ArCo: an architecture for children to control a set of robots

Céline Jost, Brigitte Le Pévédic, Dominique Duhaut

ArCo: an architecture for children to control a set of robots*

C. Jost, B. Le Pévédic and D. Duhaut

Abstract— **This article presents ArCo which is an architecture for managing a digital environment in a context of ambient intelligence. ArCo can integrate all kinds of objects and is totally configurable. Moreover it proposes a visual programming interface, AmbiProg, which allows controlling each digital object which composes an environment. The objective of this paper is to appraise AmbiProg. An experiment was realized with 16 children around 10 years old. They were asked to realize exercises in a limited period of time. The objective was to determine if programming was easy and how many times they needed to learn scenarios creation. It appeared that they succeed tasks and were able to use AmbiProg. Programming an environment was easy as pie.**

I. Introduction

Currently the domain of ambient intelligence emphasizes as a major concern. It is not only a need but also a comfort. First, the population is ageing. There is the willing to keep the elderly at home as long as possible. Second, technology improves the quality of life. The ambient intelligence takes considerations from ubiquitous computing and human-machine interaction [1]. The human should be the core of the system [2] but currently, humans are partially taken into account. There is the willing to provide computing systems which are accepted [3] and there is the willing to be able to analyze human's actions. These analyses offer the system to adapt to the user. Actually, the ambient intelligence concerns seem to be technical [4]: how to give comfort to users without imposing technology? Thus, the technology is hidden or adapted to existing objects. Objects become more and more electronic and capable of doing complex tasks. Thus, humans own numerous complex electronic objects which work independently from each other. They need to learn how to use each object. We can imagine that, in the future, a person will have 10 or 20 remote control which allows interacting with 10 or 20 different objects. The learning process will be increasingly hard. Moreover, interactions with these objects are limited to the manufacturer proposals. These kind of computing systems are closed. They cannot be adaptable. They cannot evolve with time and new technologies. Each time a new object is added to the environment, human has to learn how to use it. But, considering that the human should be a part of the program [5], the ambient intelligence should take the user willing into account. Aarts et al [6] gave a complete overview about ambient intelligence. They explain that three main points must be considered: interoperability, heterogeneity and dynamics. We [7] created a computing system which responds to these three points. We defined a communication protocol (interoperability). Different kinds of objects can communicate together to bring a complete application (heterogeneity). Our system adds and removes dynamically objects (dynamics). Moreover, the user actively participates to the environment evolution by creating interaction scenarios through a visual programming interface.

First, this paper presents our previous work in a nutshell. At the beginning, it deals with ArCo, our computing architecture, which introduces the context of this paper. Then, it focuses on the programming language which allows controlling the environment. A first experiment has ever been realized on this interface. This paper reminds its objective and results.

Second, this paper presents the new experiment which showed that programming some scenarios is easy. Children can do it. During this experiment, children not only created some interaction scenarios but also launched their interpretation and saw the environment objects being in action.

II. Previous work

A. ArCo: the environment manager

ArCo, which is represented in Fig. 1, is an **Ar**chitecture for interaction with **Co**mpanions. The companion can be a set of communicating objects: avatars, robots, television… This definition is shared by Pesty et al [8] who give some considerations about the future relationship with these kinds of companions. There are two kinds of communicating objects in ArCo. **Actuators** make actions. They help people. They communicate with people. They give information to people. **Sensors** are responsible for analyzing the computing environment (camera, micro, temperature sensor…). They describe what happens and give information to ArCo.

Actuators and **Sensors** are **MIIME Modules**. MIIME is a XML-based protocol of communication. To be compatible with ArCo, each module must be a MIIME Module, that is to say respect the communication rules established with the server (identification, connection, disconnection…). **MIIME Modules** are independent programs. They can be described by a XML description file which indicates what kind of data they can send and what kind of actions they can do. This description file allows people knowing the capabilities of each module. This information is used by **MICE**, which is a programming environment which allows people configuring and using **MIIME Modules**.

*Research supported by French National Research Agency.

C. Jost is with the Lab-STICC computing laboratory in the University of South Brittany, France (e-mail: celine.jost@gmail.com).

B. Le Pévédic is with the Lab-STICC computing laboratory in the University of South Brittany, France (e-mail: brigitte.le-pevedic@univ-ubs.fr).

D. Duhaut is with the Lab-STICC computing laboratory in the University of South Brittany, France (e-mail: Dominique.duhaut@univ-ubs.fr).

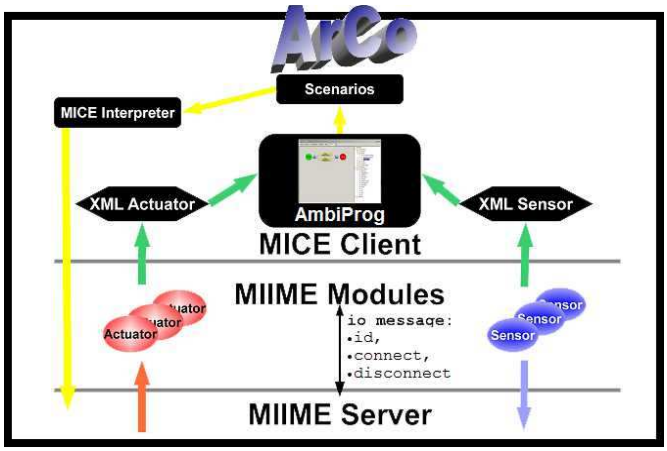

Figure 1. ArCo architecture organization

B. AmbiProg: the programming environment

It is possible to create some interaction scenarios through the graphical user interface AmbiProg. For example, an interaction scenario could be: "If the patient did not take his/her medicine, the avatar reminds the patient of doing it".

AmbiProg is composed of two main parts as show in Fig. 2. On the left, **programming area** displays graphical program between a start and a stop element. On the right, a tree contains **programming data** which are perceptions, actions and program structures.

Perceptions are the equivalent of variables in computer science. They are sent by modules which supervise sensors and retrieve associated data. A perception can be the temperature, the localization of people, the luminosity value…

Actions are the equivalent of functions in computer science. They represent the actions which can be done by actuators. For example, an action can be "robot smiles", "turn on the light"…

Program structures represent the programming language. It proposes conditional elements (if… then, if… then… else), loops (repeat, while), wait elements, break element, parallelism elements and event elements.

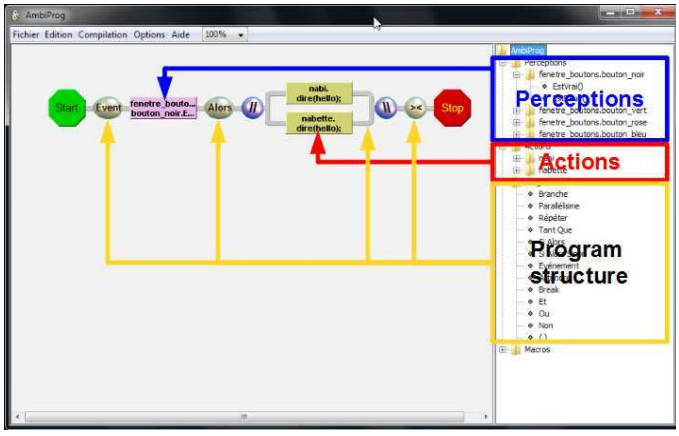

Figure 2. AmbiProg organization

AmbiProg is different from other visual programming languages like Scratch, Choregraphe (Nao) and Lego Mindstorm NXT.

Indeed they do not have the same functions. AmbiProg is designed to control a complete environment by creating interaction scenarios whereas the three others are specialized. Scratch allows creating stories. Choreographe allows controlling only Nao robots. And Lego Mindstorm NXT allows programming Logo objects and sensors.

AmbiProg is made to create interaction between all kinds of digital devices which are automatically added to AmbiProg (perceptions and actions). Its content is not fixed.

C. How to build a scenario

To create a scenario, users must drag and drop data from programming data to programming area.

Fig. 3 shows a four steps example. (1) Users have to select a data in the tree. (2) When users click on the data, some gray area highlight between the start and stop elements. It corresponds to the location where data can be placed. (3) Users drop data. (4) The graphical representation appears in the programming area.

In this example, the box represents an action.

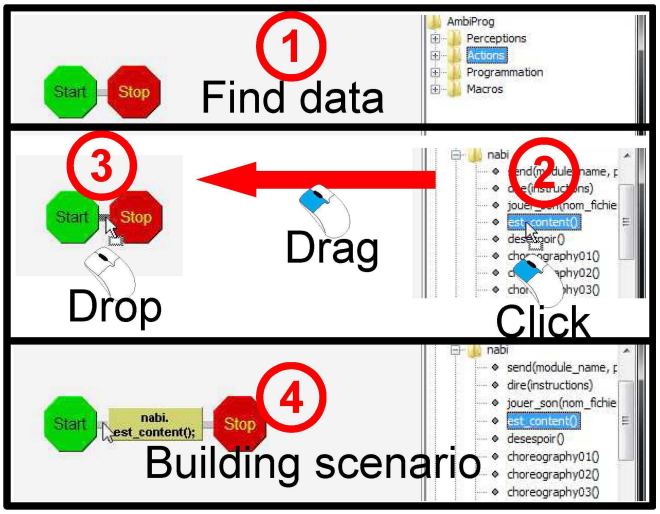

Figure 3. Programming with drag and drop

The same procedure allows users adding a program structure. Fig. 4 shows the representation of a parallelism element which allows to create at least two sub-scenario in parallel.

Figure 4. Adding a program structure

Each program structure adds some locations to drop new data. Fig. 5 shows 5 possible locations to drop an action.

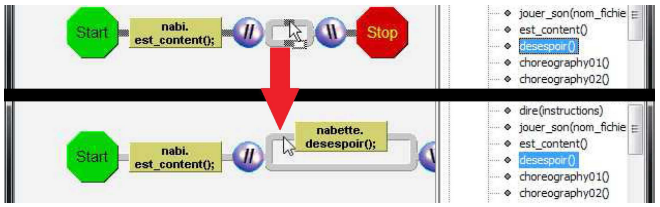

Figure 5. Visuel help to drop an action

D. The first experiment: usability and acceptability

A first experiment of AmbiProg has ever been realized [7]. The objective was to study the:

- Usability of AmbiProg: is the interface organization pertinent and easy to understand? Is the drag and drop principle judicious to programming tasks?
- Acceptability of AmbiProg: do the participants want to have this kind of interface at home in order to control their environment? Are they reluctant to this kind of interface?

This experimentation was not interested in the creation of scenarios although the participants were asked to do one. The objective was to evaluate the quality of AmbiProg out of context. It is pertinent to study first acceptability of the interface before studying scenarios creation.

This experimentation had good results. There were no statistical differences between computer scientists and other participants. Knowing computing had no impact on the learning process of the interface functioning. Moreover, AmbiProg was positively perceived by participants. They had never seen the interface before the experiment. But everyone was able to create the asked scenario in less than five minutes without showing great difficulties. Participants were favorable to use this kind of interface at home to control their environment. AmbiProg structure was validated and we decided to study the visual language itself in an ambient context.

III. EXPERIMENTATION

The objective of this experimentation was to evaluate the visual programming language in order to know if people can easily learn how to control their environment with ArCo. Each participant had to create scenarios and to launch their interpretation in order to see actuators making asked actions.

A. Global setting

Participants were pupils in the last "level" of elementary school (average age: 10.44 years old). This experimentation was realized by 16 children (8 girls and 8 boys) during one hour.

We choose young people to test ArCo because we wanted to check whether the programming language was easy to manipulate. Young people are not expert in computing, so they represent a big cross-section.

Moreover, we decided to test first with children because they do not have prejudice against technology. Thus, if children, who are motivated, do not success tasks, it is not worth testing with the elderly. We experiment step by step.

Fig. 6 illustrates the experimental setting and the materiel used during the experiment. Two participants were isolated in a room with an experimenter. The presence of two participants allowed maintaining a good level of concentration because children felt in competition. However, to avoid cheating or reciprocal influences, they were almost back to back.

They had to use a computer which displayed AmbiProg. They manipulated drag and drop with a mouse to facilitate the task. They had the possibility to control two active Nabaztag robots [13] and a tablet PC. The robots were **Actuators** and were able to make choreographies which means: rotating ears, turning on four leds with different colors and pronouncing sentences. Each participant could control the both robots. The tablet PC was a **Sensor** and displayed a basic graphical user interface with four buttons. An access point allowed communication between these five devices.

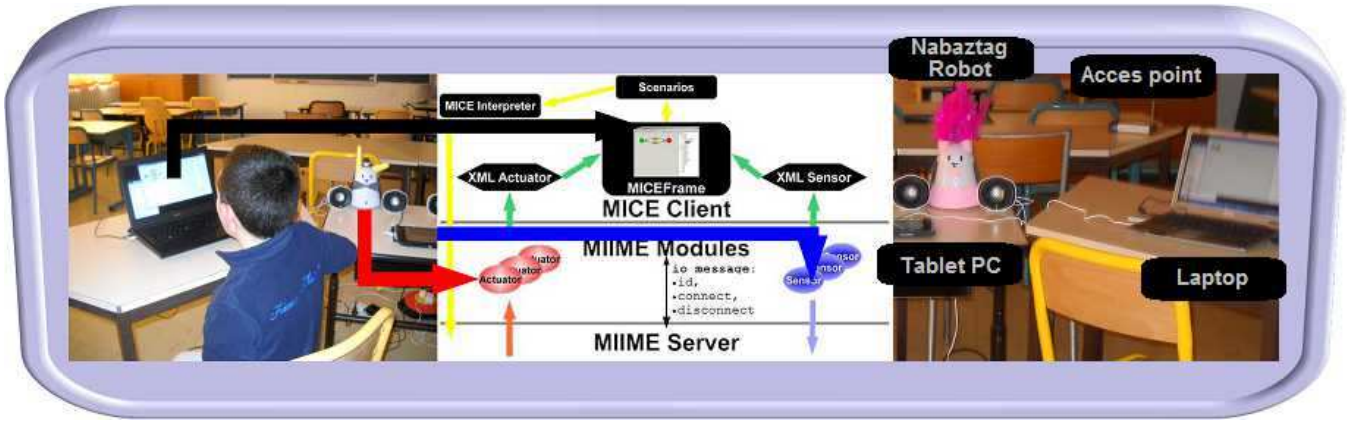

Figure 6. Experimental setting

B. Experimentation process

1) Step 1: explanation

The experimenter introduced the both robots to the participants: Nabi and Nabette. Then she explained that the tablet PC was a Sensor capable to send "Perceptions". A perception is a data about the environment. In this experimentation, there were four perceptions: "clicked on blue button", "clicked on pink button", "clicked on green button" and "clicked on black button". She explained that is was possible to control the robot by creating some scenarios. For example: "if the user has clicked on the red button, Nabette says hello".

The experimenter explained that it was possible to do it with AmbiProg. Children received a five minutes explanation about the organization of AmbiProg. It was a shorter explanation than in the first experiment.

Thus, the experimenter explained the process which would be used during all the experiment: (1) Participant had to create a scenario with respect to the given instructions. (2) Participant had to "compile" the scenario; that means clicking on the menu and saving the scenario as a predefined specific file. (3) Participant had to double-click on an executable file to launch the interpretation.

2) Step 2: exercises

Participants had to do six exercises classified in three concepts:

- Sequentiality: actions are done one after the other.
- Parallelism: several actions are done simultaneously.
- Event: actions depend on Perception. The system waits for a specific perception before doing actions.

We consider that the three concepts had almost the same difficulty level. Thus, the given order does not represent a progression of difficulty. However, there is several difficulty levels for each concept.

Explanation before "sequentiality" exercises: The experimenter explained the programming language by an example showing sequentiality (see Fig. 7). She wrote the scenario: "Nabi does the choreography 1 and then does the choreography 2".

Figure 7. Explanation before exercise 1

Exercises on sequentiality: This first explanation made the transition to the first exercise. They had to reuse the same concept and to choose other actions. It allowed checking whether the first explanation was understood. The first exercise (see Fig. 8) consisted in writing the scenario: "Nabi says hello and then says good bye".

Figure 8. Exercise 1

The second exercise completed the first one (see Fig. 9). The scenario was: "Nabi does the choreography 1, then says hello, then does the choreography 2, then says good bye". This exercise was the last of the sequential part.

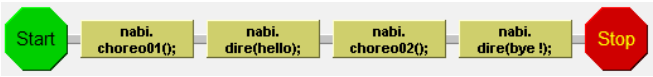

Figure 9. Exercise 2

Explanation before "parallelism" exercises: The experimenter introduced the parallelism. It increased a little the difficulty level because participants had to use a program structure element: **parallel**. It makes a second bar appear, which makes possible to drag and drop two actions in the same time. The experimenter wrote the scenario: "Nabi does the choreography 1 and in the same time Nabette does the choreography 1" (see Fig. 10)

Figure 10. Explanation before exercise 3

Exercises on parallelism: This explanation made the transition to the third exercise. They had to reuse the same concept. The exercise 3 (see Fig. 11) consisted in writing the scenario: "Nabette says hello and does the choreography 1 in the same time".

Figure 11. Exercise 3

The fourth exercise completed the previous one (see Fig. 12). It was asked to write the scenario: "Nabette says hello and does the choreography 1 in the same time. Then Nabi says hello and does the choreography 1 in the same time". This exercise used sequentiality concept and parallelism concept which increased the difficulty level.

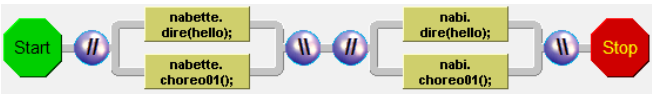

Figure 12. Exercise 4

The fifth exercise completed the previous one (see Fig. 13). It was asked to write the scenario: "Nabette says hello and does the choreography 1 in the same time. Then Nabi says hello and does the choreography 1 in the same time.

Then Nabi says pleased and does the choreography 2, and Nabette says pleased and does the choreography 2. These four actions are in the same time". The difficulty level was increased because participants had to use a program structure element: **braunch.** It adds a new bar to the **parallel** element.

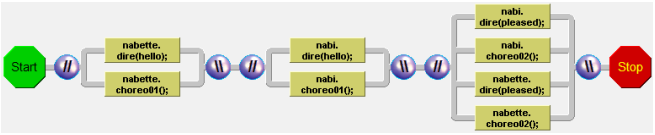

Figure 13. Exercise 5

Explanation before "event" exercises: The experimenter introduced the event concept. Participants had to use a program structure element: **event**. This element requires a perception and an action. The experimenter wrote the example scenario: "If the user has clicked on the blue button, Nabi does the blue choreography" (see Fig. 14).

Figure 14. Explanation before exercise 6

Exercise on event: This explanation made the transition to the last exercise. They had to use the event and parallelism concepts. They were asked to write the scenario: "If the user has clicked on the black button, Nabi says hello and does the choreography 3 in the same time" (see Fig. 15).

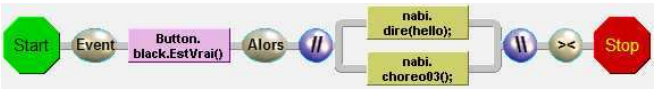

Figure 15. Exercise 6

C. Evaluation

There were two evaluations: observations made by the experimenter during the experiment and auto-evaluation questionnaire filled by the participants at the end.

1) Observations

During the experiment, the experimenter observed easiness and difficulties concerning the concepts and the exercises. She noted whether participants understood: drag and drop, compilation, instructions, sequentiality, parallelism and event. Each concept was noted understood (1) or misunderstood (0). If participants need some help about one of these concepts, it was noted as misunderstood. In this case, the experimenter helped the participant in order that he/she was able to continue the experiment. Help was not a bias because concept was noted misunderstood. The experimenter noted the success or failure of each exercise independently of concept understanding. The duration of the experiment was noted too.

2) Questionnaire

At the end of the experiment, participants had to fill a questionnaire, which corresponds to the Table I. It allows having the mental representation of children and the evaluation on AmbiProg. The A-F questions, except the question B, were evaluated with a Likert scale [14]. Participants had to choose among the following answers: Strongly disagree, Disagree, Neither agree or disagree, Agree and Strongly agree. In results study, we associate a value to these answers. 0=Strongly disagree, 1=Disagree, 2=Neither agree or disagree, 3=Agree and 4=Strongly agree. The question B did not impose specific answer. The G-H questions were yes-no questions.

TABLE I. QUESTIONNAIRE

A. I did not have any problem to do the exercises.
B. From which exercise did you think you were able to create scenario alone?
C. I think it was easy to create scenario with the graphical interface.
D. I can create others scenarios.
E. I like the graphical interface.
F. I think I can control others robots with this graphical interface.
G. Would you like to continue using the graphical interface?
H. Would you like to change something in the graphical interface?

IV. RESULTS

A. Encountered problems

1) Instructions

The half of participants received written instructions. We noticed that the children took a long time to achieve the tasks. Observations and discussion with teachers indicated that the problem came from the understanding of instructions. Children were concentrated to read and it was a problem to understand the instructions. It seemed to be a bias to our experiment. To ensure that the problem was indeed instructions, the second half of participants received oral instructions and we compared the instructions understanding in the both cases. It was important to check whether the difficulty came from instructions rather than AmbiProg.

2) Time

In parallel of this problem, we had a time problem. Teachers were not able to give children the same amount of time. Before this constraint, our idea was to be free of time and to observe how many times it took for children to do tasks. Time was not a consideration in our experimentation because our objective was to check if children were able to learn programming. The consequence of this time constraint was the impossibility to test the third concept. And, we had to accelerate children education.

B. Observations

Fig. 16 shows the difference between concepts understanding according to given instructions. The graph shows the average value obtained by each participant (1 for understood, 0 for misunderstood) The results revealed that participants had a better understanding when instructions were given verbally. Concerning the drag and drop, 100.00% of participants made it correctly against 87.5% with written instructions. There is no difference concerning the

compilation (75.00%). 37.50% of participants understood written instructions whereas 87.50% of participants understood oral instructions. 62.50% of participants understood sequentiality and 75.00% of them understood parallelism with written instructions while 100.00% of them understood both sequentiality and parallelism with oral instructions. Finally, 37.50% understood the event concepts in written condition against 0.00% in oral condition.

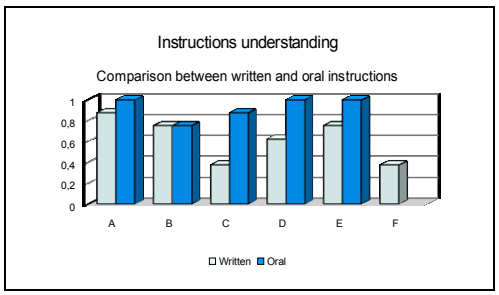

Figure 16. Comparison between written and oral instructions effects. A means drag and drop. B means Compilation. C means Instructions. D means Sequentiality. E means Parallelism. F means Event.

Almost all the participants completed exercises after at most two helps. The results are enough good because there was not a lot of time to do the explanations and exercises.

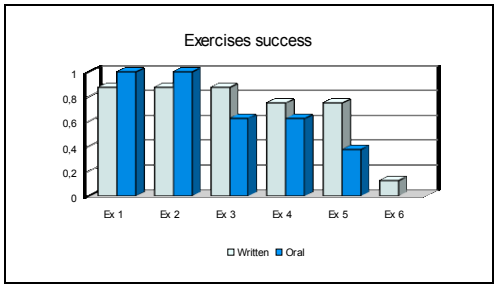

Figure 17. Comparison between exercices success according to given instructions

C. Questionnaire

The children mental representation seemed different to the observation (see Fig. 18). Despite the lack of complete control of AmbiProg, the majority of them considered having no problem to do the exercises (score of 4.44/5). The majority of them found the programming easy (score of 4.25/5). They feel capable of doing others programs alone (score of 4.56/5). Their judgment of the interface is tremendously positive. They liked the graphical interface (score of 4.88/5) and they thought it was possible to control other robots with it (score of 4/5).

Concerning their autonomy, they thought they were capable to create scenarios alone from the second or third exercise (mean of 2.31).

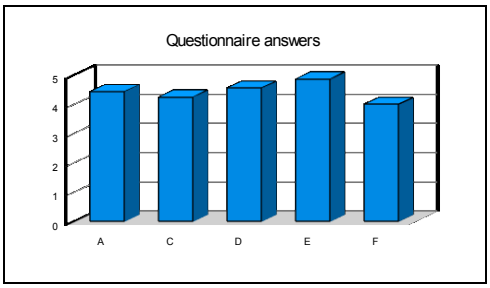

Figure 18. Answers to questions A, C, D, E and F

Concerning AmbiProg acceptability, results are positive (see Fig. 19). All the participants liked the interface and indicated that they wanted to continue using it. Only one participant wanted to change something to the interface, without any precision. All the others indicated that the interface did not need changes.

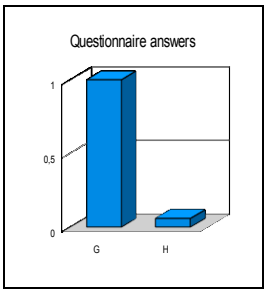

Figure 19. Answers to questions G and H

V. Discussion

A. Differences between written and oral condition

Fig. 16 and Fig. 17 compared concepts understanding and exercises success according to written or oral instructions. In spite of improvement of concept understand with oral instructions, the exercises success seemed to decrease. Actually, children better succeed the exercises in oral instructions but it does not appear in the statistics. It is easy to explain why. In the written instructions, the experimenter had more time: almost one hour. But after 8 children, the teacher explained that it was too long. The children missed a complete lesson and it was annoying. After that, the experimenter had only a half hour to carry out the experiment. She did not let children think and work a lot of time. She quickly changed exercises and had to note the exercises as uncompleted. So, exercises succeed seems better in the written instructions but it is explained by the fact that she had time to give more explanations about the instructions and the GUI. It added a bias to the experiment which should be avoided in a future experiment. Moreover, the only reason why the event concept did not be understood in the oral concept is the lack of time. The experimenter did not have the time to introduce it to children. That distorted results.

However, it allows understanding that a 10 years child needs 15 minutes to understand an AmbiProg concept.

This hypothesis should be confirmed in a future experiment. To avoid bias and time-consuming experiment, it could be a good idea to take more children during 15 minutes in order to show them only one concept. We could test each concept separately.

B. Differences between mental representation and reality

Children mental representation does not reflect the reality. They thought they have no problem to control the robot, no problem to do the exercises. It is extremely positive because they were voluntary and motivated although they had complex tasks to do. AmbiProg allowed them making program, executing program, controling robots without showing difficulties. In a learning process, it is not a problem if they make mistake when they create scenario. But the most important is their motivation. If they are motivated to use AmbiProg, they will more easily learn how to program and control their environment. AmbiProg is accepted by children. It is easy to manipulate AmbiProg. Children can control a set of robots!

VI. Conclusion and perspectives

This paper presented a computing architecture which connects a set of devices: ArCo. A communication protocol makes them communicating together ensuring an intercom-prehension. Moreover, ArCo contains a graphical user interface, AmbiProg, which allows manipulating a new visual programming language. Thanks to this language, everyone can create and activate some interaction scenarios. For example, a scenario can supervise a person health, while another one can supervise an agenda, while another one can supervise the activities… AmbiProg evolves according to the environment devices. In the domotic context, that means people do a single learning process with AmbiProg. Each time a new object is added to the environment, it is possible to control it through AmbiProg, with the same language than all others objects. It is even possible to take numerous objects into account and to do combinations between these objects, which is not currently possible.

This paper presented evaluation of AmbiProg. Two experiments showed that AmbiProg is easy to use for people who do not have computing skills. People are favorable to have this kind of application at home in order to control their environment.

ArCo was introduced at a stand during a robotic competition. Ten students with good computing skills tested AmbiProg with the robot shown in Fig 20. We observed that everyone was able to create and execute scenarios with less than 1 minute explanation. Two participants even created new robots motion and changed robot XML file which allowed integrating automatically new actions into AmbiProg. All the persons who tested AmbiProg were able to create and execute scenarios with less than 1 minute explanation. Two participants even created new robots movements and integrated new functionalities into AmbiProg.

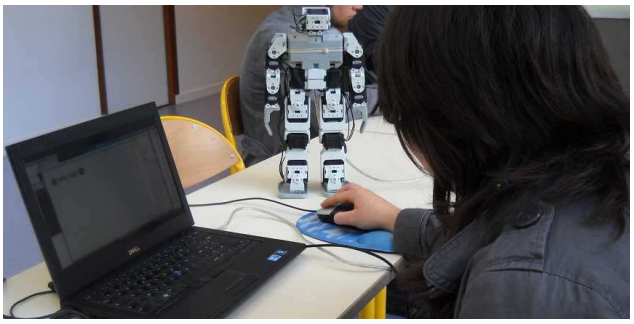

Figure 20. Robofesta stand with Bioloïd robot

We think that AmbiProg proposes a programming language closed from natural language. It allows everyone understanding AmbiProg concepts because scenarios are built almost like sentences.

The future work will be the experiment of ArCo at home. We will test our architecture with the elderly in a context of assisted ambient living.

Acknowledgment

We would like to thank teachers and pupils for their participation to our experiment.

References

[1] C. Jacquet, “Présentation opportuniste et multimodale d’informations dans le cadre de l’intelligence ambiante”. PhD thesis, Université d’Orsay Paris-Sud, 2006.

[2] M. Tedre, “What should be automated?”, Interactions, (2008), 47-79

[3] M. Grandgeorge and D. Duhaut, “Human-Robot: from Interaction to Relationship”, 14th International Conference on Climbing and Walking Robots And the Support Technologies for Mobile Machines 6th-8th september, Paris, France, 2011

[4] G. Pruvost and Y. Bellik, “Ambient Multimodal Human Computer Interaction”, in proceedings of the poster session at The European Future Technologies Conference FET09, Pragues, 2009

[5] A. Elgammal, “Human-Centered Multimedia: Representation and Challenges”, Proceedings of the 1st ACM international workshop on Human-centered multimedia, ACM,11-18, 2006

[6] E. Aarts and R. Wichert, “Ambient intelligence” in Technology Guide, Springer, 244-249, 2009

[7] C. Jost, B. Le Pévédic and D. Duhaut, “Creating Interaction Scenarios With a New Graphical User Interface”, in the 5th International Workshop on Intelligent Interfaces for Human-Computer Interaction, Italy, 2012

[8] S.Pesty and D.Duhaut, “Artificial Companion: building a impacting relation”, in IEEE-Robio 2011 December 7-11, 2011

[9] M. Young, D. Argiro and S. Kubica, “Cantata: visual programming environment for the Khoros system”, in ACM SIGGRAPH Computer Graphics, vol 29, no 2, ACM, 22-24, 1995.

[10] J. Maloney, M. Resnick, N. Rusk, B. Silverman and E. Eastmond, “The Scratch Programming Language and Environment” in ACM Transactions on Computing Education (TOCE), 2010.

[11] http://msdn.microsoft.com/en-us/library/bb483088.aspx

[12] P. Newton and J.C. Browne, “The CODE 2.0 graphical parallel

[13] http://www.nabaztag.com

[14] J.S. Uebersax, “Likert scales: dispelling the confusion”, in Statistical methods for rater agreement, vol 31, 2006.