Adv Mobile Learn Educ Res, 2022, 2(1): 234-245
DOI: 10.25082/AMLER.2022.01.007

RESEARCH ARTICLE

Forms of communications in scratch and the SOLO taxonomy

Anastasios Ladias[1] **Theodoros Karvounidis**[2*] **Dimitrios Ladias**[3]

[1] Former Bureau of School Directors, Ministry of Education, Attica, Greece
[2] Department of Informatics, University of Piraeus, Piraeus, Greece
[3] Department of Informatics, National and Kapodistrian University of Athens, Athens, Greece

Correspondence to: Theodoros Karvounidis, Department of Informatics, University of Piraeus, Piraeus, Greece; Email: tkarv@unipi.gr

Received: January 4, 2022;
Accepted: January 26, 2022;
Published: January 29, 2022.

Citation: Ladias, A., Karvounidis, T., & Ladias, D. (2022). Forms of communications in scratch and the SOLO taxonomy. *Adv Mobile Learn Educ Res*, **2**(1), 234-245.
https://doi.org/10.25082/AMLER.2022.01.007

Abstract: This paper attempts to categorize parameters of communication between codes observed in the Scratch programming environment, using the SOLO taxonomy. These parameters are the form of communication (within an object, between different objects and to serve external devices scenarios), the mechanism used for communication (Polling and Interrupt techniques) and the ratio between the number of transmitters and receivers, which will be considered in a future work. Implementing this categorization in a two-dimensional table of representative codes for each case is formed. In this table, one dimension corresponds to the forms of communication and the other dimension to the mechanisms used. The ranking of the codes in each of the dimensions is done by means of the levels of the SOLO taxonomy. The table can be used to develop criteria for assessing the qualitative characteristics of the codes produced by students within a broader assessment system.

Keywords: scratch, SOLO taxonomy, forms and mechanisms of communication

1 Introduction

Scratch is a tile-based visual programming learning environment in which the novice programmer can manage multimedia elements through code -with relative ease- and engage in authentic digital projects by creating animations, simulations and games (Resnick et al., 2009). In the Scratch programming environment, code consists of modulars parts (scenarios) that are distributed in objects and are triggered by events. During the program operation, the need for communication between the scenarios often arises. In Scratch, the ability to visually program with tiles in combination with the variety of forms of communication it can support (Moiseenko et al., 2015) in the light of the pedagogical approach of "emerging literacy" (Papanelinas, 2010), makes it an ideal tool for novice programmers on the one hand to introduce each form of communication and on the other hand for comparisons between the different forms of communication.

The aim of this paper is to investigate the various forms of communication offered in Scratch, to categorize them using the SOLO classification and to integrate them into a code evaluation framework (Karvounidis et al., 2017). Examples of the different forms of communication in Scratch which will be considered in this work are communication between scenarios (a) that exist within an object, (b) that exist in different objects, (c) that operate peripherals which either serve automation interfaces (or Internet of Things) or human interaction. The aforementioned forms of communication will be considered in relation to the mechanisms by which communication is implemented in Scratch, which is achieved either by the Interrupt technique using events and messages or by the Polling technique using memory sharing. In the Interrupt technique it is carried out with a specific "when a request is received" scenario (Figure 1a), while in the Polling technique a perpetual loop constantly asks whether a request has been perceived (Figure 1b).

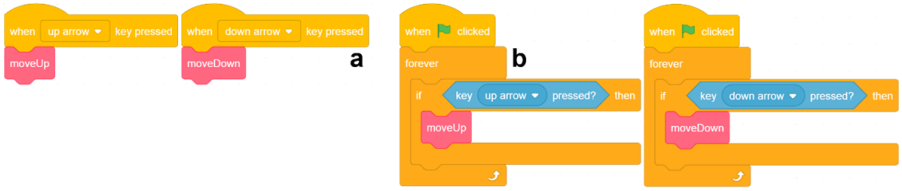

Figure 1 Examples of scenarios that detect the pressing of the "up arrow" and "down arrow" keys by (a) Interrupt (when) and (b) Polling (if) techniques

All previous forms of communication are related to the communication of a transmitter to one or more receivers. In a forthcoming work each of the previous forms of communication will be examined in the light of the different transmitter-receiver ratios used.

2 Communication using the Interrupt technique

2.1 Pseudo-communication / monologues

At the MOOC for Scratch "Code Yourself! An Introduction to Programming" (Manataki & de Kereki, 2015), one of the introductory examples of simple programming was a program of a type as such (Figure 2a) where two objects "conversed". In such a code we find that it is an "apparent communication" that is presented with synchronized monologues between two "automatons". An even more imperfect communication attempt is that of the Figure 2b, where the monologues on both sides of the objects are lacking synchronization.

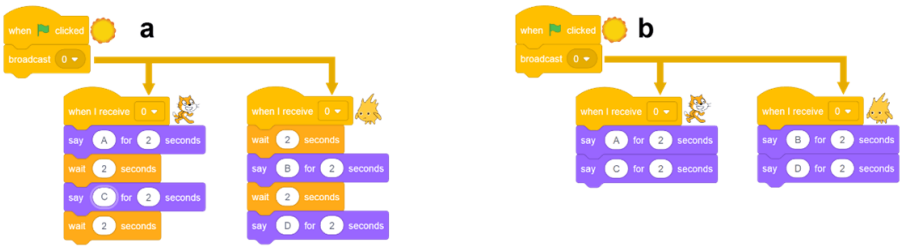

Figure 2 Phenomenal communication between scenarios of different objects

2.2 The communication deficit as communication

For reasons of completeness, it should be mentioned that silence/lack of transmission of information as a form of communication in the logic of "no news, good news".

2.3 Communication-dialogue between scenarios of different objects

The code that simulates an ordinary dialogue between two entities (people, objects, scenarios) corresponds to the one shown in Figure 3a, where following the program flow (with arrows) the dialogue A-B-C-D is completed. The use of the procedure "do…" instead of the command "say…" is being done on the one hand for reasons of future generalization and on the other hand to avoid matching of the communication between the scenarios with the appearance of program statements to the user. In addition, as shown in Figure 3b, by making good use of the mode of operation of the "transmit and wait" command (which creates a stack of calls between scenarios) it is possible, after the first phase of the dialogue (A-B-C-D), to follow the secondary statements (E-F-G) of the participants in the dialogue.

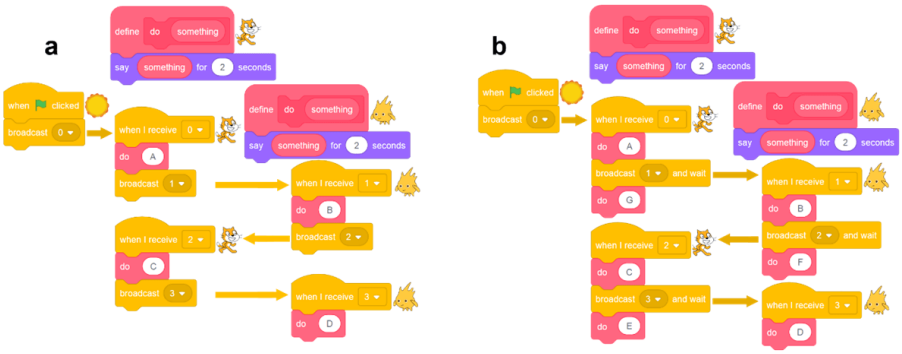

Figure 3 Dialog between scenarios of different objects using the Interrupt technique

A stack of calls is also created in the recursive way of the code of Figure 4, where the dialogue takes place and by exploiting the properties of the string as an array of characters (Karvounidis et al., 2019; Rozou et al., 2017).

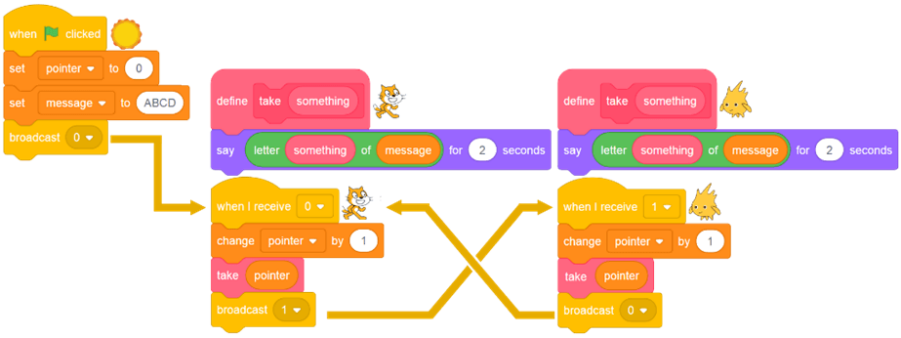

Figure 4 Dialog between scenarios of different objects using an iterative algorithm and the use of strings with the Interrupt technique

2.4 Communication-dialogue between scenarios of the same subject

If (as before) instead of communication between scenarios belonging to different objects, the communication was done as in an intercommunication (communication between scenarios of the same object), then the codes of Figure 3 would be converted into those of the Figure 5. For didactic purposes in the "do" procedure, the command "say..." is replaced by the command "think...".

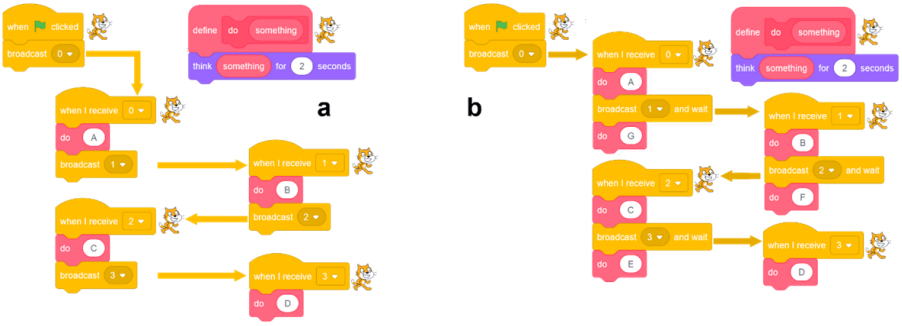

Figure 5 Dialogue between scenarios of the same object with the Interrupt technique

2.5 Forms of automation communication and computer-human interaction

In the context of communication with the Interrupt technique, a computer-based automation is included. In this case, a sensor of an external device sends information to the computer (input), which processes it based on an algorithm, the result of which causes an action (output) on an activator of (the same or another) external device. This type of communication is shown in the code of the Figure 6a, where after initialization (scenario "when the green flag is clicked"), the events are detected and processed by the algorithm (the three scenarios "when distance. . . ") that leads in determining the activator / motor power. An example of a computer-to-human communication is the scenarios of the code in Figure 6b which has as input the mouse clicks or the question-answer via a keyboard and as the output the screen.

3 Communication using the Polling technique

3.1 Communication-dialogue between scenarios of the same subject

Figure 5a shows the code that simulates an ordinary dialogue between two scenarios (of the same object) implemented with the Interrupt technique. The same dialogue implemented with the Polling technique (Ladias et al., 2019) is that of Figure 7a. In the scenario "when I receive 0", all of the values that the variable "signal" (shared memory) can take are continuously ("forever") examined, sequentially with the "if-then" instructions; in this way the requested

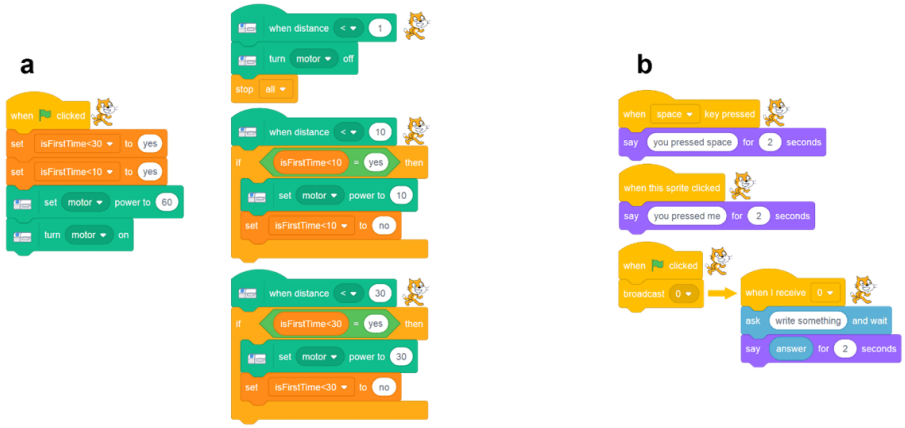

Figure 6 Forms of communication with the Interrupt technique of: (a) an automation and (b) of user interface

value is identified. In this algorithm, the search is being performed sequentially. In Scratch there is the possibility to perform the search simultaneously (Figure 7b) by scenarios which are executed in parallel (the "when I receive 0" scenarios) separately for each of all the values that the shared memory (the variable "signal") may take (Dijkstra, 1968; Bustard, 1990).

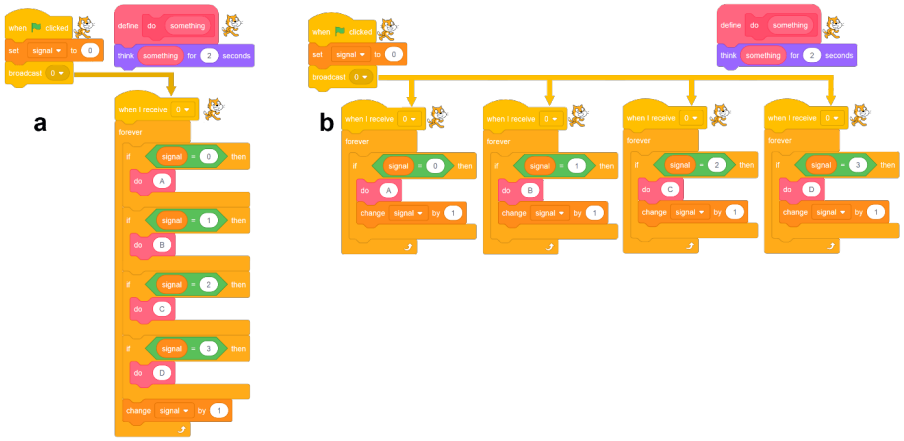

Figure 7 Dialogue between scenarios of the same object via the Polling technique: (a) Serial and (b) Parallel

3.2 Communication-dialogue between scenarios of different objects

For the case that the communicating scenarios belong to different objects then the corresponding codes are adapted as in Figure 8. The difference from the previous algorithm (of the same object) is imposed by the fact that in Scratch the procedures are defined and operate exclusively inside an object.

3.3 Forms of communication in an automation

The corresponding codes of the examples (with the Interrupt technique) of automation and user interaction (Figure 6a) in a polling technique implementation are shown for serial detection in Figure 9a and for the parallel scenario detection in Figure 9b.

3.4 Forms of automation communication and computer-to-human interaction

Codes of corresponding examples (with the Interrupt technique) of user interaction (Figure 6a implemented with the Polling technique are shown for serial detection in Figure 10a and for parallel detection Figure 10b.

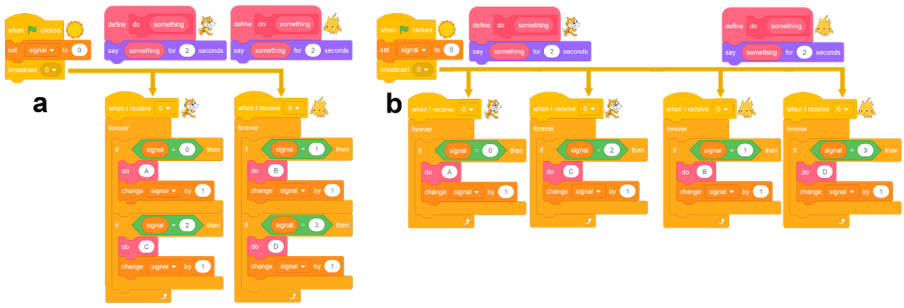

Figure 8 Dialogue between scenarios of different objects via the Polling technique: (a) Serial and (b) Parallel

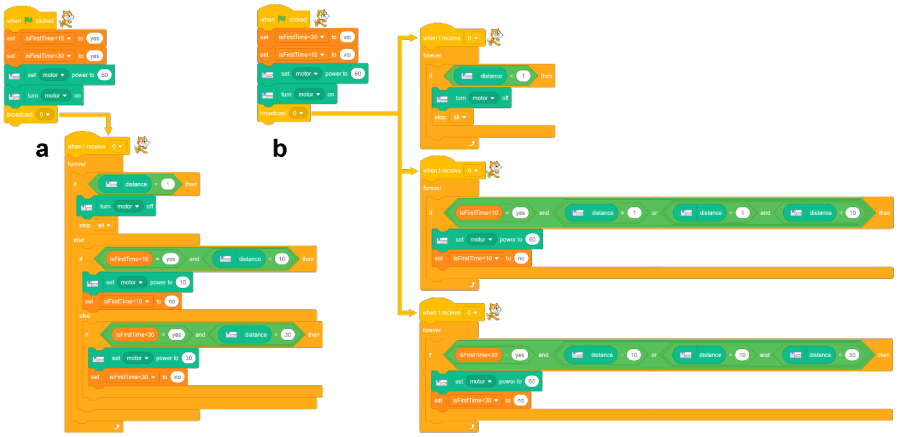

Figure 9 Form of communication in automata using the Polling technique: (a) Serial and (b) Parallel

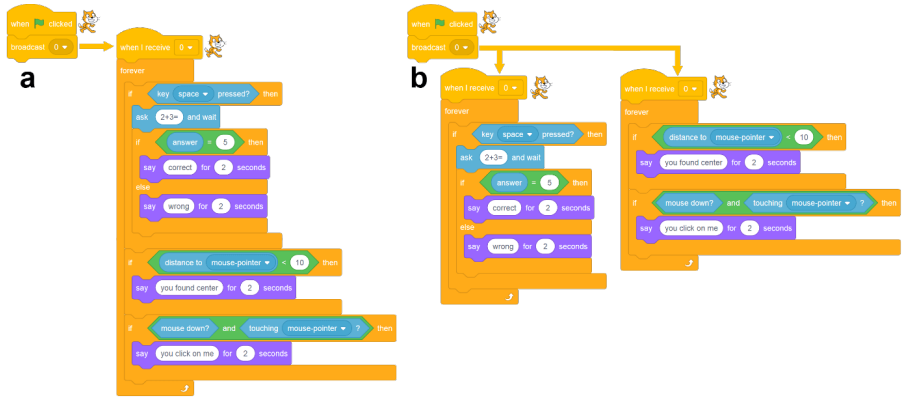

Figure 10 Computer-to-human communication with the Polling technique: (a) Serial and (b) Parallel

4 Communication using combinations of Interrupt and Polling techniques

During communication, special needs arise such as setting priorities in the service of requests that can be solved with (simple or complex) combinations of events (messages / requests) and shared memory. It should be noted that the scenarios that serve messages and requests ("when…") are satisfied with high priority (Interrupt technique) while according to the Polling technique the scenarios that detect a situation with the command "if…" they wait their turn to be satisfied. In addition, in the Polling technique there are differences in the priority of service which depend on the application of parallel or serial algorithm.

4.1 Simple combinations of using messages and shared memory between scenarios of different objects

Figure 11 shows two examples of communication between scenarios of different objects. In these, the use of messages ("when I receive 0" and "when I receive 1" scenarios is combined to locate the requests with the search with Polling technique of individual requests inside some scenarios. More specifically, the individual search and service of requests is done in (a) serially while in (b) in parallel.

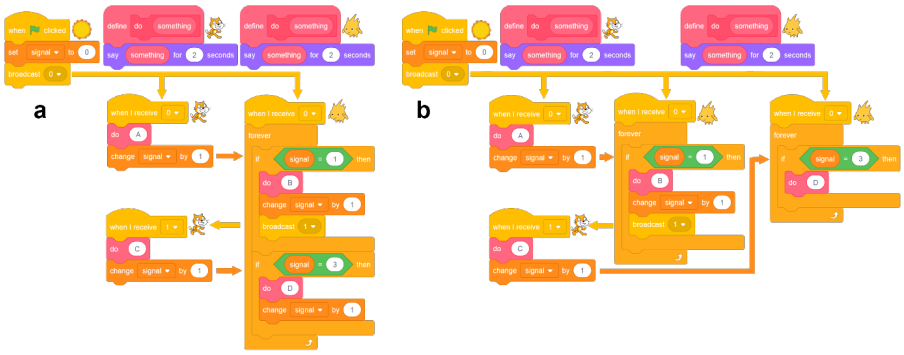

Figure 11 Simple combinations using messages and shared memory between scenarios of different objects by applying individual algorithms (a) serially and (b) in parallel

4.2 Simple combinations of using messages and shared memory for scenarios within an object

Similar to the previous examples, but for communicating scenarios located inside an object, they are presented in Figure 12 also with individual search and service of requests to be done in (a) serial while in (b) in parallel.

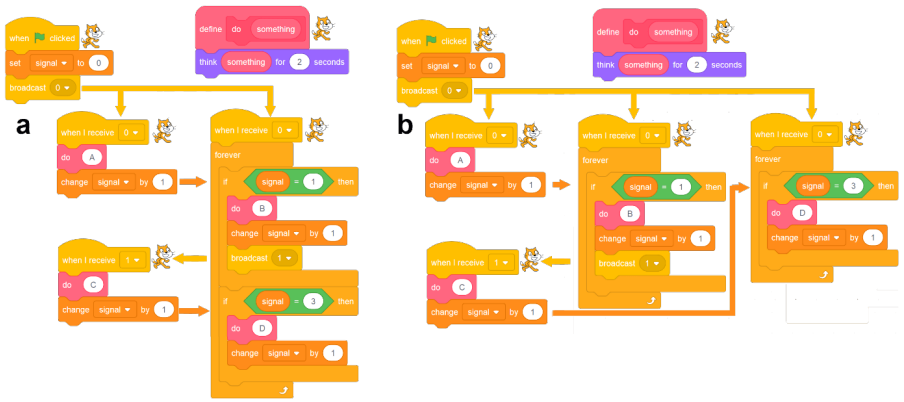

Figure 12 Simple combinations using messages and shared memory between scenarios within an object by applying individual algorithms (a) serially and (b) in parallel

4.3 Simple combinations of using Interrupt and shared communication memory for management scenarios of external automation devices

Figure 13 shows two examples of communication with external devices. In order to locate the requests, the Interrupt technique (the scenario "when distance < 1") is combined with the Polling technique for the other cases with their search and service to be done in (a) serially, while in (b) in parallel.

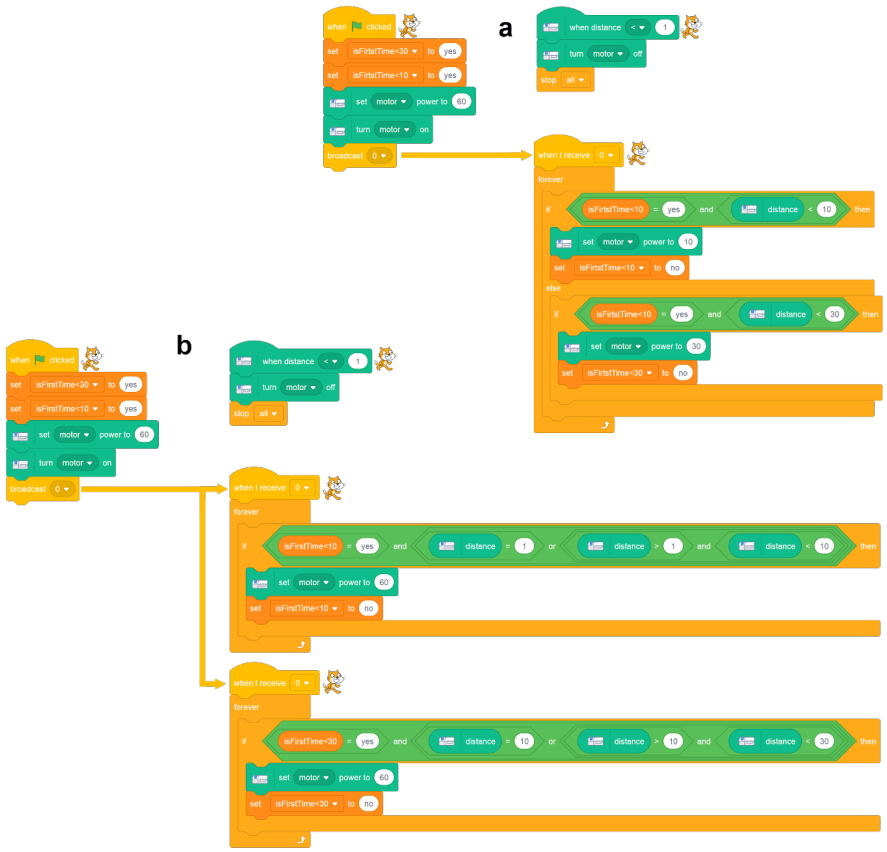

Figure 13 Simple combinations of using Interrupt and shared communication memory for management scenarios of external automation devices with application of individual algorithms (a) serially and (b) in parallel

4.4 Simple combinations of using Interrupt and shared memory communication in the human-machine interface

Two examples of communication in the human-machine interface are shown in Figure 14. In these examples, the Interrupt technique (the "when ... key pressed" scenarios) is combined for the detection of requests on the one hand with the search using the polling technique of pressing the "right arrow" and "left arrow" keys which is done serially in (a) and in parallel (b).

4.5 Complex combinations of message usage and shared memory for scripts that belong to either the same object or different objects

Figure 15 shows an example where from the three parallel scenarios ("when I receive 0") the middle one (due to Interrupt technique) is immediately activated, which while informing the shared variable "signal" via the Polling technique, it activates the right scenario "when I receive 0". Similarly, it activates the left scenario "when I receive 0", to end up using the Interrupt technique to activate the scenario "when I receive 1". In the Figure 15a the communication takes place inside an object while in the Figure 15b one the communication takes place between scenarios of different objects.

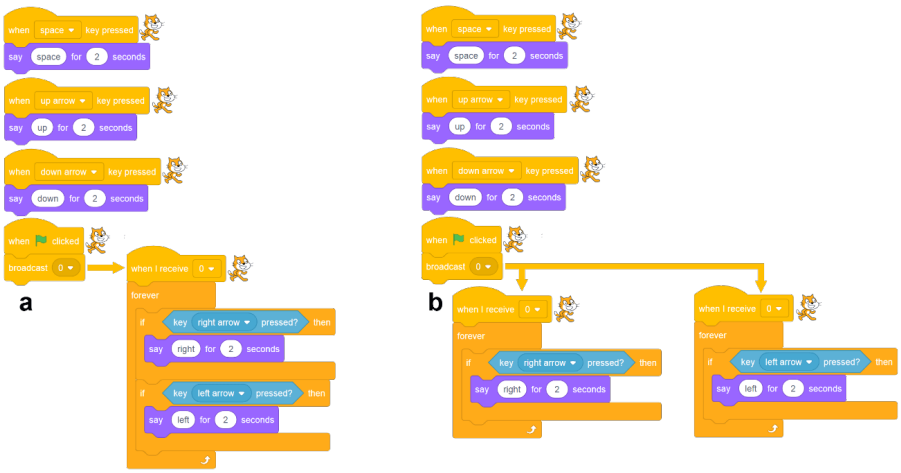

Figure 14 Simple combinations of using Interrupt and Polling for communication in the human-machine interface with application of (a) serial and (b) parallel algorithm

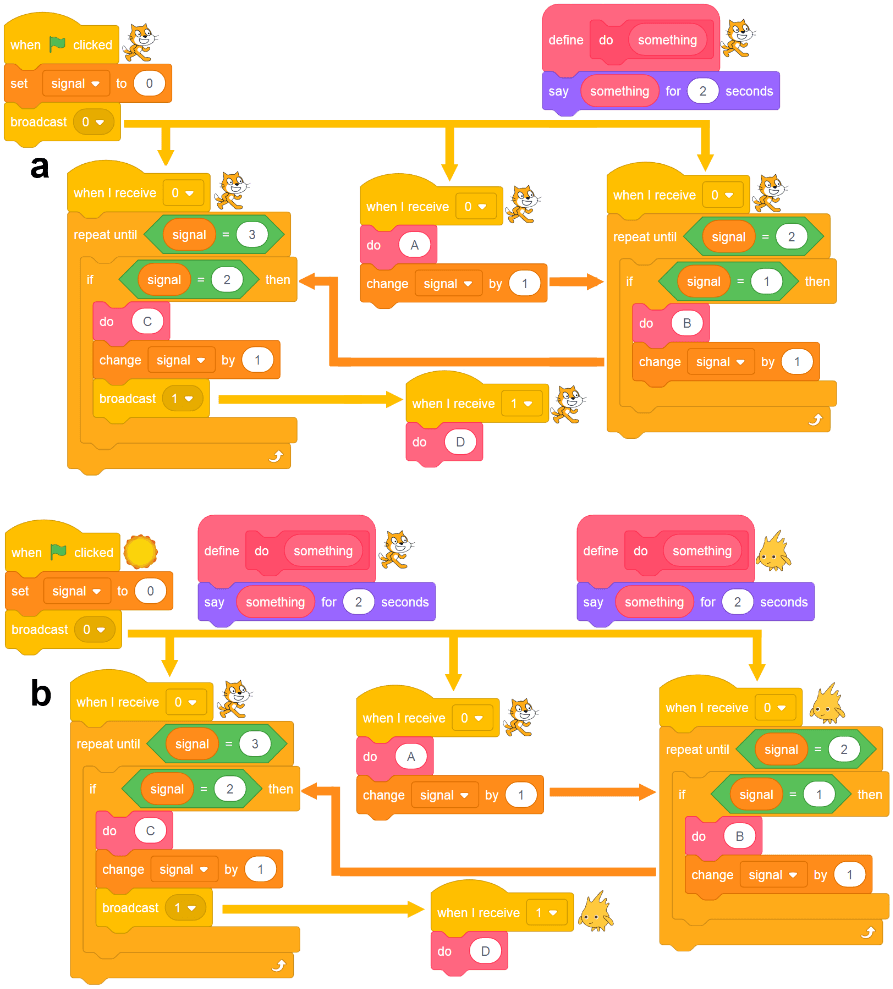

Figure 15 Example of a complex combination of Interrupt and Polling techniques between scenarios (a) inside an object and (b) between different objects

4.6 Complex combinations of message usage and shared communication memory of external automation device management scenarios

An example of communication where two external devices (e.g., a WeDo and a micro: bit) combine their sensors and actuators by communicating via computer with Scratch, is shown in Figure 16b.

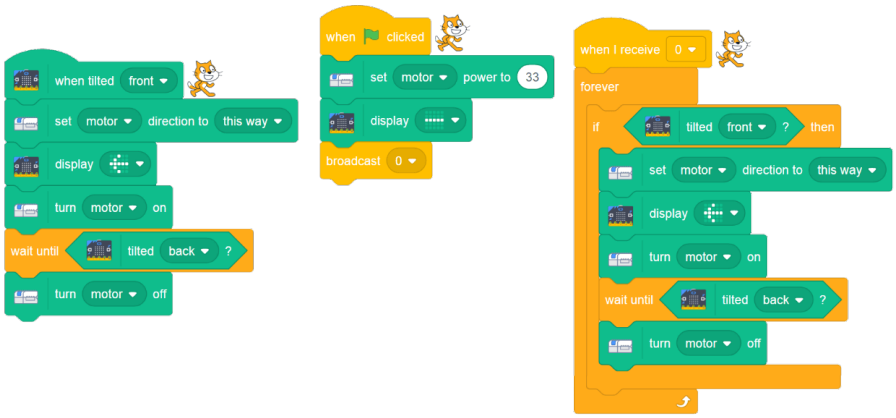

Figure 16 Example of communication between different external devices (WeDo and micro: bit) in an automation with either Interrupt technique or Polling technique

4.7 Complex combinations of messaging and shared memory communication with human-machine interface

Two examples of communication with priorities in locating and servicing requests, concerning human-machine interface, are shown in Figure 17. In these examples (with the Interrupt technique) the requests have priority caused by pressing the "space", "up arrow" and "down arrow" keys. But even among these requests there is a priority ranking that is achieved with the nested "if. . . " in the scenario "when any key pressed". Moreover requests caused by pressing the "right arrow" and "left arrow" keys have a lower priority (Polling technique). The priorities between them are determined by the serial search (Figure 17a) or the search with parallel scenarios (Figure 17b).

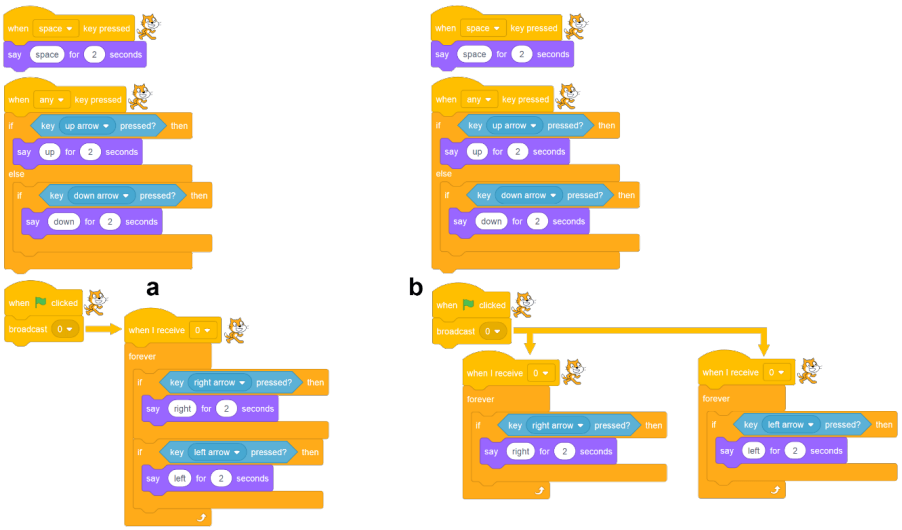

Figure 17 Examples of complex human-machine communication with priority setting in the detection of requests using (a) serial scenario and (b) parallel scenarios

Moreover, in this category can be included code for scenarios that create "improvised commands" that serve some necessary manipulations that do not exist in the Scratch programming environment, such as a version of "when MouseUp", shown in Figure 18.

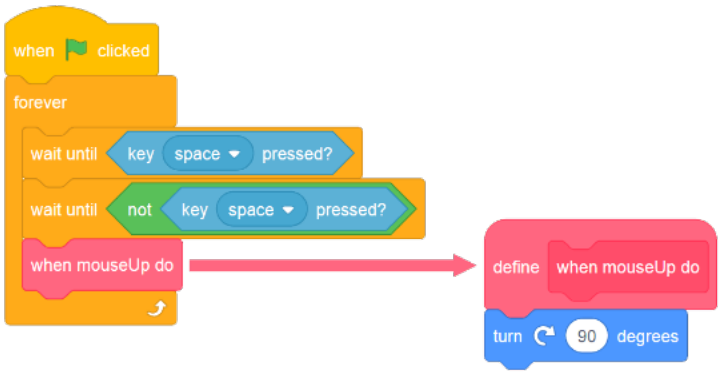

Figure 18 Scenario that serves the interface of programming environment and user through the control of the mouse, providing the possibility of action when the mouse is released / returned (When Mouse Up) by pressing it

5 SOLO taxonomy and its application to communication in Scratch

5.1 The SOLO taxonomy

The Structure of Observed Learning Outcomes taxonomy of Biggs & Collis (1982) proposes the assessment of knowledge based on the structure of the observed learning outcome, ranking these outcomes into five hierarchical levels: (a) The pre-structural in which reference is made or use is made of unconnected and unorganized information that is not meaningful; (b) The unistructural level, where a limited perspective is observed, mainly one element or aspect is used or emphasized while other components are omitted and no significant connections are made between the parts; (c) The multi-structural level, in which there is a multi-point perspective - several relevant elements or aspects are used or recognized; but there are no significant connections and a complete picture has not yet been formed; (d) The relational level, in which there is a holistic perspective where meta-connections between parts are perceived and the importance of parts in relation to the whole is demonstrated and appreciated; and (e) The level of extensive abstract, in which in addition to the features of the previous relational level. The SOLO taxonomy can be implemented in the assessment of a code (Lister et al., 2006; de Raadt, 2007; Bellou & Mikropoulos, 2008; Jimoyiannis, 2011).

Next, it will be an attempt to categorize the various forms of communication in Scratch at the levels of the SOLO taxonomy according to the mechanisms they use.

5.2 The use of the SOLO taxonomy in the categorization of codes of communication formats

At the pre-structural level of the SOLO taxonomy, the relevant paragraph with "the communication deficit" on the one hand, and the programs of the paragraph "pseudo-communication-monologues" in Figure 2 on the other hand are included. In the unistructural level are ranked the programs concerning the communication of scenarios which exist only inside an object (Figure 5, 7, 12 and 15a) in which a one-dimensional form of communication is implemented that ignores the communication with other objects or external devices. The multi-structural level includes programs that involve communication between scenarios existing in different objects (Figure 3, 4, 8, 11 and 15b), which complete the communication between entities inside the programming environment. However, they do not include the communication between the computer and the external environment. At the relational level are ranked those programs that deal with serving the requests originating from external devices with sensors and automation activators. Examples of this category are the programs in Figure 6a, 9, 13 and 16). In the level of the extended abstract, in addition to the characteristics of the relational level, the codes that manage the human-machine interface are ranked (Figure 6b, 10, 14 and 17).

5.3 The implementation of the SOLO taxonomy in the categorization of communication mechanism codes

The simplest communication mechanism is that of sending a message/request from the transmitter to the receiver. These scenarios are ranked at the unistructural level and correspond

to the programs in Figure 3, 4, 6a and 6b. The scenarios that communicate with each other via the Polling technique and use the shared memory are ranked in the multi-structural level; they correspond to the programs of Figure 7, 8, 9 and 10. The scenarios that use simple combinations of the Interrupt and Polling techniques are ranked in the relational level and they are described in the Figure 11, 12, 13 and 14. The scenarios using complex combinations of the Interrupt techniques are ranked at the extended generalization level and they are found in Figure 15a, 15b, 16 and 17.

5.4 Combining the previous communication categorizations using the SOLO taxonomy

The two previous categorizations can be combined into a two-dimensional table (Table 1) and provide a more complete representation of the communication scenario categorizations in Scratch. The vertical dimension of the table lists the levels of SOLO in relation to the form of communication, while the horizontal dimension lists the levels of SOLO in relation to the communication mechanisms used.

This combination allows each of the illustrative examples mentioned before to correspond to a position in the table; thus, its weight can be assessed and graded in terms of the SOLO taxonomy. The result is to provide the teacher with a tool which, on the one hand, he can use as a guide when assessing his students' codes and, on the other hand, to consult the table so that he can plan his personal curriculum in terms of teaching the communication between codes (Giannakos et al., 2014; Doukakis & Papalaskari, 2019).

Table 1 Combination of the categorizations of the forms and mechanisms of the communication scenarios in Scratch

Crowd ratio of transmitters - receivers			The SOLO taxonomy level depending on the communication mechanism used			
			Interrupt technique	Polling technique	Simple combinations	Complex combinations
1:1 or 1:N			Unistructural	Multi-structural	Relational	Extended abstract
The SOLO taxonomy level depending on the form of communication used	For the human-machine interface	Extended abstract	Figure 6b	Figure 10	Figure 14	Figure 17
	With external devices	Relational	Figure 6a	Figure 9	Figure 13	Figure 16
	From object to object	Multi-structural	Figure 3, 4	Figure 8	Figure 11	Figure 15b
	Within an object	Unistructural	Figure 5	Figure 7	Figure 12	Figure 15a
	Incomplete communication	Pre-structural			Figure 2	

6 Conclusions – Future steps

This work is part of a Scratch code evaluation project. By matching the forms and the mechanisms of communication between the scenarios in Scratch, a criterion can be developed - in the light of the above data - in the assessment system via which the dimension of communication in the code written by the students can be evaluated.

This table can also be used by the developers of a Curriculum related to programming learning. The table may help them to suggest a main navigation path in this educational content, following subsequent neighboring cells of the table. In this way they allow on the one hand exploratory learning with reduced guidance and help from the teacher (scaffolding) and on the other hand this path to follow a spiral approach. An example of a main navigation route could be the order of the codes of Figure 3, 4, 5, 7, 8, 9, 6a, 6b, 10...

In addition, the teacher can use this table as a guide to define alternative navigation paths in this educational content which he/she will follow to teach targeted individual parts of the content. These individual parts can be adapted to the conditions of the teacher's classroom and fit his/her personal perceptions. Examples of such alternative routes could be teaching the series of codes in Figure 2, 7, 8, 9, 10 if the teacher wanted to teach the Polling technique or the series of codes in Figure 3, 8, 11 and 15b if the teacher wanted to teach communication between scenarios of different subjects.

As shown in the upper left cell of Table 1, all data refer to the communication of a transmitter to one or more receivers. Within a next step of our research, each of the previous forms of communication in this table will be examined in the light of different transmitter-receiver ratios, thus producing superimposed two-dimensional tables for each ratio, forming a three-dimensional table for evaluating communication codes in Scratch.

References

Biggs, J. B., & Collis, K. F. (1982). Evaluating the quality of learning. The SOLO taxonomy. NY: Academic Press.

Bellou, I., & Mikropoulos, A. (2008). A method for the Hierarchical Assessment of the Programming Knowledge, 4th Panhellenic Conference on Didactics of Informatics, 111-120.

Bustard, W. D. (1990). Concepts of Concurrent Programming. Software Engineering Institute / Carnegie Mellon University (SEI-CM-24).
https://apps.dtic.mil/sti/pdfs/ADA223897.pdf

Dijkstra, E. W. (1968). Cooperating Sequential Processes. Programming Languages, F. Genuys, ed. Academic Press, 43-112.
https://doi.org/10.1007/978-1-4757-3472-0_2

de Raadt, M. (2007). A Review of Australasian Investigations into Problem Solving and the Novice Programmer. Computer Science Education, 17(3), 201-213.
https://doi.org/10.1080/08993400701538104

Doukakis, S., & Papalaskari, M. A. (2019). Scaffolding Technological Pedagogical Content Knowledge (TPACK) in Computer Science Education through Learning Activity Creation. In 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), 1-5.
https://doi.org/10.1109/SEEDA-CECNSM.2019.8908467

Giannakos, M. N., Doukakis, S., Crompton, H., Chrisochoides, N., Adamopoulos, N., & Giannopoulou, P. (2014). Examining and mapping CS teachers' technological, pedagogical and content knowledge (TPACK) in K-12 schools. In 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, 1-7.
https://doi.org/10.1109/FIE.2014.7044406

Jimoyiannis, A. (2011). Using SOLO taxonomy to explore students' mental models of the programming variable and the assignment statement. Themes in Science & Technology Education, 4(2), 53-74.

Karvounidis, Th., Argyriou, I., Ladias, An., & Douligeris, Chr. (2017). A Design and Evaluation Framework for Visual Programming Codes. In the proceedings of the 2017 IEEE Global Engineering Education Conference (EDUCON), 2017, 999-1007.
https://doi.org/10.1109/EDUCON.2017.7942970

Karvounidis, T., Ladias, A., Ladias, D., & Douligeris, C. (2019). Kinds of loops implemented with messages in Scratch and the SOLO Taxonomy. In the proceedings of the 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), 2019, 1-5.
https://doi.org/10.1109/SEEDA-CECNSM.2019.8908420

Ladias, A., Ladias, D., & Karvounidis, T. (2019). Categorization of requests detecting in Scratch using the SOLO taxonomy, In the proceedings of the 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), 1-7.
https://doi.org/10.1109/SEEDA-CECNSM.2019.8908438

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. Proceedings of the 11th annual SIGCSE conference on Innovation and Technology in Computer Science Education, 118-122.
https://doi.org/10.1145/1140123.1140157

Manataki, A., & de Kereki, I. F. (2015). Code Yourself! An Introduction to Programming. MOOC, Coursera.

Moiseenko, A. V., Brylina, I. V., Kornienko, A. A., Berestneva, O. G., & Kabanova, N. N. (2015). Visual language as a mean of communication in the field of information technology, In the proceedings of the 6th International Conference on Information, Intelligence, Systems and Applications (IISA), 2015, 1-4.
https://doi.org/10.1109/IISA.2015.7388015

Panselinas, G. (2010). Computer literacy in the modern Greek school.
http://plirancrete.blogspot.com/2010/03/blog-post.html

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All, Communications of the ACM, 52(11), 60-67.
https://doi.org/10.1145/1592761.1592779

Rozou, M., Papadakis, S., & Ladias, A. (2017). The representation of the data, in high school students' Scratch language codes. In the proceedings of the 11th Panhellenic Conference of Informatics Teachers (PEKAP), May 5-7 2017, Chalkida, Greece.
http://synedrio.pekap.gr/praktika/11o/ergasies/anakoinoseis/pekap2017-final17.pdf