Olympiads in Informatics, 2013, Vol. 7, 3–13

Growing Algorithmic Thinking Through Interactive Problems to Encourage Learning Programming

Sébastien COMBÉFIS[1,3], Virginie VAN den SCHRIECK[2], Alexis NOOTENS[3]

[1]*Department of Computer Science Engineering, Université catholique de Louvain*
Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium
[2]*École Pratique des Hautes Études Commerciales (EPHEC)*
Avenue du Ciseau 15, 1348 Louvain-la-Neuve, Belgium
[3]*Computer Science and IT in Education ASBL, Belgium*
e-mail: sebastien.combefis@uclouvain.be, v.vandenschrieck@ephec.be, alexis.nootens@csited.be

Abstract. Attracting pupils from secondary schools (12–18 years old) to learn programming is not easy. It is especially the case in Belgium where there is no or very few programming and algorithm design courses in secondary schools. Another issue is that teachers who are in charge of computer science courses are afraid of teaching a matter they do not feel comfortable with, especially when they are not informatics teachers. This paper presents ILPADS, *interactive learning of programming and algorithm design skills*, an interactive website which aims at gradually growing algorithmic thinking skills to lead pupils towards the learning of the Python programming language. That website aims to serve as working material to support teachers for their computer science courses in secondary schools. Pupils can also use the website at home to continue learning on their own. The paper presents the interactive website and mainly focuses on the design of the ILPADS activities. Future work includes testing the website in real classrooms and evaluating it.

Key words: algorithmic thinking, learning programming, teaching, distance learning, interactive learning.

1. Introduction

Trying to attract more pupils to participate at the International Olympiad in Informatics (IOI) is not an easy task. It is especially the case in countries like Belgium where there are no or only a few computer science related courses in secondary schools (12–18 years old). Although there are pupils that are good at programming and designing algorithms, some of them may just ignore it and they consequently will not participate to the selection for the IOI (Combéfis and Leroy, 2011).

Moreover, there are no or few computer science teachers in the secondary schools of those countries where informatics is not part of the curriculum. That latter fact does not ease the task of introducing computer science and especially programming to the pupils. In such countries, it is even more difficult to develop and propose activities to be organised by the teachers in their classrooms, since they do not feel comfortable with the material to teach.

One possible way to propose activities to pupils is through online platforms. Various platforms, such as Pythia (Combéfis and Le Clément de Saint-Marcq, 2012), Putka (Urbančič and Trampuš, 2012) or France-IOI (Hiron and Février, 2012) do exist but are mainly focused on directly teaching programming. Most of the time, pupils and even teachers do not have any idea about what is behind the word programming and the notion of algorithm design. However, proposing online self-contained activities helps teachers to support taught courses and provides the possibility for pupils to continue learning at home.

Before teaching programming to pupils, it is important to develop their ability to think algorithmically. Computational thinking (Wing, 2006), also referred to as algorithmic thinking (Futschek, 2006), is a key ability that can be learned independently from programming, and that is maybe easier to introduce in secondary school. It has been defined as: *"the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent"* (Cuny *et al.*, 2010).

It is possible to teach and encourage algorithmic thinking without using a computer by proposing pen-and-paper exercises, as it is for example done in the frame of the Australian Informatics Competition (Burton, 2010) or for the Bebras contest (Futschek and Dagiene, 2009). The philosophy behind those concepts is that the learners do not need any additional background than the one they have while being in their schools. Pursuing those activities drives them towards developing algorithm design skills, but not programming skills.

This paper proposes learning activities supported by an interactive website whose goal is to develop algorithmic thinking amongst pupils. The activities are first designed to improve the problem solving skills of the learners. They also gradually drive the learners towards programming skills, as an embodiment of the reasoning they developed beforehand.

Section 2 draws up related work where activities are developed either online or in a pen-and-paper fashion, to teach algorithmic thinking to secondary school pupils without any prior knowledge in computer science. Section 3 presents ILPADS and focuses in particular on the design of activities, with a concrete example. It also motivates the design according to educational sciences theories. Section 4 discusses how the approach will be evaluated. Finally, the last section concludes the paper with some perspectives.

2. Related Work

This section presents related work on activities and tools that have been developed in order to develop algorithmic thinking by pupils from secondary schools.

Algorithmic thinking is somewhat different from natural thinking. Both consist of finding a solution to a problem, but whereas in everyday life problem solving is for humans, algorithmic thinking consists in finding a solution meant to be encoded in a computer. Several abilities are part of algorithmic thinking, including analysing the problem,

finding basic actions that are adequate and constructing the algorithm with the basic actions (Futschek, 2006).

Futschek and Moschitz have been working on activities where learners can play algorithms, either virtually or by themselves (Futschek and Moschitz, 2010) or with tangible objects (Futschek and Moschitz, 2011). Their work focuses on the fact that the concepts of algorithmic thinking must be reduced to natural thinking for beginners. Learners are playing algorithms themselves, acting like intelligent processors than can execute algorithms. A model for learning by inventing algorithms proposed in Futschek and Moschitz (2010, 2011) proposes activities targeted at primary schools where pupils can manipulate tangible objects to discover the notion of algorithm. In those two approaches, the learners will not have to write an algorithm using a programming language, but will directly play with it. The activities proposed in this paper follow the same philosophy of thinking about algorithms by playing with them. But in this paper, the learners are driven up to programming their algorithms.

Other activities that have as a goal teaching algorithmic thinking to pupils are the ones proposed by CSUnplugged (Bell *et al*., 2009). The proposed activities cover various subjects in computer science from numbers representation with binary numbers to cryptography. They are meant to be organised by a trainer with a group of pupils. In opposition to the activities proposed in this paper, the role of the trainer is essential for CSUnplugged activities.

Finally, as already introduced, another way to teach algorithmic thinking is through contests (Burton, 2010; Futschek and Dagiene, 2009). In those approaches, the pupils are first confronted to the problems on their own. Teachers are not obliged to go through the questions of the contest with the pupils afterwards.

3. The ILPADS Website

This section presents ILPADS, a website which aims at supporting the learning of algorithm design skills and programming, through interactive problems.

3.1. *General Presentation*

ILPADS is a website that proposes a set of learning activities. Each activity has, as a main goal, to develop the ability for the learners to solve an algorithmic problem. Activities are centred around concrete problems and are decomposed into three stages of increasing difficulty. At each stage, the learners get a new understanding of the problem and its solution.

Figure 1 shows the three stages of an ILPADS activity. They are not all mandatory and the learners can stop at any stage. However, progressing through the stages will lead the learner towards a solution in a programming language, which is the intent in a recruiting prospective for the selection for the IOI.

In the first stage, the learners are confronted to an interactive animation allowing them to play with an instance of the problem. It allows them to discover the algorithm and build

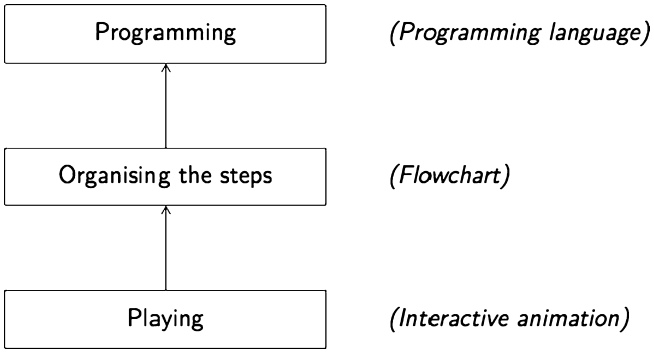

Fig. 1. The three stages composing an ILPADS activity. The stages drive the learner from the simple understanding of the algorithm to writing it in a programming language.

it in their mind. In the second stage, they have to concretise the algorithm they have in their mind. They do it with an executable flowchart that can be run on an instance of the problem. Finally, the last stage allows the learners to write a program representing their algorithm.

In order that decomposition to be possible, the chosen problem should not be too hard. It should indeed be ensured that the learners will get an algorithm idea during the first stage, that is, they have a preconceived, possibly wrong, idea in their mind. The first stage is therefore the crucial point in the design of ILPADS activities.

Each stage has a precise final objective in terms of what the learners will develop as skills. There is also a motivation for each step, related to the utility and usefulness of the trained skills.

1. Playing interactively on instances of the problem helps the learners understand the problem and guides them towards an algorithm. That stage is used as a mental gymnastics. After having played, the hypothesis is that it will be easier for the learners to solve new instances of the same problem.
2. The organisation of the algorithm found in the first stage into steps using flowcharts helps the learners take the algorithm out of their mind and concretise it. For the learners, it is the first step towards programming as they have to communicate their algorithm to the computer.
3. Finally, the learners will get to the writing of their algorithm using a programming language. That step is somewhat the holy grail as it will take the learners from a reasoning in their mind to a concretisation inside the computer.

3.2. *Activities Design*

This section presents how the different stages are implemented and what technologies are used to support them. The description is illustrated with an ILPADS activity example that is related to sorting algorithms.

3.2.1. *Playing Interactively*

In the first stage, the learners have to be able to play with the problem to be solved. Animations are important to provide an additional view of the problem. They help to understand algorithms better than a traditional textual or pictorial presentation (Rodger, 1996). ILPADS activities propose interactive animations that the learners can play with. The first stage is composed of a sequence of interactive animations that follow each other as in a comic strip (Biermann and Cole, 1999).

The learners are first completely free to play with the animation, and then some instructions are given to them in order to make an algorithm appear in their minds. As they are progressing in the story of the comics, their idea about an algorithm to solve the problem should be growing and becoming clearer in their mind.

Figure 2 shows the first interactive animation of the comics that is used for the sorting example. For that ILPADS activity, the learners are faced to a set of seven bottles, each with a different weight going from 1 to 7 ounces. The learners have to sort the bottles in increasing order of weight. To help them, they can use the provided machine which, once set with a reference weight, will output the lighter bottles on the left, the heavier ones on the right and the ones with the reference weight on the front.

The learners are completely free to play with that first interactive animation of the comics, to solve the problem, that is, sorting the bottles. The learners have to put the bottles in the right order on the result tray (not shown on the figure) and can check whether their solution is correct or not.

Since all the bottles have distinct weights, a solution to sort them is quite easily found. For example, the learners can successively set the machine with every value between 1 and 7, each time finding the next bottle to place on the result tray, which allows them to find the correct ordering.

The second box of the comics goes one step further. Now, the bottles do not necessarily have distinct weights. The weights are still between 1 and 7 ounces, but it may be the case that more than one bottle has the same weight.

Figure 3 shows a possible situation where the learners have set the machine with a reference weight of 3. Two bottles have that weight, two are lighter and three heavier.

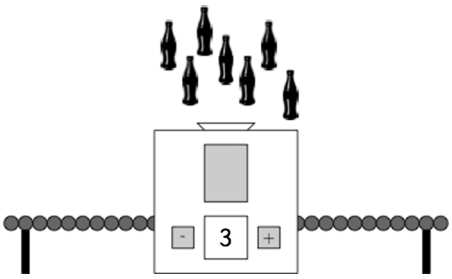

Fig. 2. The interactive animation of the sorting ILPADS activity. With that animation, the learners can ask the machine to group the bottles according to a reference value they chose. The goal for the learners is to sort all the bottles in increasing order of weight.

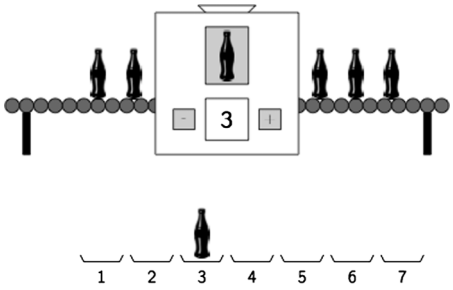

Fig. 3. The interaction animation of the comics for the sorting ILPADS activity. Contrary to the first interactive animation, there may now be bottles with the same weight. The learner cannot just rely on the position in the result tray to place the bottles.

One of the bottles weighing 3 ounce has already been placed on the result tray. The second bottle should be placed on position 4 since there are only two lighter bottles.

The idea with the second interaction animation of the comics is to drive the learners towards the selection sort algorithm. The learners should configure the machine with increasing values to fill the result tray from left to right. It is of course not the only possible solution, but the more convenient and easy to explain.

A comic strip always comes with additional textual information. Each box of the comics is enriched with textual information, coming as real-time feedback that can appear either when the learners check whether their solution is correct, or after any targeted action.

For example, for the sorting ILPADS activity in the interactive animation of the comics, the learners may get the following textual message when checking their answer:

> *"Congratulations, you succeeded in ordering the bottles in increasing order of weigh. The machine has done a total of 28 weight comparisons*."

That success message adds information about the comparisons that have been done by the provided machine. It serves as an indirect way to sensitise learners to performance issues. For the second interactive animation of the comics, the learners may be confronted to the following message when moving a bottle in the result tray:

> *"You are moving a bottle in the result tray, for the fourth time. Are you sure you cannot avoid it?"*

The goal of that message is to draw the attention of the learners on the fact that the way they are proceeding is not necessarily the best one. Another kind of textual information that may appear when the learners place a bottle on the result tray is:

> *"You placed a bottle weighing 3 ounce on position 2 but there are two lighter bottles. Are you sure it is a correct position?"*

Again, that message is used to lead the learners towards a correct solution. The added textual information is very important to support the learning and help the learners to find the algorithm to solve the problem. This is discussed in Section 3.3.

3.2.2. *Drawing Flowcharts*

Once the learners have played with the interactive animation and have gone through the different interactive animations of the comics, they are ready to get their algorithm out of their minds.

The idea of that second stage is to use the same animation as the one used in the previous stage. The difference is that learners will not be able to directly interact with the animation. The only way to control the animation is though a flowchart they have to design. It is essentially the same idea as the one of Scratch (Maloney *et al*., 2004). It allows the user to build an algorithm visually by choosing and organising together blocks representing instructions or control structures.

Figure 4 shows a flowchart for a correct algorithm for the first interaction animation of the sorting ILPADS activity. There are two kinds of elements: diamond-shaped boxes are used to make a decision and rectangular boxes represent actions. Those two kinds of elements make it possible to represent all the basic operation of computer programs, that is, conditionals, loops and sequences of actions.

The actions can be parametrized with values that are directly related to the interactive animation, so that the learners can see directly the link with the animation. Moreover, it is important that the different actions available to design the flowchart are related to the ones the learners used during the first stage. In order to present to the learners the different possible actions, whenever the learners are clicking on the different elements of the interactive animation, a list of the possible actions is presented to the learners.

At any time, the learners can execute the flowchart and see directly the result on the animation. For the learners to succeed that stage, they have to design a flowchart that can solve any instance of the problem.

The goal of an ILPADS activity is not to directly teach learners how to use flowcharts, how to compose the different elements or link them together. However, provided that

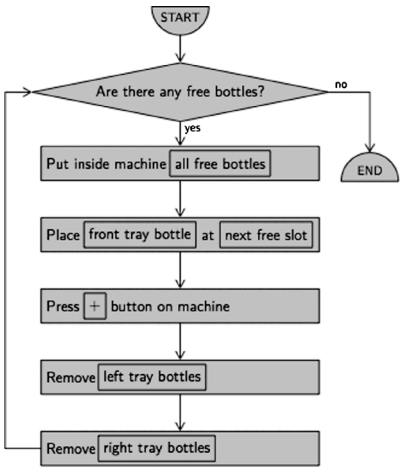

Fig. 4. Flowchart diagram that represents a correct algorithm for the first situation of the sorting ILPADS activity.

a simple flowchart example is given to the learners and since they are able to execute them and visually see the result of the execution, it could be able to teach learners to use flowcharts with a set of simple ILPADS activities based on very simple problems.

3.2.3. *Writing Down the Program*

The last stage consists of writing down the algorithm using a programming language. That stage should be made easier by the previous stage where the learners were forced to think carefully and structure their algorithm. During that stage, the learners will write their algorithm with the Python programming language.

For the learners, that final stage essentially consists in translating the flowchart designed in the previous stage into a Python program. To do so, Python functions corresponding to the actions from the flowchart are made available to the learners. The main goal is thus essentially to translate a flowchart to a computer program. Reaching that stage fulfils the main goal of the ILPADS website, which is to drive the learners from thinking in their minds up to writing their solution with a programming language.

That latter stage is supported by Pythia (Combéfis and Le Clément de Saint-Marcq, 2012) which is an online platform that can safely execute computer programs, test them and provide intelligent feedback about their correctness.

3.3. *Behind the Scene*

The interactive animations are developed using the HTML5 canvas element that is used to render 2D shapes (Smith, 2012) and Javascript that is used to animate the shapes and to interact with the user, in order to get an interactive animation.

The animations supporting the first stage are developed completely from scratch for each ILPADS activity. The Javascript code used to manage the interactive animation is decomposed in functions to have a direct mapping with the basic blocks provided to build the flowcharts. For the sorting ILPADS activity, such a function is the one executed when the user presses on the plus button (+) to increment the reference weight.

The flowcharts are also built with the canvas element of HTML5. The flowchart itself is represented as a linked structure whose nodes are related either to the functions defined for the animations of the first stage, or to functions that are especially defined for the flowcharts. An example of that latter kind of function is the one corresponding to the remove action of the sorting ILPADS activity.

Finally, as already mentioned in the previous section, the last stage is fully supported by Pythia. The programs written by the learners are checked thanks to the capabilities of the Pythia. In fact, the flowcharts designed during the second stage are checked the same way. Flowcharts can indeed be automatically translated into a Python program that is then checked by the Pythia.

3.4. *Supporting Learning and Ensuring Motivation*

The objective of the ILPADS activities is to help pupils learn new skills. Two main elements are at the heart of the design choice that was made for this work. The first one

comes from active pedagogical learning methodologies supported by the *learning by doing* motto (Dewey, 1938). By placing the learners at the heart of the learning process, and by allowing them to play with an interactive animation, it increases their motivation and involvement.

The second important element that plays a role in the learning process is the feedbacks. Pupils get feedback through the interactive animations in the first stage and through the execution of the flowcharts in the second stage. For the third stage, feedbacks are integrated within those provided by the Pythia platform. Feedbacks are also important to lead the learners towards a correct algorithm which solves the problem.

Finally, the motivation of the learners is improved by ensuring a high level of self-efficacy (Bandura, 1977). Self-efficacy can be raised through four factors as identified by Bandura. First of all, success raises self-efficacy, and the ILPADS activities are driving learners towards success, in particular through the feedbacks. The second factor that enhances self-efficacy is the possibility for the learners to compare themselves with other learners in the same situation. That is possible, from the second stage where pupils have concretised their algorithms. The learners will get the possibility to visually compare their solutions, and for example to discuss together in class.

4. Evaluation Plan

The proposed activities have not been evaluated yet. Evaluation is important when designing new kind of activities. This section briefly presents the evaluation that is going to be done, which is clearly a future work.

The idea is to measure whether pupils that had previously trained with the ILPADS website are advantaged whenever confronted with a new instance of a given problem. The hypothesis is that pupils who played with the ILPADS website before do have a better structured and general algorithm in their minds than pupils who did not get that chance. The textual feedbacks that are brought for each interactive animation of the comics bring that advantage.

Experiments include confronting two groups of pupils with a set of problems. The pupils from the first group will use ILPADS and the pupils from the second one will just have a pen and a paper. Then, both groups will be confronted with new instances of the problem and their performance will be evaluated. Performance includes the rate of correct answers and the time used to solve the problems. The hypothesis is that ILPADS will help the pupils from the first group to structure the intuitive idea they have in their minds so as to be able to apply it to new instances of a problem they previously trained on.

Another important potential issue that may be raised is that by restricting what the learners can do with the flowcharts, it may restrict the learners' algorithmic thinking. The possible actions correspond to the one the learners can execute on the interactive animation in the first stage so that the only potential restriction may come from the conditions. Analysing the solutions proposed by the second group of pupils may help the designer of the activities to detect such missing conditions.

5. Conclusion and Perspectives

This paper proposes the ILPADS website that can be used to support teachers to propose computer science related courses in secondary schools. The goal of the website is to develop algorithmic thinking through problem solving that is supported by interaction animations. The learners are guided through three stages that develop their programming and algorithm design skills. Finally, at the end of an ILPADS activity, learners get to produce a solution to the problem as a computer program.

Providing online self-contained learning activities allows pupils to learn at home. This website can be used as an incentive to learn programming and, for example, to recruit more potential candidates for the IOI. It also helps teachers that are not comfortable with computer science and have to propose related activities in their schools.

Future work about ILPADS includes putting the components of the different stages altogether into one unique website. Also, developing an ILPADS activity takes a lot of time and new activities are already being designed. One future activity is about the "Guess Who?" game and another one is about the 15-puzzle (or Gem Puzzle) game. In addition to the design of new activities, work has to be done to ease the practical realisation of the two last stages that can be automated a lot, by developing an ILPADS activity creator. Last but not least, the approach has to be tested and evaluated as described above in the paper. More generally speaking, for a given activity, much attention has to be paid during the design so as not to restrict and limit the learners algorithmic abilities. Perspectives include a deeper study of how to assess whether a given ILPADS activity is well designed or not, that is, to evaluate its quality.

References

Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological Review*, 84(2), 191–215.

Bell, T., Alexander, J., Freeman, I., Grimley, M. (2009). Computer science unplugged: school students doing real computing without computers. *Journal of Applied Computing and Information*, 13(1), 20–29.

Biermann, H., Cole, R. (1999). *Comic Strips for Algorithm Visualization*. Technical report.

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

Combéfis, S., Le Clément de Saint-Marcq, V. (2012). Teaching programming and algorithm design with pythia, a web-based learning platform. *Olympiads in Informatics*, 6, 31–43.

Combéfis, S., Leroy, D. (2011). Belgian olympiads in informatics: the story of launching a national contest. *Olympiads in Informatics*, 5, 131–139.

Cuny, J., Snyder, L., Wing, J. (2010). Demystifying computational thinking for non-computer scientists. Work in progress.

Dewey, J. (1938). *Experience and Education*. New York, The Macmillan Publishing Company.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In: *Proceedings of the 2nd International Conference on Informatics in Secondary Schools: Evolution and Perspectives (ISSEP 2006)*, 159–168.

Futschek, G., Dagiene, V. (2009). A contest on informatics and computer fluency attracts school students to learn basic technology concepts. In: *Proceedings of the 9th World Conference on Computers in Education (WCCE 2009)*.

Futschek, G., Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. In: *Proceedings of the 2010 Constructionist Approaches to Creative Learning, Thinking and Education: Lessons for the 21st Century (Constructionism 2010)*.

Futschek, G., Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. In: *Proceedings of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP 2011)*, 155–164.

Hiron, M., Février, L. (2012). A self-paced learning platform to teach programming and algorithms. *Olympiads in Informatics*, 6, 69–85.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M. (2004). Scratch: a sneak preview. In: *Proceedings of the 2nd International Conference on Creating, Connecting, and Collaborating through Computing*. Kyoto, Japan, 104–109.

Milková, E. (2012). Development of algorithmic thinking and imagination: base of programming skills. In: *Proceedings of the 16th WSEAS International Conference on Computers*.

Rodger, S. (1996). Integrating animations in courses. In: *Proceedings of the 1st Conference on Integrating Technology into Computer Science Education ItiCSE 1996*, 72–74.

Smith, M. (2012). *HTML: The Markup Language (an HTML Language Reference)*, `http://www.w3.org/TR/html-markup/`

Urbančič, J., Trampuš, M. (2012). Putka – a web application in support of computer programming education. *Olympiads in Informatics*, 6, 205–211.

Wing, J. (2006). Computational thinking. *Communication of the ACM*, 49(3), 33–35.

S. Combéfis is a PhD Student at the Université catholique de Louvain in Belgium and works as a teaching assistant for the Computer Science Engineering Department. He is also following an advanced master in pedagogy in higher education. In 2010, he founded, with Damien Leroy, the Belgian Olympiads in Informatics (be-OI). He is now part of the coordinating committee that is in charge of managing everything which is related to the national contest. In 2012, he founded the CSITEd non-profit organisation which aims at promoting computer science in secondary schools and which is in charge of organising the Bebras contest in Belgium.

V. Van den Schrieck obtained her PhD in engineering in December 2010 from the Université catholique de Louvain. She is now professor in college and provides networking training. She has always been interested into computer science education and recently joined the volunteers who are working on projects for the CSITEd non-profit organisation with as a goal to promote computer science in secondary schools.

A. Nootens is studying computer science at Université catholique de Louvain. He is now a first year bachelor student. He has worked and is interested in web development. In particular he has a growing interest in HTML5 and related technologies such as the canvas. Being freshly graduated from the secondary school, he brings a fresh view about how computer science is perceived there and is interested in developing activities that may be used by secondary school teachers. He is also volunteering for the CSITEd non-profit organisation, and is in particular working on the ILPADS project.