

Combining cognitive, semiotic and discourse analysis to explore the power of notations in visual programming

Juliana J. Ferreira¹, Clarisse S. de Souza¹, Luciana C. de Castro Salgado^{1,2},
Cleyton Slaviero², Carla F. Leitão¹, Fabio Moreira²

¹Departamento de Informática, PUC-Rio
Rio de Janeiro – RJ, Brazil

²Instituto de Computação, UFF
Niterói – RJ, Brazil

{jferreira, clarisse, lsalgado, cfaria}@inf.puc-rio.br
cslaviero@gmail.com, fmoreira1@hotmail.com

Abstract — Using game design and programming to foster computational thinking acquisition has proved to be a successful strategy in recent years. In previous research with AgentSheets, we concluded that the semiotic richness of this visual programming environment, specifically designed to support computational thinking acquisition, could be explored more extensively to the benefit of learners. In particular, we realized that there are some additional representations of AgentSheets games and simulations that are not presented as programming tools in the interface, but yet they communicate new relevant meanings to the users. This paper reports on research where we artificially introduced such representations in a small follow-up experiment with selected participants from our previous research experiment. Our goal was to investigate the impact of such additional representations on program comprehension and modification tasks. To this end we contrasted empirical evidence of their performance in the two tasks with their verbal account of experience with AgentSheets. We used a combination of discourse analysis and inspections using Semiotic Engineering methods and the Cognitive Dimensions of Notations framework. Our findings go in two directions. First, we observed that additional representations have allowed participants to expand and correct previous learning. Therefore such representations can support new teaching strategies in computational thinking acquisition programs with AgentSheets. Second, we learned that the combination of methods we used to analyze empirical data – discourse analysis with semiotic and cognitive inspection techniques – can be used systematically in other research contexts, holding the promise of insightful results.

Keywords - computational thinking, end-user programming, visual programming, cognitive dimensions of notations, semiotic engineering methods.

I. INTRODUCTION

In order to be able to create various kinds of computer content such as contemporary websites, for example, end users need to *think computationally*. In other words, they must be able to express what they mean in *algorithmic form*, using the

vocabulary, grammar and semantic constraints inherent to all *computer languages*. These are available in various guises: from classical programming languages to user-friendly visual languages, with a number of hybrid-style specification and scripting languages between the two extremes.

Computer programming has quickly become an important means for self-expression and participation in 21st century society. But to achieve this end, computer users must not only learn how to read and write in computer languages, but also to think and reason using computer languages as representational support. Hence the effort placed in stimulating computational thinking acquisition. [7]



Figure 1. Editing an agent's depiction

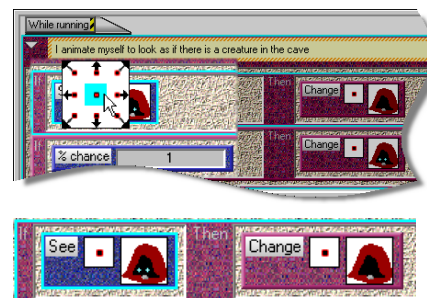


Figure 2. Editing an agent's "if-then" behavior rule

AgentSheets is a visual agent-based programming environment specifically designed to support learning of computational thinking while building games and simulations. [11] It includes a number of different notations that can be manipulated by users as they build their own games or simulations. Visual notations are the most salient ones in AgentSheets. With them, users can do most of their programming in this environment. For example, to create an agent (the basis of all *programs*) users interact with graphical interface controls (see Figure 1). Likewise, to specify rules of agent behavior, users perform interface actions like clicking,

dragging and dropping visual elements, pressing keys and typing in data (see Figure 2). Notice that there are at least two *notations* in these examples: the visual interface layout and controls language and the visual programming language (see the visual “if-then” rule representation in Figure 2).

AgentSheets users have different representations of their *programs* (typically games and simulations, as already mentioned). Some of them are *static*, like agent depictions and behavior rules shown above, whereas others are *dynamic*, like AgentSheets’ *conversational programming* feature [10], which paints true and false conditions (the “if” part of behavior rules) with different colors during *program* execution. This kind of animation helps the users understand why agents do what they do when the program is running.

In previous research using Semiotic Engineering [3] to investigate how learners express meanings computationally in AgentSheets [4], we concluded that the semiotic richness of this visual programming tool could be more extensively explored to introduce different aspects of computational thinking. In particular, we saw that AgentSheets had more representational resources than its interface design suggested. One such case was the *program report*, a static representation of the whole program (in visual notation), which was communicated in the interface as some secondary function of relatively minor interest to learners. In Figure 3 we see the visual and semantic salience of other programming tools, compared to the *program report*. The three *palettes* (visual tools for visual programming environments) are clearly presented as more relevant than the other menu entries; note in particular that there are two separators in the menu, defining three menu *sections*. The “report” appears in the third one.

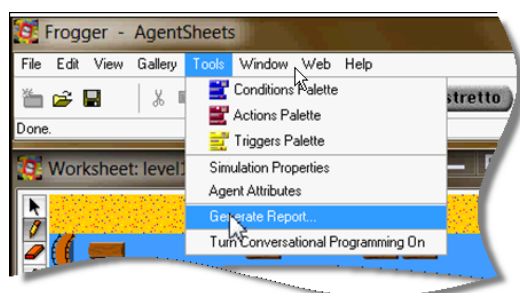


Figure 3: Menu entry for generating the program report

The *program report* is an html file that displays all agents and their corresponding behavior. Together, this is the full specification of a game or simulation – the full *program*. The complete static view of the *program* is not provided by any of the other notations in AgentSheets. For example, in order to compare the behavior of two related agents, the user has to keep two “behavior” windows open, one for each agent. In typical displays, it may be impossible to view more than half a dozen behavior rules, especially if they involve a conjunctions of conditions and/or actions. As a result, representations of *the whole* are hard to get. The only exception is, of course, the game or simulation, while it runs. This is a complete dynamic representation of the whole program.

This paper presents the results of a qualitative study with which we investigated the impact of using the AgentSheets

program report as an additional *notation* in program comprehension and program representation activities. We used a combination of three qualitative methods. Discourse Analysis [5] helped us understand the participants’ feelings and meanings in relation with AgentSheets and the activity they had to perform. Semiotic Engineering concepts and inspection methods [2] helped us analyze the meanings communicated by AgentSheets designers through the system’s interface. And finally the Cognitive Dimensions of Notations framework [1] helped us identify cognitive loads associated with the various notations used in AgentSheets, and the relations between them.

Our conclusions bring practical and methodological contributions to the broader ongoing research path in visual programming and computational thinking acquisition. In practical terms, we show why and how hybrid notations have helped participants to expand and correct previous learning. We also show that they can thus support new teaching strategies in computational thinking acquisition programs with AgentSheets. In methodological terms, we have strong indication that our combination of methods can be used systematically in other research contexts, holding the promise of insightful results.

The paper is structured in four sections. After the introduction, we present the research description, including a brief account of prior experiments and its results, as well as a description of how three methods were combined in the analysis of data collected for this study. In the third section we present our research findings and finally the last section we present our conclusions and future work.

II. RESEARCH DESCRIPTION

In previous research [4], carried out in 2010, we verified that the students’ semiotic strategies when using the AgentSheets visual language to express their intent could be further explored by instructors in order to advance important computational concepts. The next question in our research path was, naturally: “How this should be done?” In order to find an answer, we did new empirical research, combining data and findings from 2010 with new data collected in 2011. Just as before, the new study followed a qualitative methodology, where *meaning* is at the center of investigation.

A. Previous Experiment and its Results:

Our first experiment was run during an after-class computational thinking acquisition (CTA) program in Brazil for which twenty 9th-grade students, thirteen females and seven males between 14 and 16 years of age, voluntarily enrolled. [4] The program followed the Scalable Game Design (SGD) project orientation [12] and extended over seven two-hour weekly sessions. We worked with data collected in (a) face-to-face interviews with students about the games they were building and (b) successive versions of their games. We used discourse analysis [5] to interpret data collected in interviews and semiotic inspection elements [2] along with semiotic theories [8][9] to interpret static and dynamic program representations. We compared meanings and expressive choices emerging from natural language narratives describing the game with those from static agent depiction and behavior

specifications and dynamic game play experience. Our conclusion was that AgentSheets presented a wealth of related representations that could be yet further organized and explored to enhance the teaching-learning process in CTA projects.

B. Further Explorations with SIM:

In 2011 we continued our research first taking a closer look at the meanings communicated by AgentSheets designers through the system's interface. To this end we used the Semiotic Inspection Method (SIM) [2], which allows us to characterize how interface designers organize and structure various *signs* (like words, images, layout, widgets, animations, screen patterns and sequences, etc.) to *tell* the users an interactive message that we can paraphrase as this:

“Here is my understanding of who you are, what I've learned you want or need to do, in which preferred ways, and why. This is the system that I have therefore designed for you, and this is the way you can or should use it in order to fulfill a range of purposes that fall within this vision.”

In this message the first person “I” refers to the designer (or the person who represents the design team), whereas the “you” refers to the user (or targeted user community). In accordance with Semiotic Engineering [3], this method frames human-computer interaction as a special case of computer-mediated human (designer-user) communication and analyzes how this communication is *emitted*, that is, *sent* from designers to users.

Among other results not directly related to the theme of this paper, we found that although the designer's message is almost entirely communicated with signs that appear in the gallery of agents, game worksheet and agent behavior rules windows (see Figure 4), there is a complete static representation of the *program* (encoded in the same visual language used in the rest of AgentSheets) very weakly communicated, as showed in the last sub-section of the “Tools” menu in the menu bar – the *program report* (see Figure 3 above).



Figure 4. AgentSheets main visual elements

Just like the *program worksheet* is the complete dynamic representation of the *program*, the *program report* is the complete static representation of all specified elements that constitute the *program* and whose activation causes the game experience. One of the distinctive features of the *program report*, compared to all other static representations of the *program* (like behavior rules or agent depictions, for instance) is that in it we can see the *whole* set of specifications for all the *program* elements, and not only the *parts* associated to a particular selected agent. To be precise, a critically important

program element – the *program worksheet* (with background image and localization of all agent instances at the initial game state) – is not included in the *program report*. However, as further analysis has shown, nothing prevents that the *program worksheet* be included in the *program report*, turning it into the exact static representational counter-part of the animated *program* play.

C. The new qualitative study with CTA learners:

We carried out a small qualitative study to explore the impact using the *program report* in program comprehension and program modification tasks with AgentSheets. We invited four students who had followed the entire 2010 CTA project and whose games ranked among the best in the class. We met them one year after they had finished the project. During a 90-minute session we interviewed them individually, to find out what they recalled from the project, what feelings and attitudes they had regarding CTA, AgentSheets and the entire learning experience, and if somehow they had continued their learning after the project was over (or wished they would be able to do so). In the interview, they were encouraged to speak freely of their past experience in the project, so that we could collect a rich set of meanings that each participant assigned to learning computational thinking with AgentSheets,

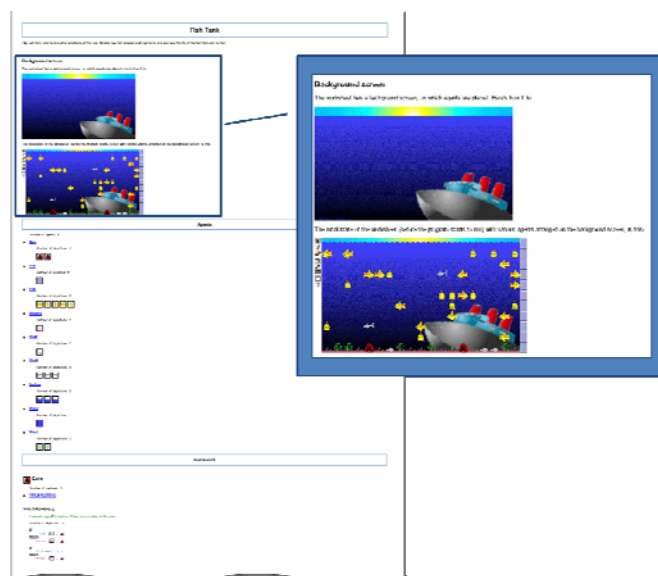


Figure 5. Extended report = Report + Background screen content

After the interview, participants went to the computer to have a new encounter with AgentSheets, after one year since their last CTA class. Their task was to play with and examine a new *program*, called Fish Tank. However, unlike what happened in their previous experience with AgentSheets, this time they had additional printed paper sheets with an extended version of the *program report*. In it, besides the full specification of all agents' names, depictions and behavior rules, we included a visual description of the initial *program worksheet*, with its background image and the location of all agent instances (see the magnified insert in Figure 5). In this context, they were asked to perform three tasks: (1) to explain

how the Fish Tank program worked; (2) to propose one modification, whatsoever, in the way it worked, showing briefly how this would be expressed in the *program report*; and (3) to effect this modification using AgentSheets. During the tasks, participants were encouraged to talk with the researcher about what they were doing.

The empirical data collected from this experiment was rich. We gathered the audio recording of the interview, the audio recording of the verbal protocols during the tasks, the participants' annotations on the printed program report sheets, the modified version of the Fish Tank program, and the researchers annotations made throughout the experiment. We also used, as additional empirical evidence, data produced by the participants in the previous 2010 CTA project experiment.

D. Combined Methodology for Data Analysis

Before we begin to describe the methodology we used for data analysis, we should explain why we collected such heterogeneous (although tightly related) kinds of data. The reason for this is rooted in the Semiotic Engineering perspective that governs this research. In it, whatever happens in human-computer interaction is the result of a computer-mediated intentionally designed communication of what a system does, why, how, where, when, as well as to whom it has been designed. Therefore, in the specific context of this research, through the system's interface, AgentSheets designers are *telling* computational thinking learners their whole design vision. Their message is heavily communicated with visual codes, which support a wide range of manipulations through which the user *talks back* to the system and gets it to achieve his or her interaction intent.

This designer-user communication through the interface can be viewed from two angles. One is the *emission* of the communication, that is, how the designer encodes what he has to say to the users. The other is the *reception* of this communication, that is, how the users perceive, interpret and react to the designer's communication.

The whole set of data collected in this experiment allowed us to investigate aspects of both the *emission* and the *reception* of the designer-user computer-mediated communication. For instance, in interview and verbal protocol data we could search evidence of how the designers' message was received by users. Likewise, in registered program states and manipulations data we could search evidence of how the mediated designer-user conversation was articulated in a real context of use. Finally, in static and dynamic, integral and partial, visual program representations used during the experiment we could search evidence of cognitive and semiotic operations that users must achieve in order to *interact* successfully with AgentSheets.

This hybrid set of data was analyzed using a combination of three methods: discourse analysis (DA) for interview and verbal protocol data; Semiotic Engineering inspection methods and concepts (SIM) for AgentSheets designer-user communication signs; and the Cognitive Dimensions of Notations (CDN) framework for visual programming notations.

CDN are design principles for creating or evaluating notations, user interfaces and programming languages used

with information artifacts [6]. They provide a common vocabulary for discussing many cognitive factors of such representation systems. Their aim is to improve the quality of discussions and decisions in design and evaluation activity [1]. There are fourteen dimensions in the CDN framework [6]. Just for sake of a very brief illustration, one of them is *hidden dependencies*. It refers to a situation where one notational entity depends on another but the dependency is not fully visible in the notation itself. Another dimension is *premature commitment*, which refers to notations that require that decisions be made prior to having all needed information and knowing all the related task ordering constraints.

CDN were chosen to analyze part of our data because the purpose of the entire AgentSheets technology is to support *learning*. Therefore, we must explore cognitive aspects of the various languages (mainly visual) used in AgentSheets if we want to know if and how this technology achieves its goal.

SIM was applied for two main purposes. First, SIM helps us to identify the various sign systems and notations with which AgentSheets designers structure and communicate their entire design vision to users. And second, SIM helps us to select which notations of all the ones used in AgentSheets should be considered for cognitive analysis, given the role they play in the computer-mediated designer-user communication that supports the users' learning process.

Finally, DA was used because it provides additional evidence to support findings obtained with CDN and SIM. Specifically, by dealing with natural language discourse about CTA, AgentSheets, and all the elements involved in the activity with the Fish Tank game, DA produces evidence that can describe or explain how users perceive and interpret signs and notations. Hence, by combining DA with CDN and SIM we obtain a kind of *closure* to the interpretive analysis cycle in our investigation.

The combination of methods is governed by the communication of design intent through interface signs. Therefore, SIM is applied first. Once we have a characterization of the designers' message to the user, we can proceed to following steps with DA and CDN. DA tells us how the designers' message is received and used by participants to accomplish specific goals, whereas CDN highlights specific cognitive characteristics of signs (emitted by designers and received by users). Together, the three help us gain a deeper understanding of learning processes supported by various interface sub-languages (*i. e.* notations or representation systems), with their structural and functional inter-relations.

For a very brief illustration of the effects of such combination, in the Fish Tank program scene (the *program worksheet*), some twenty fish or so are swimming in the same tank as two sharks. When the program starts, fish and sharks move around the tank. If the player-learner sustains the program long enough he will realize that the number of fish is slowly decreasing because, every once in a while a shark *eats* a fish. However, if the player-learner looks at the *program report*, he will quickly realize that sharks eat fish (there is a rule saying so). Not only this, he will also see that the frequency with which shark eats fish is 10% (see Figure 6). The two 'notations' (the dynamic program play language and

the static visual report language) yield different analyses relative to the visibility dimension, for instance. In the program play language the logic of the shark behavior is not as visible as in the visual report language.

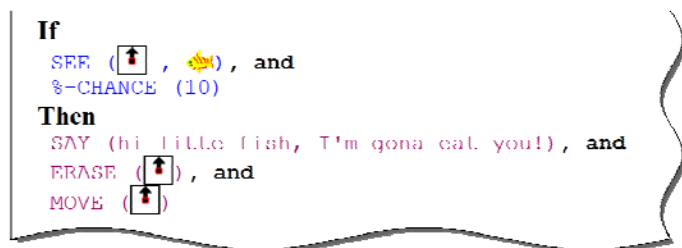


Figure 6. Part of the shark agent rules

However, for effective learning, what must be really visible is the *relationship* between one and the other, which AgentSheets expresses in the Conversational Programming notation, for instance. However, if we look at SIM results, we see that each notation is shown in separate independently-controlled windows that users may fail to configure appropriately. In this case, they will miss an important part of the designers' message about how and why AgentSheets representations are related to one another.

Discourse analysis provides us the main evidence of how the learner *actually* perceives, interprets and integrates information coming from communication expressed in different notations. Therefore we started to analyze the participants' discourse by looking for elements that could be associated with any of the fourteen dimensions defined in CDN. Upon finding such elements we then examined the following factors:

- Presence or absence of corresponding cognitive characteristic. For example, upon finding evidence that the participant was talking about 'visibility' in AgentSheets notations, we checked whether he or she was referring to the presence or absence (lack) of visibility in the notation.
- The impact of presence or absence of cognitive characteristics. For example, once we identified that the participant was talking about the *presence* of 'visibility' in a certain notation, we looked for evidence of value judgment: did this have a positive (+) or negative (-) impact on the participant's learning process.

In order to verify, instantiate and understand evidence from discourse analysis, we looked into the other collected data (*e.g.* annotations on the program report, state of the AgentSheets program, and even evidence from programs produced during the 2010 CTA project).

At a later stage, evidence from discourse analysis was grouped into two broad meaning categories:

- "Recollection of difficulty" - The participant narrates some situation when he or she faced difficulty in the learning process.

- "Aha moment" - The participant suddenly gains new understanding about the AgentSheets notation.

In fact we identified a third category of meanings which we called "AgentSheets' idiom". This refers to narratives when participants incorporated AgentSheets notation vocabulary into natural language discourse. However, because this category did not explicitly yield interesting results regarding our research question, we will not discuss it further in this paper.

III. FINDINGS

It is worth noting, at the beginning of our findings section, that the relation between the two factors used to analyze evidence in DA, presence or absence of cognitive characteristics and their impact on the learning process, led to interesting findings. This relation was very consistent within the scope of each CDN dimension. In other words, the presence or absence of the cognitive characteristic associated to the dimension had always the same impact (+ or -) on the learning process. For example, when the cognitive characteristics of *visibility* were absent, the impact was consistently negative. When present, the impact was consistently positive. This suggests that CDN is itself a consistent tool to evaluate the cognitive impact of sign selection and organization in building the various representation systems through which designers communicate with users.

Another useful observation is that most of the evidence leading to conclusions about the impact of using the AgentSheets program report as a first-class visual programming tool came from verbal protocols collected during the program comprehension and program modification tasks.

Participants reported recollections of difficulties encountered when they were building their first game program during the CTA project in 2010. They talked about the learning process that they went through to find a solution or a way around a problem. Sometimes the difficulty was overcome, but sometimes it developed into frustration. In the fragment below we find evidence of the presence of a cognitive dimension called *progressive evaluation* with a positive impact on the student learning process. The student reported how the notation let him adjust time values and see the results of the adjustment. "This was where I was testing with multiple agents ... the movement frequency ... I tested it first putting 1 second for him to move randomly, it was quite slow. Then I lowered it to 0.5 seconds, but it was still slow ... So, I kept on lowering it [till it was OK]... At the time of the [2010] project it was something that I had a hard time mastering."

The recollection was triggered by a rule defined for the shark's behavior (see Figure 7). In other words, by looking at the visual notation, the participant was able to add meanings related to this notation's *progressive evaluation* support.



Figure 7. Shark's rule of movement frequency

In some of the narratives about past difficulties, we verified that the introduction of the *extended program report* caused a

kind of “aha moment”. That is, the participant experienced a breakthrough in understanding a programming situation, a rule or action. Sometimes even wrong understandings got corrected during such “aha moments”. In fact, this kind of evidence was very typical when the extended program report was being used. The combination of the dynamic and static representations of the complete program promoted new levels of comprehension.

For example, one of these insight moments occurred with one of the participants who first thought that the Fish tank program was apparently meaningless. She said: “*The shark has a will of its own and the little fish too...nobody wants to obey me*”.

This piece of evidence can be associated with the *visibility* dimension. Its cognitive characteristic was absent from the notation (the participant could not *see* what the program was doing) and it had a negative impact on the learning process (she therefore did not *understand* the program). This misconception was encouraged by the underlying notion developed in the 2010 CTA project that AgentSheets programs were *games*, that is, something that users could play using interface controls (like arrow keys, for example). Moreover, this project was presented as the *Scalable Game Design*. So she tried to interact with the Fish Tank program by pressing the arrow keys, but nothing happened, because this was actually a *simulation* program, and not a *game*.

This situation was associated with the cognitive characteristics of the *error-proneness* dimension, which when present had a negative impact on the learning process. Although AgentSheets programs can either be (interactive) *games* or (non-interactive) *simulations* governed by encoded parameter values, its interface notations associated to the *game worksheet* during program execution (the complete *dynamic* representation of the program) do not have any indication of which is the case.

But when she looked at the *program report*, this is what she said: “*Hmm... so this is what happens!*”. She very quickly realized that there were no input condition rules governing the behavior of agents. This evidence is again related to the cognitive characteristic of the *visibility* dimension in CDN. The dynamic visual representation of the program in execution (*i. e.* the simulation) did not give visibility to the underlying logical relations established among agents. However, once the program report was made available to the participant, the underlying logic became immediately visible. More than that, we could also use the cognitive characteristics of the *hidden dependency* dimension to see that an important *sign* in the AgentSheets interface (and ultimately in the acquisition of computational thinking) – the connection between rule specifications and acting program execution – was hard to perceive. This is an interesting demonstration of how the combination of three methods can boost the analysis of observed phenomena, yielding results that exceed those that we typically obtain if they are used in isolation.

Further evidence showed that another participant experienced his “aha moment” when looking at the *program report* he saw that the *worksheet* included a background image. Together with the disposition of agent instances in different locations of the program grid in the worksheet, the background

image formed a new *sign* (a significant unit) for the participant, who said that he did not know that he could have used an image as the program background. The student reported difficulties to compose the background by creating and deploying background agents: “*Oh! So, I have this [other] possibility to compose the background of the game... I built all my background with agents, which sometimes caused a lot of trouble.*” We associated the cognitive characteristics of the *error-proneness* dimension of CDN with this piece of evidence. This also contributed to verify that whenever the presence of this dimension was associated with evidence, the participant who provided the evidence was reporting some negative impact of the notation on his or her learning process.

We had, however, with this particular instance, evidence of the positive impact of “Visibility” in the *program report* notation, especially in comparison with the active game worksheet notation. The background image is not always easy to perceive as a separate component from the set of agents. For example, the programmer may have chosen to deploy agents in all the *program worksheet* cells, making it impossible to visualize the background image when the *program* starts to run. When used in combination with the *program report* notation, the background image gains visibility, allowing the learner to think about its role in the logic or appearance of the *program*, that is, in communicating the essence of the *program* to its users (the players).

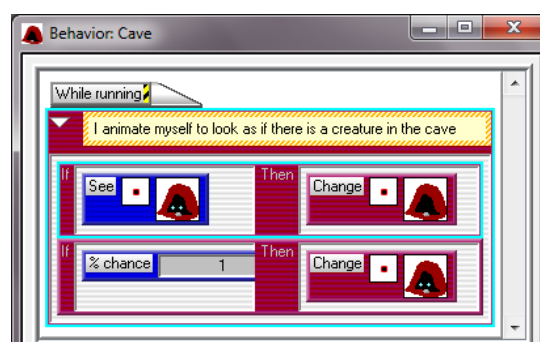







Figure 8. Cave behavior - the dot parameter

Another powerful evidence of the benefits brought about by the new “Visibility” of program components and their logic, as expressed in the *program report*, was given by all participants. When they look at the *program* running, they did not perceive some rules, different agent depictions or other element about the *program*. But looking at the *program report*, they could learn more about the *program* composition, its elements and so. For example, one of the participants did not perceive an important change in agent depiction while looking at the *running program*. The “cave” agent appearance alternated from  (empty cave) to  (cave with caveman) and vice-versa (see Figure 8). Perception was (perhaps intentionally) low because the rules governing it were that 1% of the time the agent changes from “empty cave” to “cave with caveman”, but the agent always (*i. e.* 100% of the time) changes “cave with caveman” to “empty cave” when it sees it. However, it was only when the participant looked at the *program report*, that he realized that the cave actually *had* different depictions. He had not noticed it when running the game. He said: “*the cave*

... seems to have a little pair of eyes in it ". This caused him to *reinterpret* the dynamic representations in the *program* play, now expecting to see the caveman quickly appear inside the cave on very rare occasions (1% of the time). This is a new meaning for the cave space.

The complete static representation of the *program* structure and codification in the *program report* also led to confrontations with one's own gaps in the learning process. A very strong piece of evidence was found when another participant fell upon the visual notation for self-reference () in the "cave" behavior rules just mentioned above. She asked herself repeatedly out loud: "What is this dot?" Part of the interest associated with this evidence is that this specific participant had made extensive use of this notation when she produced her first game in 2010, during the CTA project. She also explicitly acknowledged, when the researcher asked her, that she had noticed the "dot" in the rules she defined for the agents in her 2010 game, but had not known what the dot meant. This did not bother her then, but now that she depended on the "dot" meaning to make sense of the Fish Tank *program*, she was in trouble. She saw it in the *program report* that this kind of notation was used many times – so it became even *visually significant* for her now. She realized that the dot *mattered* in AgentSheets rules.

Representing self-reference visually with a *dot* violated the *closeness of mapping* dimension of CDN. When absent, this cognitive dimension of notations had a negative impact on the learning process. We should remark that the dynamic representation of the complete game at run time did not signify self-reference with the same meaning elements as signified in the *program report*. In fact, as discussed in our previous work [4] the *functional* use of self-reference did not always correspond to the *semantics* of self-reference in reality. For example, games produced in 2010 extensively denoted that the *visual* interpretation of the game included meanings that were *absent* from the underlying rules governing program execution. Once such case was when one student wanted a rabbit to eat the carrot (this is what she said in her natural language description of the game). The way she programmed the rules was such that this effect was achieved by a rule governing *the carrot's* behavior (and not the rabbit's). When *the carrot* saw the rabbit next to it, *the carrot deleted itself*.

Evidence of difficulty in interpreting "the dot" in the Fish Tank *program report* may perhaps be associated with difficulties in dealing with inverted semantic transitivity of the sort just illustrated above. We can conjecture that because the "dot" could not be *closely mapped* to a realistic semantic interpretation of the game plot, learners did not achieve satisfactory (and correct) understanding of the meaning of this notation. This is an excellent illustration of how the designer's message can be improved if, prior to deciding which visual sign will be used to represent self-reference, the designer is encouraged to take a more global perspective on what he wants to *tell* to the users. Knowing that self-reference is one of the most critical (and productive) concepts in computational thinking, AgentSheets designers may now think of using additional signs to underline the communication of self-reference (like textual support, for instance), so that learners

have a greater chance to grasp critical concepts in computing. Once again, we see the merits of combining the three methods of analysis, which help us not only in diagnosing cognitive issues in visual notations, but evaluating what these issues *mean* in the broadest concept of designer-user communication (emission and reception) with a specific purpose in view.

The most commonly encountered dimensions of CDN encountered in our analysis were *visibility*, *consistency* and *closeness of mapping*. They shared a common element in the contexts where they occurred: they involved *relations* between representations. In this sense, *visibility* was often the more obvious counter-part of *hidden dependencies*. Likewise, *consistency* could only be established in relation with some perceived *patterns* of notation occurring in multiple instances. Finally, the *closeness of mapping* dimension is *per se* a relational concept. We interpreted this result as a sign of the critical importance of giving AgentSheets users, as computational thinking learners, multiple *signs* of how parts relate with each other to compose higher-order significant wholes. However, these signs are hard to perceive in the most salient patterns of communication used in AgentSheets, namely: the gallery of agents (and the interactions the user can have with agents); the condition and action *palettes* with which to build behavior rules; the behavior rules window for each agent (and the interactions the user can have with rules); and the *program worksheet* (with the interactions made available during program design time and program play time).

The reaction of participants in view of signs in the extended version of the *program report* used in this study strongly suggests that this artifact communicated *many* of the relations that participants failed to perceive or understand when using other notations.

IV. CONCLUDING REMARKS

Findings of the qualitative study reported above have practical and methodological contributions. Practically, the study showed that, as anticipated in previous research steps of the Scalable Game Design project we are running in Brazil [4], AgentSheets offers us a wealth of *signs* that can be explored more systematically in CTA projects. The use of *program report* signs, which constitute a visual static notation to communicate the complete program structure and content, expanded the participants' *reception* of the global design message sent through AgentSheets interface signs. This added to the AgentSheets designer's communication effectiveness, which is a value in itself. However, because CTA projects involve learners *and* teachers, we asked one of the teachers who participates in our project how he thought the *program report* could be used *in class*. Here are two excerpts of his testimony:

"The *program report* could make the teaching easier. I think about time, because we had a schedule with an end date for the project. And all activities should be carried out in such a way that we had enough time to communicate content and reach our goals. So, I think of situations that required a lot of time [in my teaching]. Maybe, with the *program report*, I would have solved problems much faster, especially when I had to handle games that already included many agents. At this

stage, [using the report to] point at programming errors and connections between agents behavior would have been much easier.”

“Orienting the learners without this report was confusing, because the student needs to click on one agent, visualize its rules, many times doing the same thing many times. In general they cannot remember the whole set of rules for every agent. So, using the report gives us a broad view of all agents and how each one is programmed.”

This testimony confirms the results of our analysis of empirical data collected with participants of our studies. It denotes that promoting the program report to a first-class notation among other AgentSheets interface notations can support new pedagogical strategies in CTA projects. As a consequence, in our future work agenda we have included the development of a new AgentSheets interface that will communicate the presence and the value of complete static representations of games and simulations more clearly.

Methodologically we believe that this study provides a good framework for combining interpretive methods of analysis that focus on different but related aspects of visual programming notations. We used three methods. Semiotic Engineering inspection [2] was used to characterize how AgentSheets designers communicate their design vision to users (which include learners and teachers in CTA projects). An important product of this characterization was the identification of which notations were used by the designer and what communicative purposes they should fulfill in the global process of computer-mediated designer-user communication. Discourse analysis was used to characterize how the designers' communication was received by different users in situated contexts of activity. Finally, the Cognitive Dimensions of Notation framework [1] was used to reveal the cognitive impact of notational design on learning processes. Interestingly, when associating CDN dimensions to discourse material reporting how learners received the AgentSheets interface message, we found that those referring to relations between notational elements or between notations and their meaning were the most frequent. This is completely consistent with our claim that the program report contains relational signs that are critically important to CTA and can be hard to perceive in other notations. Moreover, the teacher's testimony mentioned above reinforces the points uncovered by the cognitive analysis of static and dynamic notations used in AgentSheets.

To the best of our knowledge, this was the first time this combination of methods was used. Therefore, a second item in our future work agenda is to use this methodological strategy in other studies with visual languages in order to reach a deeper understanding of its merits and caveats.

ACKNOWLEDGMENT

The authors want to thank the Brazilian funding agencies that support this project in different ways: CAPES, CNPq and FAPERJ. They would also like to express their gratitude to the students and teachers who participated in various experiments of the Scalable Game Design Brasil. Last but not least they

thank Alex Repenning and all the AgentSheets team for their constant support, incentive and kindness.

REFERENCES

- [1] Blackwell, A., Green, T. “Notational systems: The cognitive dimensions of notations framework” In J. M. Carroll (Ed.), *HCI models, theories and frameworks: Toward a multidisciplinary science* (pp. 103– 134). San Francisco. 2003.
- [2] De Souza, C. S. and Leitão, C. F. “Semiotic engineering methods for scientific research in HCI”. Princeton: NJ. Morgan & Claypool. 2009.
- [3] De Souza, C.S. “The Semiotic Engineering of Human–Computer Interaction”. Cambridge, MA. The MIT Press. 2004.
- [4] De Souza, C.S.; Garcia, A.C.B.; Slaviero, C.; Pinto, H.; Repenning, A., “Semiotic Traces of Computational Thinking Acquisition” In Costabile, M.F., Dittrich, Y., Fischer, G. and Piccinno, A. (Eds.) *End-User Development. Lecture Notes in Computer Science 6654*, pp. 155-170. Springer Berlin / Heidelberg. 2011.
- [5] Gee, J. P. “An Introduction to Discourse Analysis: Theory and Method”. London: Routledge. 2005.
- [6] Green, T., Blackwell, A. “Cognitive Dimensions of Information Artefacts : a tutorial.” *Applied Psychology* October. 1998.
- [7] National Research Council Committee for the Workshops on Computational Thinking. “Report of a Workshop on The Scope and Nature of Computational Thinking”. Online at <http://www.nap.edu/catalog/12840.html> . 2010.
- [8] Ogden, C. K. and Richards, I. A. “The meaning of meaning.” (8th edition) New York, NY. Harcourt, Brace & World, Inc. 1989.
- [9] Peirce, C. S. “The Essential Peirce, Selected Philosophical Writings”, Volumes 1,2. Edited by Nathan Houser and Christian J. W. Kloesel. Bloomington, IN. Indiana University Press 1992-1998.
- [10] Repenning, A. “Making Programming more Conversational.” *Visual Languages and Human-Centric Computing (VL/HCC)*, 191-194. Ieee. doi:10.1109/VLHCC.2011.6070398, 2011.
- [11] Repenning, A. and Ioannidou, “A. Agent-Based End-User Development”. *Communications of the ACM*. Vol. 47(9) 43-46. 2004.
- [12] Repenning, A., Webb, D., and Ioannidou, “A. Scalable game design and the development of a checklist for getting computational thinking into public schools” In *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10)*. ACM, New York, 265-269. 2010.