Investigating Learners' Behaviours When Interacting with a Programming Microworld

An Empirical Study Based on Playing Analytics

Fahima Djelil[1(✉)], Pierre-Alain Muller[1], and Eric Sanchez[2]

[1] University of Haute-Alsace, IRIMAS Laboratory, Mulhouse, France
{fahima.djelil,pierre-alain.muller}@uha.fr
[2] CERF, University of Fribourg, Fribourg, Switzerland
eric.sanchez@unifr.ch

Abstract. In our attempt to support Object-Oriented Programming (OOP) learning for beginners, we designed a novel microworld called PrOgO. It is based on a three-dimensional (3D) constructive game metaphor for describing OOP basic concepts and their implementation. In this paper, we describe a study about the use of PrOgO by beginners to investigate their behaviours when interacting with the programming microworld. The study is based on the collection, analysis and reporting of data about players (playing analytics). The data analysis allows the identification and the characterisation of different behaviours. From an educational perspective, the expected behaviour has been confirmed for a limited number of students. This enabled us to conclude that the design of the game needs to be improved. In addition, behaviours triggered by most students might have other educational values, which could be confirmed by other similar studies.

Keywords: Programming microworlds · Game-based learning · Learning analytics · Object-Oriented Programming · PrOgO

1 Introduction

Teaching and learning introductory Object-Oriented Programming (OOP) is recognised to be a difficult task [1]. Generally, abstract concepts describing OOP basics, and their implementation (coding) are the main difficulty faced by beginners [1]. This includes the concept of an "object", its properties and its relationship to the concept of a "class", as well as the relationship between classes. To overcome this educational difficulty, microworlds have often been used [2]. Representative OOP microworlds are Object-Karel, Alice and Greenfoot. A review of literature of these environments already exists [2]. These digital environments aim to give to beginners an intuitive and rapid understanding of abstract programming concepts, by allowing them to interact with representations of these concepts [2]. Following this trend and in our attempt to support OOP introduction for beginners, we created a new microworld called PrOgO. It is based on a three-dimensional (3D) constructive game metaphor and C++ programming language [3].

D. Passey et al. (Eds.): OCCE 2018, IFIP AICT 524, pp. 67–76, 2019.
https://doi.org/10.1007/978-3-030-23513-0_7

This paper reports on an empirical study on the use of PrOgO by beginners, to investigate its educational value. In particular, our goal is to know if the students' behaviours are relevant, by providing them with a good understanding of the addressed concepts. Playing analytics enables for the description of learners' behaviours. We collected and analysed interaction traces on the basis of event logs generated by PrOgO when students interacted with it. This work does not consider the link between learners' behaviours and their knowledge gain. This was carried out in another experiment which is already published [4].

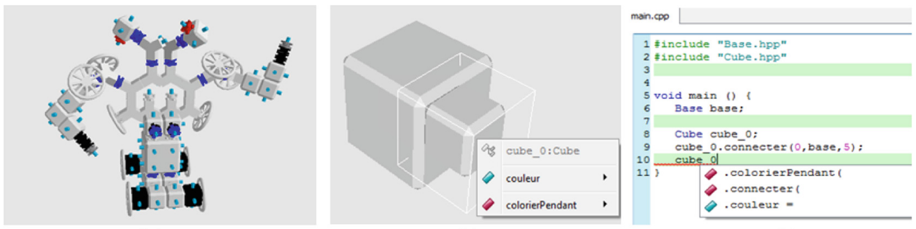
The present paper is organised as follows. Section 2 gives a brief description of the PrOgO microworld. Section 3 is dedicated to the description of the study, including the addressed research question and the related hypothesis. Section 4 describes the methodology employed, including details of the context, as well as data collection and analysis. Results are summarised in Sect. 5 and then discussed in Sect. 6. Finally, conclusions are drawn in Sect. 7.

2 The PrOgO Microworld

PrOgO was created to address the needs of computer science teachers at the University of Clermont-Auvergne (France), where OOP is introduced in the C++ language. Teachers expressed a need for a playful tool, which is easy to use in a classroom with beginners, to represent abstract OOP basic concepts and their implementation in the C++ language. Therefore, PrOgO is designed with the intention to help beginners to understand the conceptual model triggering the OOP paradigm basics.

In terms of game purpose, it is expected that the player builds and animates 3D graphical robots or mechanical structures. In terms of metaphor and learning content integration, each 3D elementary graphic consists of a visual representation of a "computing object" (an instance of a "computing class"). A computing object has attributes and methods that define its appearance and also behaviours. This is what is represented, to be learned in the PrOgO interface with graphical blocks. Like a computing object, each 3D block has some properties, such as a position, a rotation and a colour. Thus, each 3D block is a constructive game piece and can be connected to another one. Connecting different blocks and operating some actions on them enables the building and animation of a 3D structure which represents an object-oriented system (Fig. 1a). Constructing and animating this structure is a metaphor of the OOP paradigm basics. Moreover, it has been demonstrated that analogies and metaphors are at the basis of human cognition [5]. Analogies and metaphors help learners to categorise and to capture the essence of concepts. We expect that such an approach will effectively help to overcome difficulties faced by students with OOP.

The PrOgO's interface includes a 3D scene synchronised with a code completion editor. Programming statements are generated when actions are performed on the 3D graphics. The code editor is exclusively based on auto-completion. It enables students to avoid syntax errors and to be focused on the relationship between concepts and their implementation (coding) (Figs. 1b and c). Moreover, the result of each completed statement is immediately visualised in the 3D scene.



```
#include "Base.hpp"
#include "Cube.hpp"


void main () {
    Base base;

    Cube cube_0;
    cube_0.connecter(0,base,5);
    cube_0
}
```

(a) (b) (c)

Fig. 1. (a) A 3D graphic built with PrOgO representing an Object-Oriented system; (b) and (c) Interactions with objects in PrOgO respectively within its 3D scene and code editor

3 Research Question and Objective

The objective of this empirical study consists of the identification and the characterisation of students' behaviours when playing with PrOgO. We want to know to what extent the students' behaviours may offer them the opportunity to understand the core concepts of OOP. We address the following question: What characterises the students' behaviours when playing with PrOgO?

We hypothesise that learners should spend time on the PrOgO interface, and manipulate most of the learning concepts both within the 3D scene and the code editor. Such behaviour should be triggered by students, to help us to verify the educational value of PrOgO regarding learning and teaching. This behaviour would help students to understand the link between the formal abstract basic concepts of OOP (the code editor) and the visual representations of the concepts through the PrOgO interface (the 3D scene). Consequently, learners would be able to analyse the OOP paradigm and to understand the addressed concepts.

4 Methodology

The study is based on playing analytics [6], which is an emerging sub-field of learning analytics. Learning analytics offers the opportunity to practitioners and researchers for measuring, collecting, analysing and reporting data about learners [7]. This enables monitoring of learners' activities and assessing innovative educational approaches. We conducted a statistical analysis of the actions performed by the students, based on their digital traces. They consisted of event logs that recorded learners' interactions with the PrOgO interface. By monitoring students' use of PrOgO we expected to investigate whether the game was used in a way that could help them learn, then to verify its educational value. In order to determine typical learners' behaviours, and the set of performed action types underlying similarities and differences between learners, we conducted a Principal Component Analysis (PCA) [8] and an Agglomerative Hierarchical Clustering (AHC) [9] on data built from the collected traces.

4.1 Context of the Study

The study was carried out with learners from two study levels: first year university learners in the Digital Imaging Degree (University of Clermont-Auvergne, France), and final year learners in a science and technology secondary school (Lycée Charles et Adrien Dupuy, France). Sixty-seven students (55 university students and 12 secondary school students) participated in the study in October 2015. They had all previously been introduced to procedural programming but had never been introduced to OOP.

The experiment lasted for one hour. Learners were given access to a tutorial describing the game objective, the interface details, and how they could perform different tasks both in the 3D scene and the code editor of PrOgO. They were asked to play individually with PrOgO and to express their creativity for constructing and animating a virtual robot or a machine. During the experiment, learners' actions were recorded and log files were enabled to keep track of their use of PrOgO.

4.2 Data Collection and Analysis

Players' interactions were recorded in log files, which included timestamped interactions in relation to the learning basic concepts of OOP: creations of objects; modifications of objects' attribute values; and method calls. These interactions could be performed both inside the 3D scene, and the code editor (code selections and code completions). In order to prepare the data analysis, we built a new data file of aggregated data. For each participant, we stored the number of different interactions, as well as the time spent on the game. This included the number of interactions performed both inside the 3D scene, and the code editor such as the code completion actions. PCA was conducted with this aggregated data (Table 1).

Table 1. PCA input variables.

PCA variable	Signification
NB of Instanciate-Connect	Number of class instantiation and objects connection actions
NB of setAttributeColor	Number of attribute colour setting actions
NB of setAttributeRotationAngle	Number of attribute rotation angle setting actions
NB of callFuncColorFor	Number of calls to the method colorForaDuration()
NB of callFuncTurnFor	Number of calls to the method turnForaDuration()
NB of CodeSelection	Number of code selection actions
NB of CodeCompletion	Number of code completion actions
Total time	Time spent by the student

The key objective with PCA is to reduce the dimensionality of a dataset with a large number of interrelated variables, while retaining as much as possible the variation present in the dataset [8]. This reduction is achieved by converting the initial variables into a new set of uncorrelated variables, called principal components. Principal components are ordered so that the first few retain most of the variation present in the dataset [7]. The principal components are also called PCA axes or factors. This reduction of variables enables the user to find the two-dimensional plane through the high-dimensional dataset in which the data are most spread out. So, data can be plotted with respect to those two dimensions, and to produce a visual representation of the data. The PCA returned the principal components with their corresponding eigenvalues, reflecting the variability of the reduced initial data. Ideally, a small number of factors with high eigenvalues are retained to ensure good visual representations of data [8]. A second important PCA result is the correlation circle. This circle shows a projection of the initial variables in a 2-dimensional space with respect to two chosen factors that are ideally the first two. Correlation refers to the degree of dependence between two variables. In our case, it is measured according to Pearson's correlation coefficient, giving a value between −1 and +1 inclusive. The closer the coefficient is to −1 or +1, the greater is the correlation between the variables. To ensure a good interpretation of the meaning of the axes, variables should be far from the centre of the circle. A variable is well linked to an axis when the absolute value of its coordinate on the axis is high. The greater the coordinate absolute value, the greater is the link with the corresponding axis.

In order to achieve our ultimate goal, which is to identify groups of learners, we conducted an AHC on the new observations' coordinates in the sub-space containing the chosen factors. Then, we conducted a second PCA on the same initial dataset to which we added the observations groups returned by the AHC, as a supplementary qualitative variable. We obtained a 2-dimensional map. To complement the visual results from this map, we looked at the coordinates of the classes' centroids with respect to the chosen factors. Centroids are the most typical observations. They have very similar characteristics to observations within the same category but are significantly different from observations belonging to a different category. Therefore, centroids can give us a general idea of the trends inside a whole class.

5 Categories of Students

The PCA returned 8 factors estimated from the initial variables listed beforehand (Table 2). The first three eigenvalues represent 61.45% of the initial variability of the data. We retained the first three factors and ignored the last ones, since the fourth factor had an eigenvalue quite similar to the third one and the last ones had very low eigenvalues.

Table 2. Principal components returned by the PCA.

	F1	F2	F3	F4	F5	F6	F7	F8
Eigenvalue	2.36	1.48	1.08	1.04	0.79	0.73	0.3	0.22
Variability (%)	29.45	18.54	13.45	13.03	9.89	9.13	3.69	2.8
Cumulative %	29.45	47.99	61.45	74.48	84.37	93.5	97.2	100

The correlation circle showed that the horizontal axis (F1) is linked with variables NB of Instantiate-Connect, NB of CodeSelection and NB of CodeCompletions, while the vertical axis (F2) is linked with the variable Total time (Fig. 2).

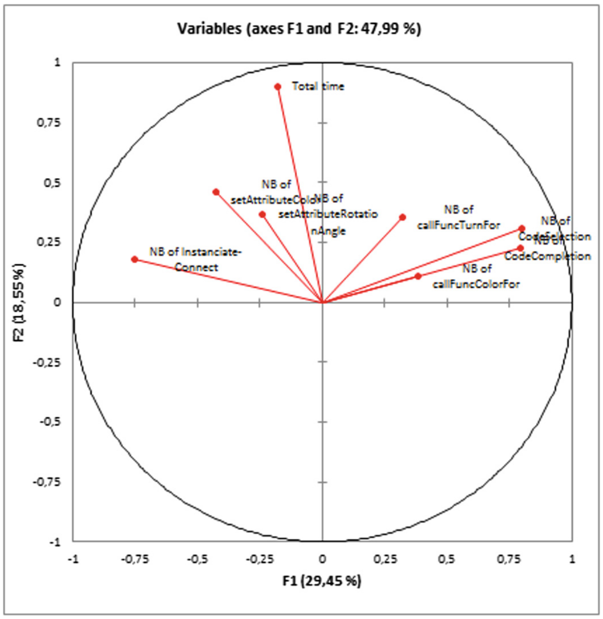


Fig. 2. Correlation circle showing the variables defining the axes F1 and F2.

Results from Table 3 confirm the set of variables defining the different axes. Variables that scored high on the factor F1 are the same listed above, with respective correlation values of −0.75, 0.79, and 0.79. F2 is positively correlated with Total time with the value 0.90, and F3 is significantly correlated with the variables NB of callFuncColorFor, and NB of callFuncTurnFor. Correlation values between these variables with F3 are respectively 0.66 and 0.62.

Table 3. Correlations between the PCA variables and the first three factors.

Variables	F1	F2	F3
NB of Instanciate-Connect	**−0.75**	0.18	0.01
NB of setAttributeColor	−0.43	0.46	−0.17
NB of setAttributeRotationAngle	−0.24	0.37	−0.01
NB of callFuncColorFor	0.39	0.11	**0.66**
NB of callFuncTurnFor	0.32	0.36	**0.62**
NB of CodeSelection	**0.79**	0.31	−0.32
NB of CodeCompletion	**0.79**	0.29	−0.34
Total time	−0.18	**0.9**	-0.03

Based on the statistical analysis, we can conclude that it is possible to distinguish students: (1) by the number of actions performed with the code editor as well as the number of class instantiation actions; (2) by the total time spent on the PrOgO interface and; (3) by the actions performed to animate the robot of the 3D-scene.

The AHC was conducted on the new observations' coordinates in the 3-dimensional space with respect to the chosen factors F1, F2, and F3. The algorithm returned four homogeneous groups showing, for each participant, the group the user belongs to. After running a PCA for a second time on the same initial variables with a supplementary qualitative variable (the group to which an observation belongs to), we obtained a 2-dimensional map (F1−F2) of the observations, coloured according to the category they belong to (Fig. 3).

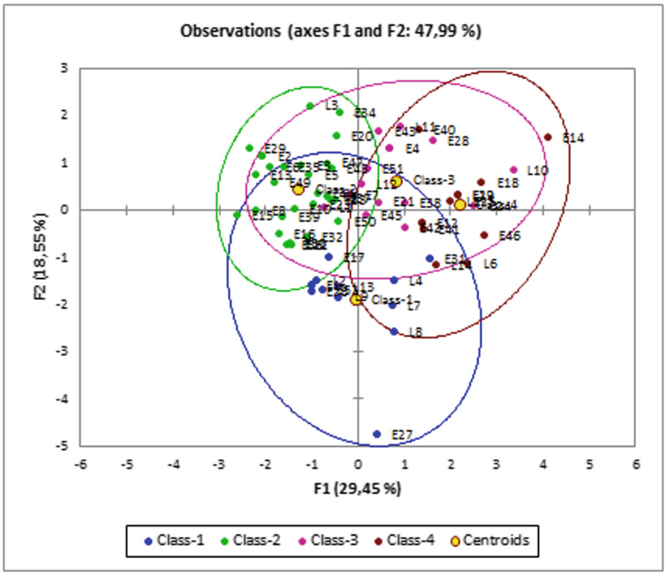


Fig. 3. Map showing the partition of the participants into 4 groups. (Color figure online)

Table 4 supplements Fig. 3. It shows the percentage of observations for each class and gives the 3-dimensional coordinates of the classes' centroids with respect to the axes F1, F2, and F3.

Table 4. Class size and central object coordinates in the subspace (F1, F2, F3).

Class	Objects (%)	Centroid (F1, F2, F3)[a]
1	18.18	(−0.42, −1.83, −0.04)
2	42.42	(−0.96, 0.13, −0.07)
3	21.21	(0.21, 0.89, 1.35)
4	18.18	(2.15, 0.31, −1.08)

[a]Class central object coordinates in the subspace (F1, F2, F3).

Class-1 objects (the blue points) are plotted in the negative semi-plane of the F2 axis, which is highly correlated with the Total time variable (Fig. 3). This indicates that class-1 members are students who have spent little time playing with PrOgO. They have low coordinate absolute values on the F1 axis, which means that they performed a small number of actions, both within the code editor and the 3D scene. Few students belonged to this class. The class-1 centroid is defined by the coordinates (−0.42, −1.83, −0.04) in the subspace (F1, F2, F3), indicating first, that the user has performed a small number of actions linked to the factors F1 and F3, and second, has spent little time playing with PrOgO (−1.83 on the F2 axis). Thus, class-1 is covered by learners who performed a small number of actions, since they have spent little time playing with PrOgO. They represent 18.18% (Table 4) and may correspond to learners who were not very interested in playing with PrOgO.

Class-2 objects (the green points) are plotted in the negative semi-plane of the F1 axis and mostly in the positive semi-plane of the F2 axis (Fig. 3). This indicates that these class members performed a high number of instantiation actions in the 3D scene, and no or very few actions within the code editor. The central object from Table 4 is defined by the coordinates (−0.96, 0.13, −0.07) in the subspace (F1, F2, F3). It represents a student who created a lot of objects in the 3D scene and connected them to each other without using the code editor. The user also played close to the mean time of the experiment and performed no or very few animation actions (as the coordinate is close to 0 on F3). As a result, class-2 members are students who constructed a 3D robot, without using the code editor. They are numerous (42.42%) and have nearly used all the time allocated to the experiment without animating their realisations.

Class-3 objects (the magenta points) are plotted in the positive semi-plane of F1 and F2. According to the F1 axis, they have lower values compared to class-4 members, and greater values compared to class-2 members. They are also located near to the variables related to the animation actions, namely NB of callFuncColorFor, and NB of callFuncTurnFor. Therefore, class-3 members stand out by having: (1) not a high number of coding actions compared to class-4; (2) not a high number of construction actions compared to class-2; (3) and finally a high number of animation actions compared to class-2 and class-4. This can be confirmed by the central object having the coordinates (0.21, 0.89, 1.35). It represents a student who used the code editor, has spent time playing with PrOgO and has animated his construction. Class-3 members are also not numerous, they represent 18.18% (Table 4).

Class-4 members (the red points) are located in the positive semi-plane of F1. They have high values on the F1 axis and moderate absolute values on F2. Class-4 members are students who performed the highest number of coding actions and had a limited use of the 3D scene. The central object is defined by the coordinates (2.15, 0.31, −1.08), representing a student who has performed most of his actions on the code editor, has used the allocated time and has performed very few animation actions. Therefore, class 4 contains participants that are differentiated by the highest number of coding actions compared to the rest of participants.

6 Discussion

Based on the statistical analysis, we obtained four categories of learners who play differently with PrOgO. A first important result is that a majority of students had mostly accepted to play with PrOgO, since those who had spent little time on its interface represented a low percentage of students (Class-1) (Table 4). The majority of students used most of the time allocated to achieve the game's objective in three different ways: students who were focused on the 3D scene with the goal to construct a meaningful 3D structure (Class-2); those who were both focused on the 3D scene (high number of animations) and the code editor (a high number of coding actions) (Class-3); those who performed the highest number of code completion actions, and low construction and animation actions within the 3D scene (Class-4). Thus, one group of learners was focused on constructions within the 3D scene, whereas another was focused on the code editor with a varying number of code completion actions.

One important result is that only a few students manipulated most of the learning concepts (Class-3). These participants performed a high number of animation actions (method calls concept), in addition to actions of construction within the 3D scene (objects' creation concept and modification of objects' attributes values concept). They were also characterised by code completion actions. From an educational perspective, this behaviour is expected. Indeed, making links between connected and animated 3D-objects, as metaphors of most of the represented OOP concepts and their coding, should help beginners to understand these concepts and how they are coded. Consequently, the hypothesis that learners should spend time playing with PrOgO to manipulate most of the represented learning concepts is confirmed for a limited number of students (Class-3). This may be due to the fact that students can perform the goal's game both by graphics and by code. They can master easily the interface while focusing only on graphics (Class-2) or on code (Class-4). From an educational perspective, these behaviours might have other values. Further studies are needed to enable this to be identified and assessed. In addition, as a design improvement of PrOgO, further usage scenarios enabling the expected behaviour should be strengthened.

7 Conclusion

In this work, we have presented the PrOgO microworld dedicated for learning OOP to beginners. We have described our methodology and the results of a study we conducted to investigate learners' behaviours when interacting with the programming concepts represented in this microworld. Our methodology is based on playing analytics, by recording and analysing the interactions of players. We processed a PCA and an AHC in the analysis of data, leading to the characterisation of different groups of learners on the basis of their behaviours when playing with PrOgO. The results showed that the behaviour which is expected from an educational perspective, was triggered by a limited number of students. Behaviours triggered by most students might have other educational values which need to be identified and assessed in further similar studies. Moreover, improvements in the design of PrOgO is needed to strengthen the expected behaviour for which the educational value is known.

The contribution of the paper also consists of the methodology used for the evaluation of the use of PrOgO. This responds to the lack of standardised methodologies for evaluating tools that support computer science education and programming in particular [10]. It is also in line with the current concerns of the computer science education research community. Indeed, querying the ACM Digital Library database with the expression “Programming Learning Analytics” returns 16,562 results (query performed in March 2018). We perceive a high interest in this methodology, since it considers a learners’ attitudes analysis, instead of collecting learners’ points of views or observing their score progression by means of knowledge tests or questionnaires. Moreover, this methodology is an alternative to comparatist approaches: instead of comparing the efficiency of a digital learning environment, we proceed with the analysis of learners’ use of the environment.

Acknowledgments. The authors would like to thank Jeane Fine for her advice in the statistical analysis, Dominique Moncorgé, Adélaïde Albouy-Kissi and Benjamin Albouy-Kissi for the scheduling of the experiment, and Delphine Huguel for the development of the PrOgO microworld. This work has been financed by the French Ministry of National Education, Higher Education and Research within the Tactileo E-Education project (2013–2016).

References

1. Börstler, J., Nordström, M., Kallin Westin, L., Moström, J.-E., Eliasson, J.: Transitioning to OOP/Java—a never ending story. In: Bennedsen, J., Caspersen, Michael E., Kölling, M. (eds.) Reflections on the Teaching of Programming. LNCS, vol. 4821, pp. 80–97. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77934-6_8
2. Djelil, F., Albouy-Kissi, A., Albouy-Kissi, B., Sanchez, E., Lavest, J.-M.: Microworlds for learning object-oriented programming: considerations from research to practice. J. Interact. Learn. Res. **27**(3), 265–284 (2016)
3. Djelil, F.: Conception et évaluation d’un micromonde de Programmation Orientée-Objet fondé sur un jeu de construction et d’animation 3D. Ph.D. thesis, University of Clermont Auvergne (2016)
4. Djelil, F., Sanchez, E., Albouy-Kissi, B., Albouy-Kissi, A.: Acquisition de connaissances de programmation en fonction des stratégies d’apprentissage: une étude empirique du micromonde PrOgO. Environnements Informatiques pour l’Apprentissage Humain (EIAH), Strasbourg, France, pp. 41–52 (2017)
5. Hofstadter, D., Sander, E.: Surfaces and Essences: Analogy as the Fuel and Fire of Thinking. Basic Books, New York (2013)
6. Sanchez, E., Mandran, N.: Exploring competition and collaboration behaviors in game-based learning with playing analytics. In: Lavoué, É., Drachsler, H., Verbert, K., Broisin, J., Pérez-Sanagustín, M. (eds.) EC-TEL 2017. LNCS, vol. 10474, pp. 467–472. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66610-5_44
7. Siemens, G., Long, P.: Penetrating the fog: analytics in learning and education. EDUCAUSE Rev. **46**(5), 30–38 (2011)
8. Jolliffe, I.: Principal Component Analysis. Springer, Heidelberg (2002). https://doi.org/10.1007/b98835
9. Day, W.H., Edelsbrunner, H.: Efficient algorithms for agglomerative hierarchical clustering methods. J. Classif. **1**(1), 7–24 (1984)
10. Djelil, F., Boisvert,C., Peter, Y., Sceq, Y., Broisin, J., De La Higuera, C.: Vers une massification de l’apprentissage instrumenté de l’informatique et une intégration des instruments de l’informatique et de leur évaluation, Orphée Grand Challenges (2017)