literary Semantics*. <https://doi.org/10.1515/jlse.1985.14.2.83>.
- Givón, T. (1992). The grammar of referential coherence as mental processing instructions. *Linguistics*, 30(1):5–56. doi: 10.1515/ling.1992.30.1.5.
- Good, J. and Howland, K. (2017). Programming language, natural language? supporting the diverse computational activities of novice programmers. *Journal of Visual Languages & Computing*, 39:78–92. Special Issue on Programming and Modelling Tools. <https://doi.org/10.1016/j.jvlc.2016.10.008>.
- Haiman, J. (1980). The iconicity of grammar: Isomorphism and motivation. *Language*, 56(3):515–540. <https://doi.org/10.2307/414448>.
- Hoc, J.-M. and Nguyen-Xuan, A. (1990). Language semantics, mental models and analogy. In Hoc, J.-M., Green, T., Samurçay, R., and Gilmore, D., editors, *Psychology of Programming*, chapter 2.3, pages 139–156. Academic Press, London. <https://doi.org/10.1016/B978-0-12-350772-3.50014-8>.
- Hollomotz, A. (2018). Successful interviews with people with intellectual disability. *Qualitative Research*, 18(2):153–170. <https://doi.org/10.1177/1468794117713810>.
- Homey (n.d.). Homey - a better smart home. <https://homey.app/>.
- Huang, J. and Cakmak, M. (2015). Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing*, UbiComp '15, pages 215–225, New York, NY, USA. Association for Computing Machinery. September 7 - 11, 2015. doi: 10.1145/2750858.2805830.
- Hubitat (n.d.). Hubitat Elevation® — Local, reliable, fast and private home automation. <https://hubitat.com/>.
- IFTTT (n.d.). IFTTT - Automate business & home. <https://ifttt.com/>.
- Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6 in Cognitive science series. Harvard University Press, USA. isbn: 0674568826.
- Johnson-Laird, P. N. (1989). Mental models. In *Foundations of Cognitive Science*, chapter 12, page 469–499. MIT Press, Cambridge, MA, USA. doi: 10.5555/102953.102965.
- Johnson-Laird, P. N. (2004). The history of mental models. In *Psychology of Reasoning: Theoretical and Historical Perspectives (1st ed.)*, pages 179–212. Psychology Press. <https://doi.org/10.4324/9780203506936>.

- Johnson-Laird, P. N. (2010). Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250. doi: 10.1073/pnas.1012933107.
- Johnson-Laird, P. N., Legrenzi, P., and Legrenzi, M. S. (1972). Reasoning and a sense of reality. *British journal of Psychology*, 63(3):395–400. <https://doi.org/10.1111/j.2044-8295.1972.tb01287.x>.
- Keyvan, K. and Huang, J. X. (2022). How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges. *ACM Computing Surveys*, 55(6):1–40. <https://doi.org/10.1145/3534965>.
- King, E., Yu, H., Lee, S., and Julien, C. (2024). Sasha: Creative goal-oriented reasoning in smart homes with large language models. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(1). <https://doi.org/10.1145/3643505>.
- Knauff, M., Fangmeier, T., Ruff, C. C., and Johnson-Laird, P. N. (2003). Reasoning, models, and images: Behavioral measures and cortical activity. *Journal of cognitive neuroscience*, 15(4):559–573. doi: 10.1162/089892903321662949.
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A. F., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., and Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3). doi: 10.1145/1922649.1922658.
- Koivisto, J. and Hamari, J. (2019). The rise of motivational information systems: A review of gamification research. *International Journal of Information Management*, 45:191–210. <https://doi.org/10.1016/j.ijinfomgt.2018.10.013>.
- Kulesza, T., Stumpf, S., Burnett, M., and Kwan, I. (2012). Tell me more? the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the sigchi conference on human factors in computing systems*, CHI '12, pages 1–10, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/2207676.2207678.
- Lainwright, N. and Pemberton, M. (2024). Assessing the response strategies of large language models under uncertainty: A comparative study using prompt engineering. Preprint on webpage at <https://osf.io/preprints/osf/34yqj-v1>.
- Landsberg, M. E. (1995). *Syntactic iconicity and linguistic freezes*. De Gruyter Mouton. <https://doi.org/10.1515/9783110882926>.
- Lewis, C. and Mack, R. (1982). Learning to use a text processing system: Evidence from “thinking aloud” protocols. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, CHI '82, page 387–392, New York, NY, USA. Association for Computing Machinery. March 15 - 17, 1982. doi: 10.1145/800049.801817.

- Li, C., Chan, E., Denny, P., Luxton-Reilly, A., and Tempero, E. (2019). Towards a framework for teaching debugging. In *Proceedings of the Twenty-First Australasian Computing Education Conference, ACE '19*, pages 79–86, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3286960.3286970.
- Lieberman, H., Paternò, F., Klann, M., and Wulf, V. (2006). End-user development: An emerging paradigm. In Lieberman, H., Paternò, F., and Wulf, V., editors, *End user development*, pages 1–8. Springer, Dordrecht. doi: 10.1007/1-4020-5386-X_1.
- Liu, Y., Le-Cong, T., Widyasari, R., Tantithamthavorn, C., Li, L., Le, X.-B. D., and Lo, D. (2024). Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Trans. Softw. Eng. Methodol.*, 33(5). doi: 10.1145/3643674.
- Lucow, W. H. (1964). An experiment with the cuisenaire method in grade three. *American Educational Research Journal*, 1(3):159–167. <https://doi.org/10.3102/00028312001003159>.
- Lukkarinen, A., Malmi, L., and Haaranen, L. (2021). Event-driven programming in programming education. *ACM Transactions on Computing Education (TOCE)*, 21(1):1 – 31. doi: 10.1145/3423956.
- Lussier-Desrochers, D., Lachapelle, Y., Leclerc, N., Pigot, H., Bauchet, J., and Giroux, S. (2012). Assessing the effect of domotics used as an assistant to meal preparation with people with an intellectual disability. In *The Fifth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, pages 1–6. <https://api.semanticscholar.org/CorpusID:251380315>.
- Majuri, J., Koivisto, J., and Hamari, J. (2018). Gamification of education and learning: A review of empirical literature. In *Proceedings of the 2nd international GamiFIN conference, GamiFIN 2018*. CEUR-WS. <http://ceur-ws.org/Vol-2186/paper2.pdf>.
- Manca, M., Paternò, F., Santoro, C., and Corcella, L. (2019). Supporting end-user debugging of trigger-action rules for iot applications. *International Journal of Human-Computer Studies*, 123:56–69. <https://doi.org/10.1016/j.ijhcs.2018.11.005>.
- Manfredi, R., Andrao, M., Greco, F., Desolda, G., Treccani, B., Zancanaro, M., et al. (2022). Toward a better understanding of end-user debugging strategies: A pilot study. In *CEUR WORKSHOP PROCEEDINGS*, volume 3172, pages 31–35. CEUR-WS. 6 June 2022. <https://ceur-ws.org/Vol-3172/short6.pdf>.
- Manzano-León, A., Camacho-Lazarraga, P., Guerrero, M. A., Guerrero-Puerta, L., Aguilar-Parra, J. M., Trigueros, R., and Alias, A. (2021). Between level up and game over: A systematic literature review of gamification in education. *Sustainability*, 13(4):2247. doi: 10.3390/su13042247.
- Markopoulos, P., Nichols, J., Paternò, F., and Pipek, V. (2017). Editorial: End-user development for the internet of things. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–3. doi: 10.1145/3054765.

- Markovits, H. (1988). Conditional reasoning, representation, and empirical evidence on a concrete task. *The Quarterly Journal of Experimental Psychology Section A*, 40(3):483–495. <https://doi.org/10.1080/02724988843000032>.
- Markovits, H. and Lortie-Forgues, H. (2011). Conditional reasoning with false premises facilitates the transition between familiar and abstract reasoning. *Child Development*, 82(2):646–660. <https://doi.org/10.1111/j.1467-8624.2010.01526.x>.
- Markovits, H. and Vachon, R. (1990). Conditional reasoning, representation, and level of abstraction. *Developmental Psychology*, 26(6):942. doi: 10.1037/0012-1649.26.6.942.
- Marquez, J., Villanueva, J., Solarte, Z., and Garcia, A. (2016). Iot in education: Integration of objects with virtual academic communities. In Rocha, Á., Correia, A. M., Adeli, H., Reis, L. P., and Mendonça Teixeira, M., editors, *New advances in information systems and technologies*, pages 201–212, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-31232-3_19.
- Mattioli, A. and Paternò, F. (2020). A visual environment for end-user creation of iot customization rules with recommendation support. In *Proceedings of the 2020 International Conference on Advanced Visual Interfaces, AVI '20*, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3399715.3399833.
- Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. (2011). Habits of programming in scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, ITiCSE '11*, page 168–172, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/1999747.1999796>.
- Meissner, J. L., Vines, J., McLaughlin, J., Nappey, T., Maksimova, J., and Wright, P. (2017). Do-it-yourself empowerment as experienced by novice makers with disabilities. In *Proceedings of the 2017 conference on designing interactive systems*, pages 1053–1065. <https://doi.org/10.1145/3064663.3064674>.
- Microsoft (n.d.). Microsoft power automate – process automation platform — microsoft. <https://www.microsoft.com/en-us/power-platform/products/power-automate>.
- Miro (n.d.). Miro — the innovation workspace. <https://miro.com/>.
- Monge Roffarello, A. and De Russis, L. (2023). Defining trigger-action rules via voice: A novel approach for end-user development in the iot. In Spano, L. D., Schmidt, A., Santoro, C., and Stumpf, S., editors, *End-User Development: 9th International Symposium on End User Development, IS-EUD 2023*, pages 65–83, Cham. Springer, Springer Nature Switzerland. June 6–8, 2023. doi: 10.1007/978-3-031-34433-6_5.
- Moray, N. (1987). Intelligent aids, mental models, and the theory of machines. *Int. J. Man-Mach. Stud.*, 27(5–6):619–629. doi: 10.1016/S0020-7373(87)80020-2.

- Mørch, A. I., Litherland, K. T., and Andersen, R. (2019). End-user development goes to school: collaborative learning with makerspaces in subject areas. In Malizia, A., Valtolina, S., Mørch, A., Serrano, A., and Stratton, A., editors, *End-User Development: 7th International Symposium, IS-EUD 2019*, pages 200–208, Cham. Springer International Publishing. July 10–12, 2019. doi: 10.1007/978-3-030-24781-2_16.
- Moreira, F. T., Ferreira, M. J., and Cardoso, A. (2017). Higher education disruption through iot and big data: A conceptual approach. In Zaphiris, P. and Ioannou, A., editors, *Learning and Collaboration Technologies. Novel Learning Ecosystems: 4th International Conference, LCT 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part I*, pages 389–405, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-58509-3_31.
- Moreira, F. T., Magalhaes, A., Ramos, F., and Vairinhos, M. (2018). The power of the internet of things in education: an overview of current status and potential. In Mealha, Ó., Divitini, M., and Rehm, M., editors, *Citizen, Territory and Technologies: Smart Learning Contexts and Practices: Proceedings of the 2nd International Conference on Smart Learning Ecosystems and Regional Development-University of Aveiro, Portugal, 22-23, June 2017 2*, pages 51–63, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-61322-2_6.
- Morra, D., Andrao, M., Matera, M., Mores, M., Treccani, B., and Zancanaro, M. (In-submission). Llms for iot programming: A study involving individuals with intellectual disabilities. *ACM Transactions on Computer-Human Interaction (TOCHI)*. In-submission.
- Morra, D., Caslini, G., Mores, M., Garzotto, F., and Matera, M. (2024). Makenodes: Opening connected-iot making to people with intellectual disability. *International Journal of Human-Computer Studies*, 190:103325. <https://doi.org/10.1016/j.ijhcs.2024.103325>.
- Moyà-Köhler, J. and Domènech, M. (2022). Challenging “the hands of technology”: An analysis of independent living for people with intellectual disabilities. *International journal of environmental research and public health*, 19(3):1701. <https://doi.org/10.3390/ijerph19031701>.
- Münste, T. F., Schiltz, K., and Kutas, M. (1998). When temporal terms belie conceptual order. *Nature*, 395(6697):71–73. <https://doi.org/10.1038/25731>.
- MUSE-TechParty (2023). Muse tech party — muse - museo delle scienze di trento. <https://www.muse.it/programma-muse-tech-party/>.
- Myers, B. A., Ko, A. J., and Burnett, M. M. (2006). Invited research overview: end-user programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems, CHI EA '06*, page 75–80, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/1125451.1125472.
- Nair, R., Chen, M., Dutt, A. S., Hagopian, L., Singh, A., and Du, M. (2022). Significant regional inequalities in the prevalence of intellectual disability and trends from 1990 to 2019: a systematic analysis of gbd 2019. *Epidemiology and Psychiatric Sciences*, 31:e91. doi: 10.1017/S2045796022000701.

- Nardi, B. A. (1993). *A small matter of programming: perspectives on end user computing*. MIT press. isbn: 0-262-14053-5.
- Nauha, L., Keränen, N. S., Kangas, M., Jämsä, T., and Reponen, J. (2018). Assistive technologies at home for people with a memory disorder. *Dementia*, 17(7):909–923. <https://doi.org/10.1177/1471301216674816>.
- Ngo, T., Kunkel, J., and Ziegler, J. (2020). Exploring mental models for transparent and controllable recommender systems: a qualitative study. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization, UMAP '20*, pages 183–191, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3340631.3394841.
- Node-RED (n.d.). Low-code programming for event-driven applications : Node-red. <https://nodered.org/>.
- Norman, D. (1986). Cognitive engineering. In *User Centered System Design: New Perspectives on Human-Computer Interaction*, chapter 3, pages 31–61. Lawrence Erlbaum Association, Mahwah, US. doi: 10.1201/b15703-3.
- Norman, D. A. (1983). Some observations on mental models mental models. In Gentner, D. and Stevens, A., editors, *Mental Models, 1st Ed.*, pages 15–22. Psychology Press, New York, US. <https://doi.org/10.4324/9781315802725>.
- Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. MIT Press, Cambridge, Massachusetts. <https://ia902800.us.archive.org/3/items/thedesignofeverydaythingsbydonnorman/The%20Design%20of%20Everyday%20Things%20by%20Don%20Norman.pdf>.
- Ohtsuka, K. and Brewer, W. F. (1992). Discourse organization in the comprehension of temporal order in narrative texts. *Discourse Processes*, 15(3):317–336. <https://doi.org/10.1080/01638539209544815>.
- Oliveira, W., Hamari, J., Shi, L., Toda, A. M., Rodrigues, L., Palomino, P. T., and Isotani, S. (2023). Tailored gamification in education: A literature review and future agenda. *Education and Information Technologies*, 28(1):373–406. <https://doi.org/10.1007/s10639-022-11122-4>.
- OpenAI (n.d.). Chatgpt. <https://chatgpt.com/>.
- Ouyang, S., Zhang, J. M., Harman, M., and Wang, M. (2023). Llm is like a box of chocolates: the non-determinism of chatgpt in code generation. *arXiv preprint arXiv:2308.02828*. <https://arxiv.org/abs/2308.02828>.
- Pandey, S. and Srivastava, S. (2011). Tiblo: a tangible learning aid for children with dyslexia. In *Procedings of the Second Conference on Creativity and Innovation in Design, DESIRE '11*, pages 211–220, New York, NY, USA. Association for Computing Machinery. October 19 - 21, 2011. doi: 10.1145/2079216.2079247.

- Pane, J. F., Myers, B. A., et al. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2):237–264. <https://doi.org/10.1006/ijhc.2000.0410>.
- Paternò, F. and Santoro, C. (2019). End-user development for personalizing applications, things, and robots. *International Journal of Human-Computer Studies*, 131:120–130. <https://doi.org/10.1016/j.ijhcs.2019.06.002>.
- Paton, N. W. (1999). *Active rules in database systems*. Monographs in Computer Science (MCS). Springer Science & Business Media, New York, NY. <https://doi.org/10.1007/978-1-4419-8656-6>.
- Pavlovia (n.d.). Pavlovia. <https://pavlovia.org/>.
- Payne, S. J. (2007). Mental models in human-computer interaction. In Sears, A. and Jacko, J., editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, 2nd Ed.*, pages 89–102. CRC press. <https://doi.org/10.1201/9781410615862>.
- Pea, R. D. (1986). Language-independent conceptual “bugs” in novice programming. *Journal of educational computing research*, 2(1):25–36. doi: 10.2190/689T-1R2A-X4W4-29J2.
- Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., and Lindeløv, J. K. (2019). Psychopy2: Experiments in behavior made easy. *Behavior research methods*, 51:195–203. <https://doi.org/10.3758/s13428-018-01193-y>.
- Perera, C., Aghaee, S., and Blackwell, A. F. (2015). Natural notation for the domestic internet of things. In Díaz, P., Pipek, V., Ardito, C., Jensen, C., Aedo, I., and Boden, A., editors, *End-User Development: 5th International Symposium, IS-EUD 2015*, pages 25–41, Cham. Springer International Publishing. May 26-29, 2015. https://doi.org/10.1007/978-3-319-18425-8_3.
- Petrick, S. R. (1976). On natural language based computer systems. *IBM Journal of Research and Development*, 20(4):314–325. doi: 10.1147/rd.204.0314.
- Piaget, J. (1964). *The early growth of logic in the child: Classification and seriation, 1st Ed.*, volume 83 of *International Library of Psychology*. Routledge. <https://books.google.es/books?id=9xKrQ34CX0cC>.
- Pianesi, F. and Varzi, A. C. (2000). Events and event talk: An introduction. In Higginbotham, J., Pianesi, F., and Varzi, A. C., editors, *Speaking of events*, pages 3–47. Oxford University Press. <https://philpapers.org/archive/VAREAE.pdf>.
- Pot, E., Monceaux, J., Gelin, R., and Maisonnier, B. (2009). Choregraphe: a graphical tool for humanoid robot programming. In *Ro-man 2009-the 18th ieee international symposium on robot and human interactive communication*, pages 46–51. IEEE. doi: 10.1109/ROMAN.2009.5326209.
- PsychoPy (n.d.). PsychoPy. <https://www.psychopy.org/>.
- Qualtrics (n.d.). Qualtrics xm - experience management software. <https://www.qualtrics.com/>.

- Rahmati, A., Fernandes, E., Jung, J., and Prakash, A. (2017). Ifttt vs. zapier: A comparative study of trigger-action programming frameworks. *arXiv preprint*. <https://doi.org/10.48550/arXiv.1709.02788>.
- Rajapakse, R., Brereton, M., Sitbon, L., and Roe, P. (2015). A collaborative approach to design individualized technologies with people with a disability. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*, pages 29–33. <https://doi.org/10.1145/2838739.2838824>.
- Ramlowat, D. D. and Pattanayak, B. K. (2019). Exploring the internet of things (iot) in education: a review. In Satapathy, S. C., Bhateja, V., Somanah, R., Yang, X.-S., and Senkerik, R., editors, *Information Systems Design and Intelligent Applications: Proceedings of Fifth International Conference INDIA 2018*, volume 2, pages 245–255, Singapore. Springer. 10.1007/978-981-13-3338-5_23.
- React (n.d.). React. <https://react.dev/>.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11):60–67. doi: 10.1145/1592761.1592779.
- Rivkin, D., Hogan, F., Feriani, A., Konar, A., Sigal, A., Liu, S., and Dudek, G. (2023). Sage: Smart home agent with grounded execution. *arXiv preprint arXiv:2311.00772*. <https://doi.org/10.48550/arXiv.2311.00772>.
- Robb, N., Boyle, B., Politis, Y., Newbutt, N., Kuo, H. J., and Sung, C. (2021). Participatory technology design for autism and cognitive disabilities: A narrative overview of issues and techniques. *Recent advances in technologies for inclusive well-being: virtual patients, gamification and simulation*, pages 469–485. https://doi.org/10.1007/978-3-030-59608-8_25.
- Romano, M., Díaz, P., and Aedo, I. (2023). Empowering teachers to create augmented reality experiences: the effects on the educational experience. *Interactive Learning Environments*, 31(3):1546–1563. doi: 10.1080/10494820.2020.1851727.
- Rouse, W. B. and Morris, N. M. (1986). On looking into the black box: Prospects and limits in the search for mental models. *Psychological bulletin*, 100(3):349–363. <https://doi.org/10.1037/0033-2909.100.3.349>.
- Rubin, E. (2001). Figure and ground. In *Visual Perception*, pages 225–229. Philadelphia: Psychology Press. The first edition is dated 1958. isbn: 0863775985.
- Russell, D. M. and Chi, E. H. (2014). Looking back: Retrospective study methods for hci. In Olson, J. S. and Kellogg, W. A., editors, *Ways of Knowing in HCI*, pages 373–393. Springer, New York, NY. doi: 10.1007/978-1-4939-0378-8_15.
- Saccardo, F., Decarli, G., Missagia, V. I., Andrao, M., Gini, F., Zancanaro, M., and Franchin, L. (2024). Emotions and interactive tangible tools for math achievement in primary schools. *Frontiers in Psychology*, 15:1440981. <https://doi.org/10.3389/fpsyg.2024.1440981>.

- Safari, M. C., Wass, S., and Thygesen, E. (2021). 'i got to answer the way i wanted to': Intellectual disabilities and participation in technology design activities. *Scandinavian Journal of Disability Research*. <https://doi.org/10.16993/sjdr.798>.
- Safari, M. C., Wass, S., and Thygesen, E. (2023). Motivation of people with intellectual disabilities in technology design activities: the role of autonomy, competence, and relatedness. *Behaviour & Information Technology*, 42(1):89–107. <https://doi.org/10.1080/0144929X.2021.2015442>.
- Salovaara, A., Bellucci, A., Vianello, A., and Jacucci, G. (2021). Programmable smart home toolkits should better address households' social needs. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3411764.3445770.
- Sammet, J. E. (1966). The use of english as a programming language. *Communications of the ACM*, 9(3):228–230. <https://doi.org/10.1145/365230.365274>.
- Sas, C. and Neustaedter, C. (2017). Exploring diy practices of complex home technologies. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–29. <https://doi.org/10.1145/3057863>.
- Schrepp, M., Hinderks, A., and Thomaschewski, J. (2017). Design and evaluation of a short version of the user experience questionnaire (ueq-s). *International Journal of Interactive Multimedia and Artificial Intelligence*, 4 (6), 103-108. <http://doi.org/10.9781/ijimai.2017.09.001>.
- Scoccia, G. L. (2023). Exploring early adopters' perceptions of chatgpt as a code generation tool. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 88–93. doi: 10.1109/ASEW60602.2023.00016.
- Semary, H., Al-Karawi, K. A., Abdelwahab, M. M., and Elshabrawy, A. (2024). A review on internet of things (iot)-related disabilities and their implications. *Journal of Disability Research*, 3(2):20240012. doi: 10.57197/JDR-2024-0012.
- SharpTools (n.d.). Sharptools - intuitive dashboards, powerful automation. <https://sharptools.io/>.
- Shepard, S. and Metzler, D. (1988). Mental rotation: effects of dimensionality of objects and type of task. *Journal of experimental psychology: Human perception and performance*, 14(1):3. <https://doi.org/10.1037/0096-1523.14.1.3>.
- Simonsen, J. and Robertson, T. (2013). *Routledge international handbook of participatory design*, volume 711. Routledge New York. <https://doi.org/10.7146/politica.v11i4.67214>.
- SmartRules (n.d.). Smartrules – make your smart home smart. <https://smartrulesapp.com/>.
- SmartThings (n.d.). Smart home with Samsung SmartThings — Samsung. <https://www.samsung.com/ie/smartthings/>.

- Smith, E., Sumner, P., Hedge, C., and Powell, G. (2023). Smart-speaker technology and intellectual disabilities: agency and wellbeing. *Disability and Rehabilitation: Assistive Technology*, 18(4):432–442. <https://doi.org/10.1080/17483107.2020.1864670>.
- Spring (n.d.). Spring. <https://spring.io/>.
- Steger, M. and Schneider, E. W. (2012). Complexity as a function of iconicity: The case of complement clause constructions in new englishes. In *Linguistic Complexity*, pages 156–191. De Gruyter, Berlin, Boston. doi: 10.1515/9783110229226.156.
- Talmy, L. (1983). How language structures space. In *Spatial orientation: Theory, research, and application*, pages 225–282. Springer. https://doi.org/10.1007/978-1-4615-9325-6_11.
- Tetteroo, D. (2023). Challenges of enabling end-users to develop systems with ai. In *End-User Development: 9th International Symposium, IS-EUD 2023, Cagliari, Italy, June 6–8, 2023, Proceedings*, page 241–249, Berlin, Heidelberg. Springer-Verlag. https://doi.org/10.1007/978-3-031-34433-6_16.
- Toda, A. M., Klock, A. C., Oliveira, W., Palomino, P. T., Rodrigues, L., Shi, L., Bittencourt, I., Gasparini, I., Isotani, S., and Cristea, A. I. (2019). Analysing gamification elements in educational environments using an existing gamification taxonomy. *Smart Learning Environments*, 6(1):1–14. <https://doi.org/10.1186/s40561-019-0106-1>.
- Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4):189. <https://doi.org/10.1037/h0061626>.
- Ulloa, M., Prado-Cabrera, D., and Cedillo, P. (2021). Systematic literature review of internet of things solutions oriented to people with physical and intellectual disabilities. In *Proceedings of the 7th International Conference on Information and Communication Technologies for Ageing Well and e-Health - ICT4AWE*, pages 228–235. INSTICC, SciTePress. doi: 10.5220/0010480902280235.
- UNESCO (n.d.). Unesco institute of statistic (uis) data browser. <https://data.uis.unesco.org/index.aspx?queryid=3801>.
- Ur, B., McManus, E., Pak Yong Ho, M., and Littman, M. L. (2014). Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems, CHI '14*, pages 803–812, New York, NY, USA. Association for Computing Machinery. 26 April 2014– 1 May 2014. doi: 10.1145/2556288.2557420.
- Ur, B., Pak Yong Ho, M., Brawner, S., Lee, J., Mennicken, S., Picard, N., Schulze, D., and Littman, M. L. (2016). Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI '16*, pages 3227–3231, New York, NY, USA. Association for Computing Machinery. May 7 - 12, 2016. doi: 10.1145/2858036.2858556.

- Valtolina, S., Barricelli, B. R., and Di Gaetano, S. (2020). Communicability of traditional interfaces vs chatbots in healthcare and smart home domains. *Behaviour & Information Technology*, 39(1):108–132. doi: 10.1080/0144929X.2019.1637025.
- Valtolina, S. and Matamoros, R. A. (2023). Eud strategy in the education field for supporting teachers in creating digital courses. In Spano, L. D., Schmidt, A., Santoro, C., and Stumpf, S., editors, *End-User Development: 9th International Symposium on End User Development, IS-EUD 2023*, pages 250–267, Cham. Springer, Springer Nature Switzerland. June 6–8, 2023. doi: 10.1007/978-3-031-34433-6_17.
- Van Dijk, T. A., Kintsch, W., et al. (1983). *Strategies of discourse comprehension*. Academic press New York. isbn: 0127120505.
- Vendler, Z. (1957). Verbs and times. *The philosophical review*, 66(2):143–160. <https://doi.org/10.2307/2182371>.
- Waranusast, R., Bang-ngoen, A., and Thipakorn, J. (2013). Interactive tangible user interface for music learning. In *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*, pages 400–405. 27-29 Nov. 2013. doi: 10.1109/IVCNZ.2013.6727048.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.
- Wiedenbeck, S. (2005). Facilitators and inhibitors of end-user development by teachers in a school. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, pages 215–222. IEEE, IEEE. 20-24, September, 2005. doi: 10.1109/VLHCC.2005.36.
- Wiedenbeck, S. and Engebretson, A. (2004). Comprehension strategies of end-user programmers in an event-driven application. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*, pages 207–214. 26-29 Sept. 2004. doi: 10.1109/VLHCC.2004.12.
- Wood, G., Willmes, K., Nuerk, H.-C., Fischer, M. H., et al. (2008). On the cognitive link between space and number: A meta-analysis of the snarc effect. *Psychology science quarterly*, 50(4):489. <https://www.psychologie-aktuell.com/fileadmin/download/PsychologyScience/4-2008/03-Wood.pdf>.
- Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dongsheng, D., Kwik, H., Hui Tan, A., Hwa, L., Li, M., and Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, 29(2-3):205–253. doi: 10.1080/08993408.2019.1565235.
- Yarosh, S. and Zave, P. (2017). Locked or not? mental models of iot feature interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17*, pages 2993–2997, New York, NY, USA. Association for Computing Machinery. May 6 - 11, 2017. doi: 10.1145/3025453.3025617.

- Yetiştirilen, B., Özsoy, I., Ayerdem, M., and Tüzün, E. (2023). Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv*. <https://arxiv.org/abs/2304.10778>.
- Young, R. M. (1983). Surrogates and mappings: Two kinds of conceptual models for interactive devices. In Gentner, D. and Stevens, A., editors, *Mental Models, 1st Ed.*, pages 43–60. Psychology Press, New York, US. <https://doi.org/10.4324/9781315802725>.
- Zancanaro, M., Gallitto, G., Yem, D., and Treccani, B. (2022). Improving mental models in iot end-user development. *Hum.-Centric Comput. Inf. Sci*, 2:48. <https://doi.org/10.22967/HCIS.2022.12.048>.
- Zapier (n.d.). Automate without limits — Zapier. <https://zapier.com/>.
- Zhong, L. and Wang, Z. (2023). Can chatgpt replace stackoverflow? a study on robustness and reliability of large language model code generation. *arXiv*. <https://arxiv.org/abs/2308.10335>.
- Zhong, Y. and Liu, C. (2014). A domain-oriented end-user design environment for generating interactive 3d virtual chemistry experiments. *Multimedia tools and applications*, 72:2895–2924. doi: 10.1007/s11042-013-1554-1.
- Zoom (n.d.). Zoom video communications [computer software]. <https://zoom.us>.
- Zwaan, R. A. (2008). Time in language, situation models, and mental simulations. *Language learning*, 58(s1):13–26. <https://doi.org/10.1111/j.1467-9922.2008.00458.x>.

Appendices

Appendix A

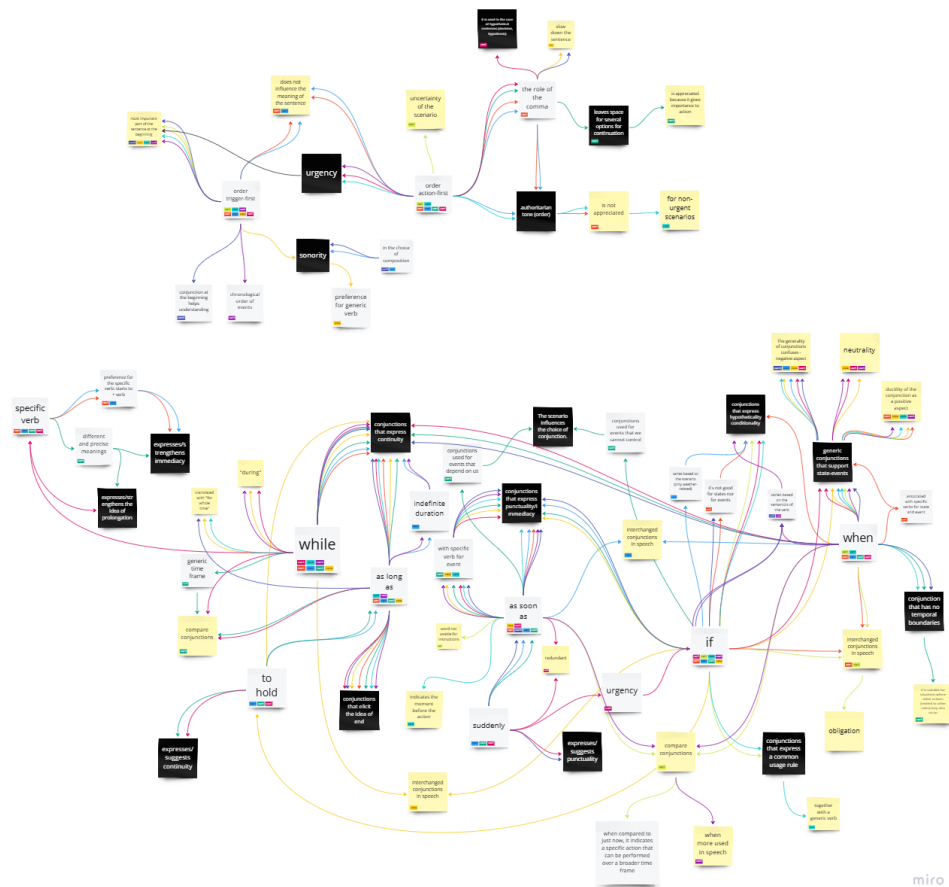


Figure 1: The figure represents a *Miroboard* screen where researchers worked to connect, reorganize, and interpret the codes that emerged from the thematic analysis.

Appendix B

Table 1: The table represents the frequencies of the preferred order divided for scenarios. Some safety-related scenarios – highlighted in **bold** – made our participants explicitly state and motivate their preference for the reverse order.

id	occurrence	order	frequency	percent
S1	<i>it rains</i>	action-first	4	33.33
		trigger-first	8	66.67
E1	<i>it rains</i>	action-first	3	25.00
		trigger-first	9	75.00
S2	<i>alarm rings</i>	action-first	2	16.67
		trigger-first	10	83.33
E2	<i>alarm rings</i>	action-first	1	8.33
		trigger-first	11	91.67
S3	<i>sun shines</i>	action-first	3	25.00
		trigger-first	9	75.00
E3	<i>sun shines</i>	action-first	2	16.67
		trigger-first	10	83.3
S4	<i>Carlo speaks</i>	action-first	4	33.33
		trigger-first	8	66.67
E4	<i>Carlo speaks</i>	action-first	4	33.33
		trigger-first	8	66.67
S5	<i>Prof. teaches</i>	action-first	2	16.67
		trigger-first	10	83.33
E5	<i>Prof. teaches</i>	action-first	3	25.00
		trigger-first	9	75.00
S6	<i>one works</i>	action-first	4	33.33
		trigger-first	8	66.67
E6	<i>one works</i>	action-first	2	16.67
		trigger-first	10	83.33
S7	<i>one studies</i>	action-first	2	16.67
		trigger-first	10	83.33
E7	<i>one studies</i>	action-first	2	16.67
		trigger-first	10	83.33
S8	<i>one runs</i>	action-first	3	25.00
		trigger-first	9	75.00
E8	<i>one runs</i>	action-first	3	25.00
		trigger-first	9	75.00
S9	<i>dog barks</i>	action-first	3	25.000
		trigger-first	9	75.000
E9	<i>dog barks</i>	action-first	2	16.67
		trigger-first	10	83.33

continued from previous page

id	occurence	order	frequency	percent
S10	<i>car moves</i>	action-first	5	41.67
		trigger-first	7	58.33
E10	<i>car moves</i>	action-first	2	16.67
		trigger-first	10	83.33
S11	<i>detects smoke</i>	action-first	4	33.33
		trigger-first	8	66.67
E11	<i>detects smoke</i>	action-first	2	16.67
		trigger-first	10	83.33
S12	<i>one sleeps</i>	action-first	5	41.67
		trigger-first	7	58.33
E12	<i>one sleeps</i>	action-first	3	25.00
		trigger-first	9	75.00

Appendix C

Table 2: In the table are listed the trigger-action rules (original Italian version and English translation) linked to the 24 scenarios used in our study.

id	verb	trigger-action rules	
		specific conjunction	generic conjunction
S1	<i>generic</i>	<i>mentre piove, tieni l'ombrello aperto</i> (while it rains, keep the umbrella open) <i>finché piove, tieni l'ombrello aperto</i> (as long as it rains, keep the umbrella open)	<i>se piove, tieni l'ombrello aperto</i> (if it rains, keep the umbrella open)
	<i>specific</i>	<i>mentre sta piovendo, tieni l'ombrello aperto</i> (while it is raining, keep the umbrella open) <i>finché sta piovendo, tieni l'ombrello aperto</i> (as long as it is raining, keep the umbrella open)	<i>se sta piovendo, tieni l'ombrello aperto</i> (if it is raining, keep the umbrella open)
E1	<i>generic</i>	<i>quando piove, chiudi subito la finestra</i> (when it rains, close the window immediately) <i>appena piove, chiudi subito la finestra</i> (as soon as it rains, close the window immediately)	<i>se piove, chiudi subito la finestra</i> (if it rains, close the window immediately)
	<i>specific</i>	<i>quando inizia a piovere, chiudi subito la finestra</i> (when it starts to rain, close the window immediately) <i>appena inizia a piovere, chiudi subito la finestra</i> (as soon as it starts to rain, close the window immediately)	<i>se piove, chiudi subito la finestra</i> (if it rains, close the window immediately)
S2	<i>generic</i>	<i>mentre la sveglia suona, tieni alto il volume</i> (while the alarm clock rings, keep the volume up) <i>finché la sveglia suona, tieni alto il volume</i> (as long as the alarm clock rings, keep the volume up)	<i>se la sveglia suona, tieni alto il volume</i> (if the alarm clock rings, keep the volume up)
	<i>specific</i>	<i>mentre la sveglia sta suonando, tieni alto il volume</i> (while the alarm clock is ringing, keep the volume up) <i>finché la sveglia sta suonando, tieni alto il volume</i> (as long as the alarm clock is ringing, keep the volume up)	<i>se la sveglia sta suonando, tieni alto il volume</i> (if the alarm clock is ringing, keep the volume up)

continued from previous page

id	verb	specific conjunction	generic conjunction
E2	<i>generic</i>	<i>quando</i> la sveglia suona, accendi subito la macchinetta del caffè (when the alarm clock rings, turn on the coffee machine immediately) <i>appena</i> la sveglia suona, accendi subito la macchinetta del caffè (as soon as the alarm clock rings, turn on the coffee machine immediately)	<i>se</i> la sveglia suona, accendi subito la macchinetta del caffè (if the alarm clock rings, turn on the coffee machine immediately)
	<i>specific</i>	<i>quando</i> la sveglia inizia a suonare, accendi subito la macchinetta del caffè (when the alarm clock starts to ring, turn on the coffee machine immediately) <i>appena</i> la sveglia suona, accendi subito la macchinetta del caffè (as soon as the alarm clock starts to ring, turn on the coffee machine immediately)	<i>se</i> la sveglia inizia a suonare, accendi subito la macchinetta del caffè (if the alarm clock starts to ring, turn on the coffee machine immediately)
S3	<i>generic</i>	<i>mentre</i> splende il sole, tieni le tapparelle abbassate (while the sun shines, keep the blinds down) <i>finché</i> splende il sole, tieni le tapparelle abbassate (as long as the sun shines, keep the blinds down)	<i>se</i> splende il sole, tieni le tapparelle abbassate (if the sun shines, keep the blinds down)
	<i>specific</i>	<i>mentre</i> sta splendendo il sole, tieni le tapparelle abbassate (while the sun is shining, keep the blinds down) <i>finché</i> sta splendendo il sole, tieni le tapparelle abbassate (as long as the sun is shining, keep the blinds down)	<i>se</i> sta splendendo il sole, tieni le tapparelle abbassate (if the sun is shining, keep the blinds down)
E3	<i>generic</i>	<i>quando</i> splende il sole, invia subito una notifica (when the sun shines, send notification immediately) <i>appena</i> splende il sole, invia subito una notifica (as soon as the sun shines, send notification immediately)	<i>se</i> splende il sole, invia subito una notifica (if the sun shines, send notification immediately)
	<i>specific</i>	<i>quando</i> inizia a splendere il sole, invia subito una notifica (when the sun starts to shine, send notification immediately) <i>appena</i> inizia a splendere il sole, invia subito una notifica (as soon as the sun starts to shine, send notification immediately)	<i>se</i> inizia a splendere il sole, invia subito una notifica (if the sun starts to shine, send notification immediately)

continued from previous page

id	verb	specific conjunction	generic conjunction
S4	generic	<i>mentre</i> Carlo parla, tieni il microfono acceso (while Carlo speaks, keep the microphone on) <i>finché</i> Carlo parla, tieni il microfono acceso (as long as Carlo speaks, keep the microphone on)	<i>se</i> Carlo parla, tieni il microfono acceso (if Carlo speaks, keep the microphone on)
	specific	<i>mentre</i> Carlo sta parlando, tieni il microfono acceso (while Carlo is speaking, keep the microphone on) <i>finché</i> Carlo sta parlando, tieni il microfono acceso (as long as Carlo is speaking, keep the microphone on)	<i>se</i> Carlo sta parlando, tieni il microfono acceso (if Carlo is speaking, keep the microphone on)
E4	generic	<i>quando</i> Carlo parla, abbassa subito il volume della musica (when Carlo speaks, turn down the music volume immediately) <i>appena</i> Carlo parla, abbassa subito il volume della musica (as soon as Carlo speaks, turn down the music volume immediately)	<i>se</i> Carlo parla, abbassa subito il volume della musica (if Carlo speaks, turn down the music volume immediately)
	specific	<i>quando</i> Carlo inizia a parlare, abbassa subito il volume della musica (when Carlo starts to speak, turn down the music volume immediately) <i>appena</i> Carlo inizia a parlare, abbassa subito il volume della musica (as soon as Carlo starts to speak, turn down the music volume immediately)	<i>se</i> Carlo inizia a parlare, abbassa subito il volume della musica (if Carlo starts to speak, turn down the music volume immediately)
S5	generic	<i>mentre</i> il Professore spiega, tieni il registratore acceso (while the Professor teaches, keep the recorder on) <i>finché</i> il Professore spiega, tieni il registratore acceso (as long as the Professor teaches, keep the recorder on)	<i>se</i> il Professore spiega, tieni il registratore acceso (if the Professor teaches, keep the recorder on)
	specific	<i>mentre</i> il Professore sta spiegando, tieni il registratore acceso (while the Professor is teaching, keep the recorder on) <i>finché</i> il Professore sta spiegando, tieni il registratore acceso (as long as the Professor is teaching, keep the recorder on)	<i>se</i> il Professore sta spiegando, tieni il registratore acceso (if the Professor is teaching, keep the recorder on)

continued from previous page

id	verb	specific conjunction	generic conjunction
E5	<i>generic</i>	<i>quando</i> il Professore spiega, attiva subito la modalità silenziosa (when the Professor teaches, turn on silent mode immediately) <i>appena</i> il Professore spiega, attiva subito la modalità silenziosa (as soon as the Professor teaches, turn on silent mode immediately)	<i>se</i> il Professore spiega, attiva subito la modalità silenziosa (if the Professor teaches, turn on silent mode immediately)
	<i>specific</i>	<i>quando</i> il Professore inizia a spiegare, attiva subito la modalità silenziosa (b the Professor starts to teach, turn on silent mode immediately) <i>appena</i> il Professore inizia a spiegare, attiva subito la modalità silenziosa (as soon as the Professors tarts to teach, turn on silent mode immediately)	<i>se</i> il Professore inizia a spiegare, attiva subito la modalità silenziosa (if the Professor starts to teach, turn on silent mode immediately)
S6	<i>generic</i>	<i>mentre</i> si lavora, tieni la temperatura a 18 gradi (while one works, keep the temperature at 18 degrees) <i>finché</i> si lavora, tieni la temperatura a 18 gradi (as long as one works, keep the temperature at 18 degrees)	<i>se</i> si lavora, tieni la temperatura a 18 gradi (if one works, keep the temperature at 18 degrees)
	<i>specific</i>	<i>mentre</i> si sta lavorando, tieni la temperatura a 18 gradi (while one is working, keep the temperature at 18 degrees) <i>finché</i> si sta lavorando, tieni la temperatura a 18 gradi (as long as one is working, keep the temperature at 18 degrees)	<i>se</i> si sta lavorando, tieni la temperatura a 18 gradi (if one is working, keep the temperature at 18 degrees)
E6	<i>generic</i>	<i>quando</i> si lavora, attiva subito la modalità non disturbare (when one works, activate do not disturb mode immediately) <i>appena</i> si lavora, attiva subito la modalità non disturbare (as soon as one works, activate do not disturb mode immediately)	<i>se</i> si lavora, attiva subito la modalità non disturbare (if one works, activate do not disturb mode immediately)
	<i>specific</i>	<i>quando</i> si inizia a lavorare, attiva subito la modalità non disturbare (when one starts to work, activate do not disturb mode immediately)	<i>se</i> si inizia a lavorare, attiva subito la modalità non disturbare (if one starts to work, activate do not disturb mode immediately)

continued from previous page

id	verb	specific conjunction	generic conjunction
		<i>appena</i> si inizia a lavorare, attiva subito la modalità non disturbare (as soon as one starts to work, activate do not disturb mode immediately)	
S7	generic	<i>mentre</i> si studia, tieni la porta chiusa (while one studies, keep the door closed) <i>finché</i> si studia, tieni la porta chiusa (as long as one studies, keep the door closed)	<i>se</i> si studia, tieni la porta chiusa (if one studies, keep the door close)
	specific	<i>mentre</i> si sta studiando, tieni la porta chiusa (while one is studying, keep the door closed) <i>finché</i> si sta studiando, tieni la porta chiusa (as long as one is studying, keep the door closed)	<i>se</i> si sta studiando, tieni la porta chiusa (if one is studying, keep the door closed)
E7	generic	<i>quando</i> si studia, disattiva subito tutte le notifiche (when one studies, turn off all notifications immediately) <i>appena</i> si studia, disattiva subito tutte le notifiche (as soon as one studies, turn off all notifications immediately)	<i>se</i> si studia, disattiva subito tutte le notifiche (if one studies, turn off all notifications immediately)
	specific	<i>quando</i> si inizia a studiare, disattiva subito tutte le notifiche (when one starts to study, turn off all notifications immediately) <i>appena</i> si inizia a studiare, disattiva subito tutte le notifiche (as soon as one starts to study, turn off all notifications immediately)	<i>se</i> si inizia a studiare, disattiva subito tutte le notifiche (if one starts to study, turn off all notifications immediately)
S8	generic	<i>mentre</i> il cane abbaia, tieni attiva la cancellazione del rumore (while the dog barks, keep noise cancellation on) <i>finché</i> il cane abbaia, tieni attiva la cancellazione del rumore (as long as the dog barks, keep noise cancellation on)	<i>se</i> il cane abbaia, tieni attiva la cancellazione del rumore (if the dog barks, keep noise cancellation on)
	specific	<i>mentre</i> il cane sta abbaiano, tieni attiva la cancellazione del rumore (while the dog is barking, keep noise cancellation on)	<i>se</i> il cane sta abbaiano, tieni attiva la cancellazione del rumore (if the dog is barking, keep noise cancellation on)

continued from previous page

id	verb	specific conjunction	generic conjunction
E8		<i>finché il cane sta abbaiano, tieni attiva la cancellazione del rumore (as long as the dog is barking, keep noise cancellation on)</i>	
	generic	<i>quando il cane abbaia, apri subito la porta (when the dog barks, open the door immediately)</i>	<i>se il cane abbaia, apri subito la porta (if the dog barks, open the door immediately)</i>
		<i>appena il cane abbaia, apri subito la porta (as soon as the dog barks, open the door immediately)</i>	
	specific	<i>quando il cane inizia ad abbaiare, apri subito la porta (when the dog starts to bark, open the door immediately)</i> <i>appena il cane inizia ad abbaiare, apri subito la porta (as soon as the dog starts to bark, open the door immediately)</i>	<i>se il cane inizia ad abbaiare, apri subito la porta (if the dog starts to bark, open the door immediately)</i>
S9	generic	<i>mentre si corre, tieni monitorata la frequenza cardiaca (while one runs, keep your heart rate monitored)</i> <i>finché si corre, tieni monitorata la frequenza cardiaca (as long as one runs, keep your heart rate monitored)</i>	<i>se si corre, tieni monitorata la frequenza cardiaca (if one runs, keep your heart rate monitored)</i>
	specific	<i>mentre si sta correndo, tieni monitorata la frequenza cardiaca (while one is running, keep your heart rate monitored)</i> <i>finché si sta correndo, tieni monitorata la frequenza cardiaca (as long as one is running, keep your heart rate monitored)</i>	<i>se si sta correndo, tieni monitorata la frequenza cardiaca (if one is running, keep your heart rate monitored)</i>
	generic	<i>quando si corre, attiva subito il conta passi (when one runs, activate the step counter immediately)</i> <i>appena si corre, attiva subito il conta passi (as soon as one runs, activate the step counter immediately)</i>	<i>se si corre, attiva subito il conta passi (if one runs, activate the step counter immediately)</i>
	specific	<i>quando si inizia a corre, attiva subito il conta passi (when one starts to run, activate the step counter immediately)</i> <i>appena si inizia a corre, attiva subito il conta passi (as soon as one starts to run, activate the step counter immediately)</i>	<i>se si inizia a corre, attiva subito il conta passi (if one starts to run, activate the step counter immediately)</i>

continued from previous page

id	verb	specific conjunction	generic conjunction
S10	generic	<i>mentre</i> la macchina si muove, tieni i fari accesi (while the car moves, keep headlights on) <i>finché</i> la macchina si muove, tieni i fari accesi (as long as the car moves, keep headlights on)	<i>se</i> la macchina si muove, tieni i fari accesi (if the car moves, keep headlights on)
	specific	<i>mentre</i> la macchina si sta muovendo, tieni i fari accesi (while the car is moving, keep headlights on) <i>finché</i> la macchina si sta muovendo, tieni i fari accesi (as long as the car is moving, keep headlights on)	<i>se</i> la macchina si sta muovendo, tieni i fari accesi (if the car is moving, keep headlights on)
E10	generic	<i>quando</i> la macchina si muove, accendi subito la radio (when the car moves, turn on the radio immediately) <i>appena</i> la macchina si muove, accendi subito la radio (as soon as the car moves, turn on the radio immediately)	<i>se</i> la macchina si muove, accendi subito la radio (if the car moves, turn on the radio immediately)
	specific	<i>quando</i> la macchina si inizia a muovere, accendi subito la radio (when the car starts to move, turn on the radio immediately) <i>appena</i> la macchina si inizia a muovere, accendi subito la radio (as soon as the car starts to move, turn on the radio immediately)	<i>se</i> la macchina si inizia a muovere, accendi subito la radio (if the car starts to move, turn on the radio immediately)
S11	generic	<i>mentre</i> si rileva del fumo, tieni attivo l'allarme anti-incendio (while one detects smoke, keep the fire alarm active) <i>finché</i> si rileva del fumo, tieni attivo l'allarme anti-incendio (as long as one detects smoke, keep the fire alarm active)	<i>se</i> si rileva del fumo, tieni attivo l'allarme anti-incendio (if one detects smoke, keep the fire alarm active)
	specific	<i>mentre</i> si sta rilevando del fumo, tieni attivo l'allarme anti-incendio (while one is detecting smoke, keep the fire alarm active) <i>finché</i> si sta rilevando del fumo, tieni attivo l'allarme anti-incendio (as long as one is detecting smoke, keep the fire alarm active)	<i>se</i> si sta rilevando del fumo, tieni attivo l'allarme anti-incendio (if one is detecting smoke, keep the fire alarm active)

continued from previous page

id	verb	specific conjunction	generic conjunction
E11	<i>generic</i>	<i>quando</i> si rileva del fumo, accendi subito gli irrigatori anti-incendio (when one detects smoke, turn on the fire sprinklers immediately) <i>appena</i> si rileva del fumo, accendi subito gli irrigatori anti-incendio (as soon as one detects smoke, turn on the fire sprinklers immediately)	<i>se</i> si rileva del fumo, accendi subito gli irrigatori anti-incendio (if one detects smoke, turn on the fire sprinklers immediately)
	<i>specific</i>	<i>quando</i> si inizia a rilevare del fumo, accendi subito gli irrigatori anti-incendio (when one starts to detect smoke, turn on the fire sprinklers immediately) <i>appena</i> si inizia a rilevare del fumo, accendi subito gli irrigatori anti-incendio (as soon as one starts to detect smoke, turn on the fire sprinklers immediately)	<i>se</i> si inizia a rilevare del fumo, accendi subito gli irrigatori anti-incendio (if one starts to detect smoke, turn on the fire sprinklers immediately)
S12	<i>generic</i>	<i>mentre</i> si dorme, tieni spenti i fornelli (while one sleeps, keep the stove off) <i>finché</i> si dorme, tieni spenti i fornelli (as long as one sleeps, keep the stove off)	<i>se</i> si dorme, tieni spenti i fornelli (if one sleeps, keep the stove off)
	<i>specific</i>	<i>mentre</i> si sta dormendo, tieni spenti i fornelli (while one is sleeping, keep the stove off) <i>finché</i> si sta dormendo, tieni spenti i fornelli (as long as one is sleeping, keep the stove off)	<i>se</i> si sta dormendo, tieni spenti i fornelli (if one is sleeping, keep the stove off)
E12	<i>generic</i>	<i>quando</i> si dorme, attiva subito l'allarme (when one sleeps, activate the alarm immediately) <i>appena</i> si dorme, attiva subito l'allarme (as soon as one sleeps, activate the alarm immediately)	<i>se</i> si dorme, attiva subito l'allarme (if one sleeps, activate the alarm immediately)
	<i>specific</i>	<i>quando</i> si inizia a dormire, attiva subito l'allarme (when one starts to sleep, activate the alarm immediately) <i>appena</i> si inizia a dormire, attiva subito l'allarme (as soon as one starts to sleep, activate the alarm immediately)	<i>se</i> si inizia a dormire, attiva subito l'allarme (if one starts to sleep, activate the alarm immediately)

Appendix D

Table 3: The table presents RTs for selecting specific vs. generic conjunctions in trials with fewer than three choices (n indicates the number of participants from a total of 82). T -test results for each trial type are reported ("*" marked instances where a *Welch's t*-test was used due to significant variance differences identified by *Fisher's F*-test).

occur.	trial	verb	n	conj	$M(s)$	$SD(s)$	t	df	p
event	<i>if-when</i>	gen.	22	if	7.73	5.10	-1.86	130	0.06
				when	6.20	4.34			
		sp.	9	if	9.46	5.80	-2.09*	42.40	0.04
				when	6.74	3.46			
	<i>if-as soon as</i>	gen.	21	if	6.64	4.40	1.25*	106.10	0.2
				as soon as	7.70	6.80			
		sp.	13	if	7.49	5.23	-1.54	76	0.13
				as soon as	5.87	3.96			
	<i>when-as soon as</i>	gen.	20	as soon as	6.47	4.68	2.43	118	0.02
				when	4.63	3.56			
		sp.	12	as soon as	7.38	5.19	0.07	70	0.94
				when	7.29	5.15			
state	<i>if-while</i>	gen.	28	if	6.25	6.81	-0.05	166	0.96
				while	6.20	5.58			
		sp.	20	if	5.82	4.1	0.72	118	0.47
				while	6.45	5.32			
	<i>if-as long as</i>	gen.	23	if	7.01	5.12	-0.49	118	0.62
				as long as	6.46	7.68			
		sp.	20	if	6.32	4.1	0.22	118	0.82
				as long as	6.5	4.68			
	<i>while-as long as</i>	gen.	12	while	7.13	5.14	0.57	70	0.57
				as long as	7.85	5.74			
		sp.	16	while	6.92	4.13	0.45	94	0.65
				as long as	7.3	4.13			

Appendix E

Table 4: The table presents the frequencies with which rules were written in action-first and trigger-first order for each scenario ("sc." refers to scenario and "n" represents the number of rules analyzed per scenario). Additionally, the table includes the chi-square (χ^2) values for each scenario, comparing the number of action-first vs. trigger-first rules.

sc.	n	order				χ^2	df	p
		<i>action-first</i>	%	<i>trigger-first</i>	%			
s1	169	36	21.30	133	78.70	55.68	1	< .001
s2	171	40	23.39	131	76.61	48.43	1	< .001
s3	165	46	27.88	119	72.12	32.30	1	< .001
s4	174	35	20.12	139	79.89	62.16	1	< .001
s5	174	45	25.86	129	74.14	40.55	1	< .001
s6	171	39	22.81	132	77.19	50.579	1	< .001
s7	178	44	24.72	134	75.28	45.51	1	< .001
s8	165	32	19.39	133	80.61	61.82	1	< .001
s9	177	40	22.60	137	77.40	53.16	1	< .001
s10	164	34	20.73	130	79.27	56.20	1	< .001
s11	171	37	21.64	134	78.36	55.02	1	< .001
s12	171	46	26.90	125	73.10	36.50	1	< .001

Appendix F



Figure 2: The figure represents the first version of *SMARTER* designed to support the use of "Cousinarie roads" in mathematics learning for elementary school children.

Appendix G

PROJECT DESIGN: SMARTER BOX

The project describes the box design for containing hardware (*Arduino*-like) with 5 RFID sensors and placing on the surface tiles with embedded circular RFID tags (diameter of 25 mm). See Figure 3.

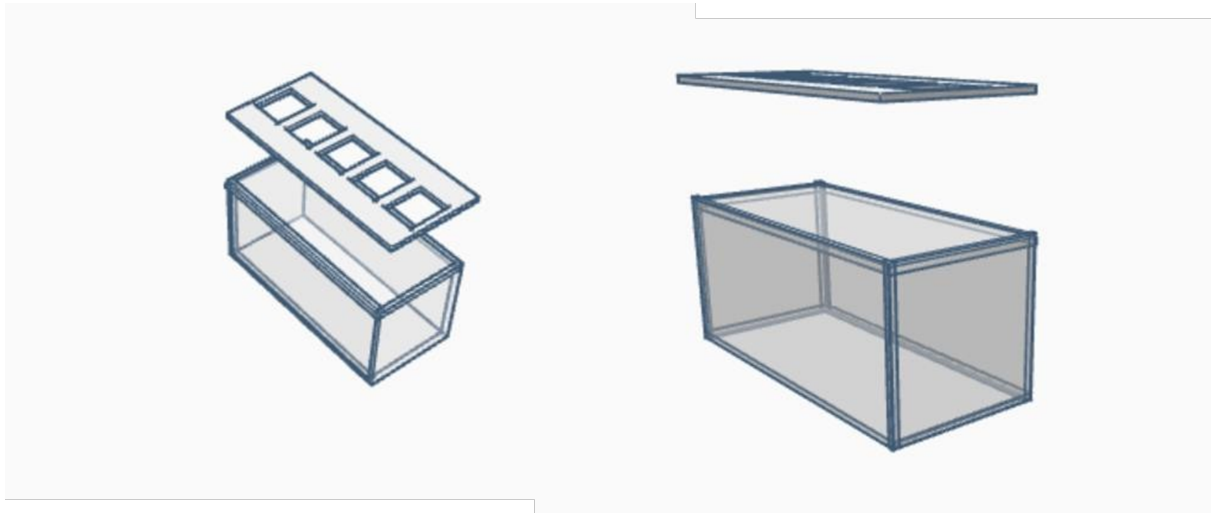


Figure 3: The figure illustrates the 6-sided box and the detachable top surface.

Box Description

Outer dimensions ($W \times H \times D$): 250 x 110 x 110 mm

Inner dimensions ($W \times H \times D$): 240 x 100 x 100 mm

Material: 5 mm-thick plywood

Notes for construction: the top surface of the box should be "embedded" 8 mm lower than the top line, allowing the removable top surface to be inserted in the box by 3 mm.

Detachable Top Surface

Dimensions ($W \times H \times D$): 240 x 100 x 3 mm

Material: 5 mm-thick plywood

The removable top surface has 5 holes (50 x 40 mm), centered and spaced 8 mm apart (see Figure 4).

Tiles

Dimensions ($W \times H \times D$): 35 x 45 x 8 mm

Material: 3 mm-thick plywood with a central circular hole (diameter of 25 mm) glued on 5 mm-thick plywood of the same dimensions (with no holes). See Figure 5.

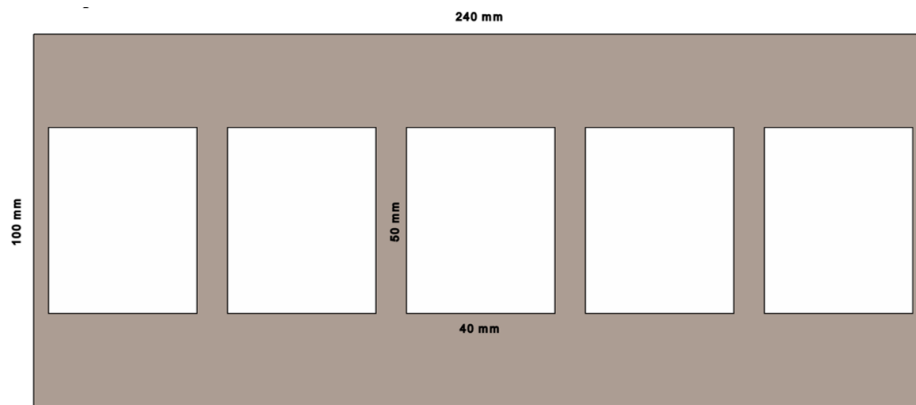


Figure 4: The figure represents the detachable top surface of the SMARTER box.

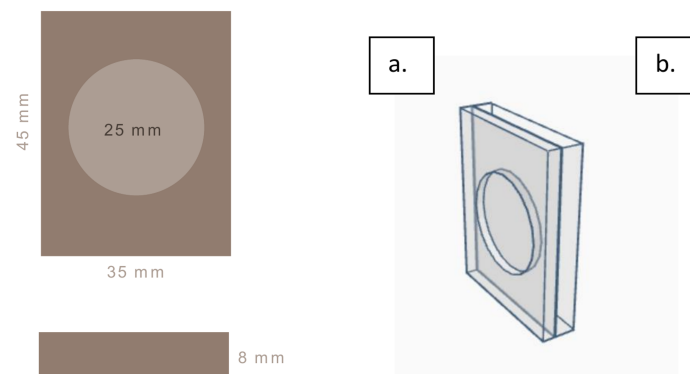


Figure 5: The left panel of the figure shows the dimensions of the tiles. The panel on the right shows how the final tiles are composed: connecting a 3 mm-thick drilled rectangle (a) with a 5 mm-thick one (b).

The project is based on an existing design (see Figure 6; cf. *Boxes.py*) by making the following changes:

- the base will not be overhanging;
- the top side visible in the figure represents our fixed top surface;
- holes on the side surfaces for the support triangles will be positioned 8 mm lower to leave adequate space for the detachable top surface.



Figure 6: The figure shows the screen of the *Boxes.py* box from which we took our cue for the design of the *SMARTER* Box.

Appendix H

Table 5: The table reports MQTT messages to interact with the *SMARTER* OS. In black are commands to send to the *SMARTER* device, while in blue are the ones sent by the *SMARTER* OS to the MQTT broker.

type	topic	payload	description
Action	<device.ID>/action	action: "play", value:<file_name>	Play the audio called <file_name>
Action	<device.ID>/action	action: "stop"	Stops audio
Action	<device.ID>/action	action: "led", value: <color>	Turn LED on with the color specified in <color> in RGB values
Action	<device.ID>/action	action: "led", value:""	Turn LED off
Action	<device.ID>	"reset"	Reboots the device
Event	<device.ID>/event	event: "card_placed", reader: <i>, value: <x>	A tile with value <x> has been placed in position <i>
Event	<device.ID>/event	event: "card_removed", reader: <i>, value: <x>	A tile with value <x> has been removed from position <i>
State	<device.ID> <device.ID>/state	"state" [5]{reader:<i>, <x>}	Requires the full board state Returns the full board state (5 reader/value objects)
State	<device.ID>/<i> <device.ID>/state	"state" reader:<i>, value: <x>	Requires the state of the reader <i> Returns the value <x> present on the reader <i> (" " = no tile)

Appendix I

The full list of the verbal primitives used by participants in creating trigger-action rules can be found in Table 6 for the concrete primitives and Table 7 for the abstract ones.

Table 6: Actions, events, and states implemented in the **concrete** version of the *SENSATION* platform.

action	description
<i>Turn off LED</i>	Turns off the LED
<i>Turn on LED <color></i>	Turns on the LED of the specified color <color> green, red, blue, white, cyan, yellow
<i>Play audio <file name></i>	Plays audio file <file name> list of available audio files
event	
<i>The tile <type> is placed in position <number></i>	Triggered when a specific tile is inserted into a specific position <type> 0 to 9, +, -, ×, ÷, =, >, < <number> 1 to 5
<i>The tile <type> is removed from position <number></i>	Triggered when a specific tile is removed from a specific position <type> 0 to 9, +, -, ×, ÷, =, >, < <number> 1 to 5
state	
<i>The position <number> is empty</i>	TRUE if the specified position is empty <number> 1 to 5
<i>The position <number> is occupied by the tile <type></i>	TRUE if the specified position is occupied by a specific tile <type> 0 to 9, +, -, ×, ÷, =, >, < <number> 1 to 5

Table 7: Actions, events, and states implemented in the **abstract** version of the *SENSATION* platform.

action	description
Turn off LED	Turns off the LED
Turn on LED <color>	Turns on the LED of the specified color <color> green, red, blue, white, cyan, yellow
Play audio <file name>	Plays audio file <file name> List of available audio files
event	
A <type> tile is placed	Triggered when a tile with the specified type is placed <type> any, digit, operator, symbol
A <type> tile is removed	Triggered when a tile with the specified type is removed <type> any, digit, operator, symbol
The expression becomes <true/false>	Triggered when the set of tiles forms a correct or incorrect math expression <true/false> true, false
state	
The position to the <right/left> of the last tile <placed/removed> is empty	TRUE if the position immediately to the right or left of the placed/removed tile is empty <right/left> right, left / <placed/removed> placed, removed
The position to the <right/left> of the last tile <placed/removed> is occupied by a <type> tile	TRUE if the position immediately to the right or left of the placed/removed tile is occupied by a tile of the specified type <right/left> right, left / <placed/removed> placed, removed / <type> any, digit, operator, symbol
In the set of tiles to the <right/left> of the last tile <placed/removed> there is a <specify>	TRUE if the set of tiles to the right/left corresponds to a number, sequence, or operation. <right/left> right, left / <placed/removed> placed, removed / <specify> number, sequence of digits, math operation
In the set of positions to the <right/left> of the last tile <placed/removed> there is a <specify>	TRUE if the set of positions to the right/left are placed a specific number of empty/occupied positions. <right/left> right, left / <placed/removed> placed, removed / <specify> no tiles are placed, at least <1, 2, 3, 4, 5> tiles are placed, less than <2, 3, 4, 5> tiles are placed
The last tile <placed/removed> is a <type> tile	TRUE if the last tile placed corresponds to the tile type. <placed/removed> placed, removed / <type> any, digit, operator, symbol

Appendix J

Table 8: The table presents the 86 raw codes extracted from the content analysis of teachers' verbal reports and their organization into sub-themes and themes.

Theme 1. Providing Instructions Step-by-step: Metaphor "Command"		
Creating a rule for each tile placement	STATE.01	The participant does not understand that she can use multiple states (concrete)
	EACH.01	The participant states she has to define a rule for the placement of each tile (both abstract and concrete)
	ABSTR.05	The participant seems to not understand that she can use the same rule for different operations (abstract)
Creating a state for every device slot	EACH.02	The participant wants to define a state for the placement of each tile (both abstract and concrete)
	WHILE.01	The participant expresses that the states are sequential and not co-occurrent (both abstract and concrete)
	BLOCK.01	The participant would prefer a block-based solution
	WHILE.04	The state is intended as the final solution
Declaring rules without actions	STEP.01	The participant interprets action-states-event as consequential (e.g., 1st number, symbol, 2nd number)
	TRIG.01	The participant does not understand the concept of trigger
	ACT.01	The participant does not want to insert the action
Theme 2. Understanding the Trigger-action Logic: Metaphor "React"		
Understanding the last tile placed triggers the action	TRIG.02	The participant seems to understand the concept of trigger (both abstract and concrete)
	DEB.04	The participant debugs using different tiles within the same category to trigger the rule (alternative case – abstract)
	FEED.01	The participant reflects on when she wants the feedback from the tool
Understanding the concept of "last tile"	ABSTR.08	The participant understands what "the last tile" means (abstract)
	DEB.02	The participant uses counter-examples during the debugging
Grasping the link between states and events	TRIG.03	The participant seems to understand the temporal relationship between states and events (both abstract and concrete)
	DEB.07	The participant debugs following the trigger-action logic (abstract and concrete)
Understanding the trigger-action logic only partially	WHEN.01	The participant wants to create a rule without a state (event-based – both abstract and concrete)
	WHILE.02	The participant wants to create a rule without the event (state-based – both abstract and concrete)
Theme 3. Instructing the System: Teaching Experience: Metaphor "Instruct"		
Focusing on children interactions - "Child"	CHILD.01	The participant expresses that the students are the ones who will use the EUD tool (both abstract and concrete)
	CHILD.02	The action is interpreted as the result of the operation and not the LED turning on/off

continued from previous page

	TRIG.02	The participant seems to understand the concept of trigger (both abstract and concrete)
	TRIG.03	The participant seems to understand the temporal relationship between states and events (both abstract and concrete)
	WHILE.03	The participant reported that the state is antecedent to the event
	CHILD.03	The participant refers to the student when she explains what happens on the IoT device (both abstract and concrete)
	CHILD.06	The participant represents the event as the tile placed by the student
	CHILD.07	The participant does not want to select an event because she represent it is as the tile placed by the student (state-based)
	FEED.01	The participant reflects on when she wants the feedback
	LING.05 /	The participant interprets "tile removed" as the tile placed later on by the student
	CHILD.08	
	DEB.07	The participant debugs following the trigger-action logic (abstract and complete)
Instructing the device as a student - "Assignment"	WHILE.03	The participant represents the state as antecedent to the event
	OBJ.01	The participant interprets the task as she has to create assignments rather than trigger-action rules
	ACT.01	The participant does not want to insert the action
	CHILD.05	The participant would make students debug the rules
	CONJ.02	The participant makes confusion between states and events
Theme 4. Influence of Conceptual Mental Models of Computer		
Computers can be Smart	COMP.01	The participant deems the system as smart, i.e., it can do smart things
	ABSTR.06	True and false is a simple concept and does not need to be defined (it simplifies things)
	DEB.04	The participant debugs with other tiles of the same category (alternative case)
	SYST.01	The participant understands that she can group rules
	SYST.02	The participant understands that she can use the same rule in multiple groups
	SYST.03	The participant understands that she can use two tiles to create a two-digit number
	DEB.05	The participants would want the system to define the correctness of her rules
Computers can be Mindless	COMP.02	The participant deems the system as mindless - i.e., it cannot do smart things
	COMP.04	"Computers need you to explain them things"
	ABSTR.07	The participant stated that she has to define what is true and false in the abstract condition
	SYST.04	The participant reports her belief that the order of creation of the primitives influences the functioning of the rules
	ORDER.02	The participant reports that the order of the rule primitives should influence the debugging
	DEB.06	The participant tests the rule by placing the tiles based on how they appear in the primitives
	ABSTR.05	The participant seems to not understand that she can use the same rule for different operations (abstract)

continued from previous page

Theme 5. Learning during the interaction		
	LEARN.02	The participant seems to learn during the task (both abstract and concrete)
	LEARN.03	The participant tries to apply what she learned during the first task in the second task
	LEARN.04	The participant seems to expect version-specific features in the second task (both abstract and concrete)
Understanding the actual functioning of the system	COMP.03	Correct expectations about the system's functioning
	SYST.03	The participant understands that she can use two tiles to create a two-digit number
Theme 6. Abstract and concrete primitives influence the interaction		
Abstract primitives can be difficult but are generalizable	ABSTR.01	The participant does not understand "the last tile"
	ABSTR.02	The participant does not understand and/or appreciate the language of the abstract version
	ABSTR.06	True and false is a simple concept and does not need to be defined (it simplifies things)
	ABSTR.04	The participant understands the flexibility of the abstract version
	REAS.02	The participant spontaneously uses the language adopted in the abstract version
Concrete primitives are easy but slow	ATTIT.08	The participant is relieved by the generalizability used in the abstract version
	ABSTR.07 / COMP.02	The participant reports her belief that she has to define true and false in the abstract condition
	CONCR.01	The concrete version is easy
	CONCR.02	The concrete version is fast
	CONCR.03	The concrete version is slow and cumbersome
	STATE.01	The participant does not understand that she can use more than one state (concrete version)
	CONCR.04 / REAS.01	The participant's spontaneous reasoning is similar to the concrete primitives
Theme 7. Language can help/hinder participants' performance		
Linguistic aspects and conjunctions	CONJ.01	The conjunctions are confusing (language)
	LANG.02	The participant defines the rule in her own words
	CONJ.03	Using "while" and "when" helps in creating the rules
	CONJ.04	The participant uses other conjunctions (e.g., if) to talk about the rules spontaneously
Verbal primitives platform-related	ABSTR.01	The participant does not understand "the last tile"
	ABSTR.02	The participant does not understand and/or appreciate the language of the abstract version
	LANG.01	The participant does not understand the verbal primitives (language — both abstract and concrete)

continued from previous page

	LANG.03	The language used in the web app is different from the one used in everyday life
	LANG.05	"the tile removed" is intended as the one missing and placed by the student
	LANG.06	The participant would like a version halfway between the abstract and concrete
	OR.02	The participant would like to have OR in the events
	ABSTR.06	True and false is a simple concept and does not need to be defined (it simplifies things)
	ABSTR.08	The participant understands "the last tile"
Theme 8. How participants perceived/described the system		
Concrete primitives	CONCR.01	The concrete version is easy
	CONCR.02	The concrete version is fast
	STATE.01	The participant does not understand that she can use multiple states (concrete)
	CONCR.03	The concrete version is slow and cumbersome
Abstract primitives	ABSTR.05	The participant does not understand that she can use the same rule for different operations (abstract)
	ABSTR.04	The participant understands the flexibility of the abstract version
	ABSTR.07	The participant stated that she has to define what is true and false in the abstract condition
	UX.05	The participant uses the IoT device for visual support
Dealing with TAP	BLOCK.01	The participant would prefer a block-based solution
	SYST.01	The participant understands that she can group rules
	SYST.02	The participant understands that she can use the same rule in multiple groups
	OR.01	The participant treats the events as if they were in OR
	OR.02	The participant would like to have OR for the events
	NOT.01	The participant would like to have NOT in the events
	LANG.06	The participant would want a version halfway between the abstract and concrete
System's usability	UX.01	The "X" to delete the rules looks like it's indicating an error
	UX.02	The rules on the homepage are "not attractive"
	UX.04	The participant does not understand that she can create multiple rules
	RULE.01	The participant wants to use the IoT device to create the rules
	IoT.01	The IoT device is appealing
Using the system in real world	CHILD.05	The participant would make students debug the rules
	DEB.01	The participant understands she can't debug with pen and paper
	LANG.04	The participant would like the platform in English when she teaches in English
	IoT.02	The IoT device is adequate for children with specific learning disabilities
	IoT.03	The IoT device is adequate for young children
Theme 9. Attitude during the interaction		
	ATTIT.01	Participant feels it is possible to learn while using the platform
	ATTIT.02	It takes time to create rules (both abstract and concrete)
	ATTIT.03	It's complicated to create rules (both abstract and concrete)

continued from previous page

DEB.03	The participant debugs a single aspect before creating the whole rule (to verify her reasoning is correct - abstract)
DEB.04	The participant debugs using different tiles within the same category (alternative case – abstract)
ATTIT.04	The participant express frustration while creating rules
ATTIT.05	The participant is hyped even though she finds the task difficult
ATTIT.06	The participant expresses that students would be better at using the EUD tool
ATTIT.07	The participant starts from the action because "it's easier"
ATTIT.08	The participant defines herself as "not technological"
ATTIT.09	The participant is relieved because of the generalizability of the abstract version
ATTIT.10	The participant reports that customizing the IoT tool is interesting
CHILD.05	The participant would make students debug the rules

Appendix K

Table 9: The table shows the results of the normality (*Shapiro-Wilk*) and homoscedasticity (*Brown-Forsythe* and *Levene*) tests performed on the ambiguities detected ("ambiguity_n") and questions generated ("question_n") considering intent recognition ("intent_yes", "intent_no"), for completeness ("complete", "incomplete"), and clarity ("easy", "moderate", "difficult") incomplete rules.

	test	data	statistic	value	p
ambiguity_n	<i>Shapiro-Wilk</i>	intent_yes	<i>W</i>	0.90	0.01
		intent_no	<i>W</i>	0.91	0.01
	<i>Brown-Forsythe</i>		<i>F</i> (1, 54)	0.45	0.51
	<i>Shapiro-Wilk</i>	incomplete	<i>W</i>	0.89	.001
		complete	<i>W</i>	0.95	0.42
	<i>Brown-Forsythe</i>		<i>F</i> (1, 54)	0.83	0.37
	<i>Shapiro-Wilk</i>	easy	<i>W</i>	0.83	.001
		moderate	<i>W</i>	0.95	0.19
		difficult	<i>W</i>	0.92	0.25
	<i>Levene</i>		<i>F</i> (2, 53)	0.38	0.69
question_n	<i>Shapiro-Wilk</i>	intent_yes	<i>W</i>	0.90	0.01
		intent_no	<i>W</i>	0.79	0.01
	<i>Brown-Forsythe</i>		<i>F</i> (1, 54)	0.97	0.33
	<i>Shapiro-Wilk</i>	incomplete	<i>W</i>	0.90	0.01
		complete	<i>W</i>	0.86	0.01
	<i>Brown-Forsythe</i>		<i>F</i> (1, 54)	0.14	0.71
	<i>Shapiro-Wilk</i>	easy	<i>W</i>	0.91	0.09
		moderate	<i>W</i>	0.81	.001
		difficult	<i>W</i>	0.85	0.03
	<i>Levene</i>		<i>F</i> (2, 53)	5.195	0.01

Appendix L

Table 10: Ambiguities detected and level of clarity. Post hoc comparisons using the *Bonferroni* correction. The table presents mean differences between conditions, standard errors (SE), and adjusted p-values.

contrast		mean diff	SE	t	p _{bonf}
difficult	easy	−1.18	0.58	−2.02	0.15
	moderate	−1.41	0.54	−2.62	0.03
easy	moderate	−0.23	0.48	−0.48	1.00

Appendix M

Table 11: User-generated instructions before and after rounds ("r") of interaction with the interface.

id	r	scenario	user-generated instruction	refined instruction
P1	1	video1	<i>Turn on the light when you enter the bathroom and turn it off when you leave the bathroom and flush the toilet.</i>	Light activation: When the bathroom door is opened (entry). Light de-activation and flushing: When the door is closed (exit), without delay.
P1	2	self-defined	<i>When I wake up to go to the daily center warm the milk.</i>	Every weekday, at 7:30 AM, when the alarm rings, the system will turn on the stove to heat the milk.
P1	3	self-defined	<i>Alexa turn on the washing machine and puts the 30-minute short.</i>	Alexa prepares the washing machine: Turns it on and asks which program to set (e.g., 30-minute short cycle). This happens before you leave for athletics. Start the wash: When you return, tell Alexa, "I'm back, start the washing machine," and it starts the chosen program. Automatic shutdown: The washing machine turns off automatically after the wash.
P2	1	video1	<i>The girl that is going to the bathroom needs to turn on the light.</i>	When the motion sensor detects the girl entering the bathroom, the light turns on automatically. The light stays on until the girl manually turns it off using the switch.
P2	2	video2	The window doesn't open from outside, call the police.	If the window opening sensor detects an attempt to open it: 1. An automatic call to the police is made immediately. 2. A notification is sent to the iPad and computer. 3. A sound alarm is activated in the house to scare the intruder.
P2	3	video3	<i>When the house is cold the fire turns on.</i>	The smart fireplace automatically turns on when the temperature drops below 18°C. The smart fireplace automatically turns off when the temperature exceeds 22°C. This rule is always active, at any time of the day.
P2	4	video4	<i>When the pot overheats the fire detector activates.</i>	When the pot reaches 200°C, the fire detector activates, and the system automatically turns off the stove using a smart plug.
P3	1	video1	<i>Is night, she goes to the bathroom, the light turns on, and she brushes her teeth.</i>	At night, when a person enters the bathroom, the light turns on automatically. After 5 minutes, a scent diffuser activates to make the environment more pleasant. When the person leaves the bathroom, the light turns off.
P3	2	video2	<i>Is night dark, close the window.</i>	If it's night and the room is dark, close the window only if it's open. Send a notification message when the window is closed.
P4	1	video1	<i>Turn on when it's dark.</i>	The bathroom light turns on automatically when a light sensor detects it's dark and turns off automatically when the brightness returns to sufficient levels. This rule is active at any time of the day.

Appendix N

Table 12: Summary of UEQ-S scores (divided by scales and the global score) collected from the caregivers ($N = 4$).

UEQ-S scales	M	SD	N
<i>Pragmatic Quality</i>	0.81	2.18	4
<i>Hedonic Quality</i>	2.00	0.94	4
Overall	1.41	1.53	4

Table 13: Descriptive statistics of UEQ-S score per item (it.). Interpretation guidelines (Schrepp et al., 2017): values between -0.80 and -0.80 represent a neutral evaluation of the corresponding scale, values > 0.8 represent a positive evaluation, and values < -0.8 represent a negative evaluation. The range of the scales was between -3 ("negative pole") and $+3$ ("positive pole").

it.	M	SD	N	negative p.	positive p.	scale
1	1.75	2.50	4	obstructive	supportive	pragmatic
2	0.25	3.20	4	complicated	easy	pragmatic
3	0.00	2.16	4	inefficient	efficient	pragmatic
4	1.25	2.06	4	confusing	clear	pragmatic
5	1.25	2.06	4	boring	exciting	hedonic
6	2.50	1.00	4	not interesting	interesting	hedonic
7	1.75	1.50	4	conventional	inventive	hedonic
8	2.50	0.58	4	usual	leading edge	hedonic

