Resources and textbooks for computer science education in French primary schools

Isabelle Vandevelde, Cédric Fluckiger, Sandra Nogry

IARTEM · e-journal

Vol. 14 no. 1
Peer-reviewed research article

DOI: https://doi.org/10.21344/iartem.v14i1.954
ISSN: 1837-2104

Resources and textbooks for computer science education in French primary schools

Isabelle Vandevelde, Théodile-CIREL, University of Lille, France
Cédric Fluckiger, Théodile-CIREL, University of Lille, France
Sandra Nogry, Paragraphe Lab, Cergy-Paris University

Abstract

This article examines a corpus of texts that define the scope and objectives of computer science (CS) education at primary school level in France, including textbooks, curricula, and institutional documents. Faced with these new programs, and in the absence of any specific training on methods for teaching computer science, teachers have had to make do by relying on a disparate set of documents ranging from prescriptive and guidance texts, official directives and curricula, institutional documents, textbooks, and other books. This article provides an analysis of these documents from a computer science pedagogy perspective with the aim of exploring how they change and evolve through the grades of education. We begin with a transversal analysis to highlight changes in the content taught from one cycle to the next. Then, we focus on how a specific notion, the notion of loop, is introduced to students, in order to characterise how the same notion is formulated and evolves across the different textbooks. In this way, we show that loops are defined differently across textbooks, using vocabulary that is increasingly precise and connected to other areas of knowledge, without being always connected to the digital field.

Keywords: textbooks, computer science, digital, primary school

Introduction[1]

For two decades, French primary schools took a fragmented approach to computer science (CS) education, teaching it as part of other subjects rather than as an autonomous discipline (Baron & Drot-Delange, 2016). Things have changed in recent years. Arguments have been made for including CS in K–12 curriculum (Fluck et al., 2016), extending and renewing the arguments from Papert or Wing (2006). In 2015, computer science was included in the French "common core of knowledge, competencies, and culture" within the field of "languages for thinking and communicating" (MEN[2], 2015c; our translation). Since 2016, provision has been made for computer science education at primary and middle school (*collège*) levels, while new elective courses and new specialties have been introduced at high school (*lycée*) level.

A terminological clarification is in order. The translation of the French *science informatique* is not exactly "computer science" nor "programming" or "digital humanities". The terms *informatique* (computer science or informatics) and *numérique* (digital) refer to ill-defined networks of meanings,

1 This article is based on research conducted as part of the IE CARE research project funded by the French National Research Agency (ANR). Some of the results presented here were the subject of a paper delivered at the Didapro 8 conference in Lille on February 7, 2020.

2 Ministère de l'Education Nationale, the French Ministry of National Education.

which can overlap to a significant degree. Their use is largely determined by changing fashions (Baron & Boulc'h, 2012). This question largely depends upon the specificities of the French language. In this text we shall use either "*computer science*", "CS education" or "informatics" (see Fluck et al., 2016).

As is the case with the introduction of any new content in French curricula, CS education is shaped by a wide range of texts, prescriptions, recommendations, and institutional discourse. Textbooks[3] aimed at teachers and students constitute other resources available for teachers. These documents combine to form a discourse around the teaching of computer science, serving as a reference and a guide for teachers, informing their practices, and shaping their representation of what they are required to teach and why.

This article aims to describe this textual landscape (school curricula, guidelines, textbooks) with a view to identifying recurring features of how teaching content is presented to teachers.

School curricula for CS education have been a recurring topic in recent years. Many analyses of curricula have been carried out in countries with various educational traditions. For example, since 2006, computer science has been an autonomous discipline in Vietnam. Nguyen Thi Hong & al. (2010), focusing on online sharing tools between teachers, point out that CS-related school curricula are organised through 12 main contents (operating system, algorithmic, word processing, etc.). Otero & Baron (2010) compare ministerial prescriptions of informatics between Argentina and France. They highlight that in Argentina, informatics curricula are organised by skills to be acquired. These skills appear to be independent of the underlying computing concepts. In French-speaking Belgium, CS curricula seem to be as heterogeneous as the organisation of the Belgian educational system. Joris & Henry (2010) identify six teaching programs in computing. Recently, Slot, Lorentzen & Hansen (2021) focused on the identity, content and practice of "technology understanding" as it is thought and expressed in different contexts.

It is worth noting that the use of computers and software is generally excluded from consideration of CS education by research focusing on computational thinking (Fluck et al., 2016; Wing, 2006; Brennan & Resnick, 2012; Lodi & Martini, 2022 for a review). Practices are mainly programming practices. For instance, Brennan & Resnick (2012) distinguish three aspects of computational thinking: computational concepts, computational practices (including testing and debugging, etc.) and how programmers describe themselves.

If the study of computing in school curricula has been an international recurring topic, the content of CS textbooks has rarely been analyzed recently, though it used to be a common object in the 1980s (Means, 1987), sometimes in higher education (Lin et al., 1999). More recently, related subjects such as gender stereotypes in textbooks (Papadakis, 2018) have been studied. However, the content of primary school textbooks has received less attention in recent research. This needs further investigation.

After presenting the theoretical framework and methodology used to select and analyse the texts, the article considers how computer science is defined within the selected corpus of texts. An analysis of 1170 items (texts of exercises) in a corpus of 10 textbooks is then carried out. Finally, the article adopts a deeper qualitative approach to consider how specific content (the notion of loop) is presented in four textbooks based on the observation that the content in question is symptomatic of how computer science is presented to students in French primary schools.

Computer science education: Defining the conceptual framework

Recent developments in informatics education are approached in this article from the perspective of the French didactics of "computer science". We combine theoretical and methodological contributions from the field of *didactique des disciplines* (which study teachings and learnings within school disciplines) with contributions from computer science researchers. This allows us to better understand and characterise the types of computer science content taught in schools.

3 The recent introduction of CS content in early years, including at preschool level, where the use of school textbooks in the strict sense makes little sense, has led to publishers producing (as we will see) volumes that are not textbooks in the strict sense but that make explicit allowance for them to be used as such. In this article, we will refer to all books as "textbooks".

In our theoretical framework, we use the term "content" to refer to everything that is taught, not just knowledge related to informatics of programming; it can also refer to knowledge, know-how, values, etc. (Daunay et al., 2015, Delcambre, 2007). In other words, our aim is to describe the "content" of computer science or informatics within prescribed and recommended texts.

Among these various types of content, which activities pertain to computer science? Learning how to use a keyboard, send an email, or use a search engine may fall under computer science for some, but is unlikely to count as such in the mind, say, of a computer scientist. Since the question of establishing what counts as computer science and what does not is a matter of debate (Fluckiger, 2019a), we prefer not to consider the question. The way in which content is referred to or described in textbooks will be an object of analysis rather than an a priori assumption on the researcher's part.

Therefore, our research questions are:

1. What content related to informatics and computer science can be found in textbooks designed for French primary school students? How can it be described and categorised?
2. Since our theoretical position is that a school discipline can only be examined in relation to the wider educational goals (Reuter, 2014), how and to what extent does informatics content refer to purposes, social activities, and needs of future citizens in our corpus of textbooks?
3. In the absence of a formal curriculum for computer science in French primary and lower secondary education (Fluckiger, 2019a), how do textbooks show the evolution of content across the curriculum? The notion of loop will be examined in more depth: how content is introduced to students and how notions are formulated and evolve across different textbooks.

Methodology: Composition of the corpus

From the didactic perspective adopted here (Reuter, 2014), the disciplinary content of informatics is examined in relation to the categories of content involved. To establish the different categories of content involved in computer science education, we examine a corpus consisting of two sets of texts:

- Current curriculum dating from 2015 and aimed at children aged between 2 and 14 years in French schools, containing 55 CS-learning objectives,
- Primary-level computer science textbooks.

A sample of 10 computer science textbooks was analyzed. They were selected among the thousands available using a range of external, objectively reproduceable criteria[4]. These books target students aged 3 to 14 years in what the French education system refers to as "cycles"[5] numbered 1 to 4 (each cycle include several successive grades). The 10 books are distributed as follows: 1 book for cycle 1 and 3 books for each of the other cycles.

Once they had been selected, we split the textbooks into elementary units referred to as "textual elements". For example, a single *exercise* counts as a "textual element". In the case of texts (lessons, explanation, narration), a single *page of text* is treated as a "textual element". A given chapter may, for example, consist of 8 elements, namely 5 *pages of text* and 3 *exercises*. By proceeding in this way, we were able to

4 There are many computer science books aimed at students available in the market. For example, a search using the French term "*informatique*" ("computer science") in the online search engine of the bookstore Decitre, one of the most popular stores used by French teachers, yielded 2,283 items (in September 2021). Among criteria, we used the publication date, no earlier than 2015, and the explicit reference to the age, class/level, or cycle of the students.

5 In France, cycle 1 includes the three years of kindergarten (*petite, moyenne* and *grande* section, or PS, MS, and GS), corresponding to preschool and year 1 in England. Cycle 2 consists of cours *préparatoire* (CP), *élémentaire* 1 (CE1) and *élémentaire* 2 (CE2), corresponding to years 2, 3, and 4. Cycle 3, covering cours *moyen* 1 (CM1), *moyen* 2 (CM2) and *sixième* (6e), corresponds to years 5, 6, and 7. Cycle 4 includes *cinquième* (5e), *quatrième* (4e), and *troisième* (3e), corresponding to years 8, 9, and 10.

assign each "textual element" to a type of content, yielding a corpus of 1,170 "textual elements", including 730 *pages of text* and 440 *exercises*, a sufficient number for a first approach statistical analysis.
The remainder of the article presents the results of our corpus analyses.

Transversal analysis of *content* in 10 computer science textbooks

In order to answer the first research question, how CS content in textbooks can be described and categorised, we conducted a transversal analysis of 10 CS textbooks.

Types of textbooks

The ten selected textbooks may be categorised in three main groups:

- *Resource-textbooks* for use in class (the most common type of textbook)
- *Exercise books*
- *Children's storybooks*

Considering each of these categories, the findings show that *resource-textbooks* (n = 4) all include mainly *pages of text*. In this type of textbook, 87% (n = 619) of the "textual elements" are pages of text, while 13% are *exercises* (n = 94). The primary objective is to teach students a range of concepts relating to coding in blocks, JavaScript, HTML, and Python, and to explain how to use software platforms such as Scratch and Minecraft.

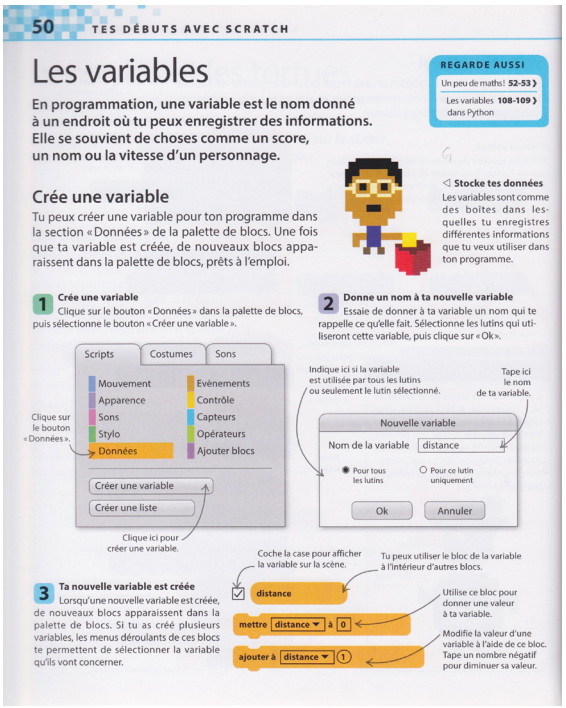


Figure 1. *Extract from the resource-textbook: Vorderman et al., 2017, p. 50, providing an introduction to coding in Scratch and Python*

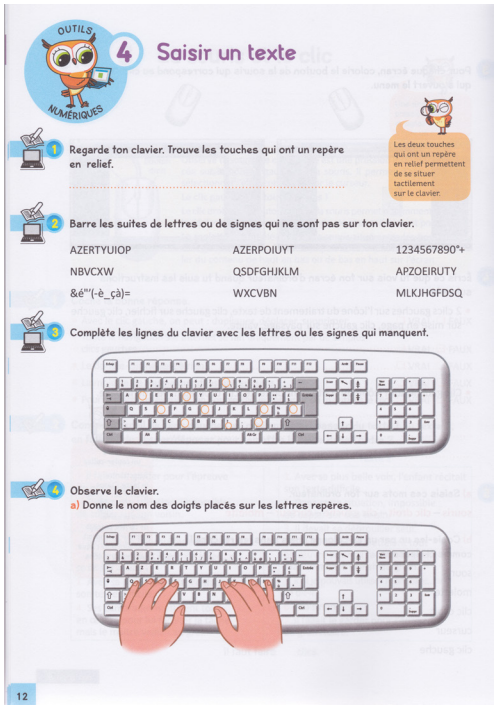


Figure 2. *Extract from exercise book: Croq et al., 2015, p. 12, including both exercises to be completed directly in the book and exercises requiring the use of a computer or tablet to learn how to use computerised tools and acquire basic coding notions*

Exercise books (n = 4) include mainly exercises, with *pages* of text accounting for just 7% (n = 22) of all "textual elements" compared to 93% (n = 311) of *exercises*. It is the specific form of the *exercises* that distinguishes *exercise books* from the other categories of books: students write their answers in dedicated spaces. The proposed exercises are designed to help students develop or consolidate their CS knowledge and skills.

In the case of books falling under the children's storybooks category (n = 2), 72% of "textual elements" are *pages of text* (n = 89), compared to 28% corresponding to exercises (n = 35). Compared to *resource-textbooks,* this type of book includes just 2.5 pages of text for every page of exercises, with the story typically being a pretext for learning and serving an activity prompt.

Finally, looking at the profile of the authors, we found that most are computer science experts (programmers, computer engineers, developers, etc.). Of the 22 authors identified, just 4 are teachers working at preschool, primary, middle school, or high school level.

Figure 3. *Extract from the resource-textbook: Vorderman et al., 2017, p. 50, providing an introduction to coding in Scratch and Python*

What categories of content do textbooks include?

Computer science helps us to categorise the different types of content in the textbooks. The tripartite characterisation of computer science proposed by Mirabail distinguishes between what pertains to science, technology, and the social and cultural agent of change (Mirabail, 1990). Dowek makes a distinction between four key computer science concepts: machine, information, algorithmics, and language (Dowek, 2012). Finally, Bruillard developed a categorisation based on different attractors: algorithm, materials and networks, and human activities (Bruillard, 2009).

This prior epistemological work on computer science and our initial analyses of the selected textbooks provided the basis for developing a typology of three categories of content: learning about how technologies work, learning algorithmics, and learning how to use computerised tools. These categories of content can be found in prescriptive texts, textbooks, curricula, and institutional texts.

Learning about how technologies work involves studying how machines operate, the history of technologies, and the issues and challenges surrounding their development. The aim may be, for example, to encourage children to "describe the simple architecture of a computing device" (MEN, 2015b, p. 67; translation ours), to show how peripheral devices connect to the central unit (Cohen & Marcialis, 2018), or to learn about the importance of informatics and the notion of loop in automated car assembly lines (Croq et al., 2015a).

Learning algorithmics, programming, languages, binary, elements of cryptography, etc. can involve, for example, getting students to read a coded message with arrows to trace the itinerary of a virtual robot on a chequerboard (Croq et al., 2015b). Here, the aim is to "code and decode in order to predict,

represent, and execute movements in familiar spaces, in a grid pattern, on a screen" (MEN, 2015b, p. 84; translation ours)

Learning how to use computerised tools aims to teach students how to use computing tools, software, search engines, etc. The aim might be, for example, to learn how to use word processing software through exercises (Cohen & Marcialis, 2018, p. 14), or how to "write on a keyboard quickly and efficiently" (MEN, 2015b, p. 111; translation ours).

We analysed computer science or informatics *content* by cross-tabulating three categories of content – how technologies work, algorithmics, and the use of computerised tools – with two other variables: cycle and type of content (exercise or lesson page).

Changes in content between cycles

How are the three categories of computer science content theoretically distributed by year?

We begin by cross-tabulating, within textbooks, the four cycles and the three types of computer science content identified.

Table 1. *Cross-tabulation: Number of "textual elements" (pages of text or exercises) in textbooks by cycle and category of content*

Cycle	Algorithmics	Use	Technology	Total
Cycle 1	84	0	1	85
Cycle 2	130	0	33	163
Cycle 3	452	299	39	790
Cycle 4	29	100	3	132
Total	**695**	**399**	**76**	**1170**

First, our findings point to an effect associated with the variable category of content. The category technology is heavily underrepresented (6%) in the textbooks examined, while algorithmics is a common feature (59%).

Second, we find a significant difference in the number of items across the different cycles, with a limited number of items in cycle 1 (7%; note that our criteria resulted in the selection of just one textbook in this cycle) and the highest number in cycle 3 (68%).

Finally, the category of content varies by cycle. Among the textbooks examined, those aimed at audiences in cycles 1, 2, and 3 focus on questions of algorithmics. The question of use is only introduced in cycle 3. Likewise, questions relating to the technological dimension of computer science are concentrated in cycles 2 and 3.

We find that half of the "textual elements" of textbooks relate to algorithmics, while one third relate to questions around the use of computerised tools. Analysis highlights an interaction between cycle and category of content ($chi2 = 278.08$, $dl = 6$, $p < 0.001$).

What about the types of computer science content in curricula? Is there evidence of a similar logic? We begin by comparing, within curricula, the four cycles and the type of computer science content identified.

Table 2. *Cross-tabulation: Categories of content by cycle in curricula*

Cycle	Algorithmics	Use	Technology	Total
Cycle 1	0	2	0	2
Cycle 2	1	4	1	6
Cycle 3	1	8	3	12
Cycle 4	16	13	6	35
Total	**18**	**27**	**10**	**55**

As with the textbooks, curricula focus primarily on questions relating to the use of computing and computer science (49%) and algorithmics (33%), and the number of items of content examined gradually increases as students move through the cycles, from 2 content items in cycle 1 to 35 items in cycle 4. More specifically, cycles 1 to 3 focus primarily on content relating to learning how to use technologies, while cycle 4 focuses on learning algorithmics.

There is also strong evidence of variation across cycles, as shown by the statistically significant interaction between cycle and category of content (chi2 = 9.02, dl = 6, $p < 0.001$) – first in terms of learning how to use IT tools, which begins in cycle 1 (introduction to a variety of tools) before being further developed in cycle 2 (dactylography), extending into cycle 3 (use of standard software), and ending in cycle 4 (use of networked tools). Second, as regards, nothing is taught in cycle 1, with learning beginning in cycle 2 (coding of movements), consolidated in cycle 3 (consolidation of specific notions), and concluding in cycle 4 (application of notions learned).

The results obtained from our analysis of curricula differ from those obtained from the analysis of textbooks. Based on the examined curricula, cycles 1 to 3 (corresponding, in France, to preschool, primary school, and the first year of secondary school) focus on the use of technologies, with algorithmics being primarily addressed in secondary school (cycle 4). Conversely, the textbooks targeting cycles 1 to 3 focus on questions around algorithmics, while cycle 4 textbooks concentrate mostly on the use of technologies.

It is worth noticing that curricula, while they may list the skills that students are expected to have developed by the end of each cycle, provide no indication of the duration and scale of a given competency relative to others.

Categories of content, types of content, and cycles

We then considered whether there is a correlation between the type of content and the type of textual element (text or exercise) evidenced in the selected textbooks.

Table 3. *Cross-tabulation: the elements page of text or exercise by type of content.*

Type of "textual element"	Page of text	Exercise	Total
Algorithmics	502	193	695
Use	180	219	399
Technology	48	28	76
Total	**730**	**440**	**1170**

Across the textbooks as a whole, there is more text than exercises (62% of text compared to 38% of exercises). this observation seems to go against the common assumption that computer science needs to be applied in order to be learned (Fluckiger, 2019a).

Of the 10 textbooks considered, 6 contain more text than exercises. In the textbooks examined, computer science is applied (in the sense that students get to "do" computer science) but also explained (i.e., students are encouraged to research computer science).

A relationship was also found between the elements *page of text or exercise* and the type of computer science content examined ($chi2 = 73.16$, $dl = 2$, $p < 0.001$). Learning how to use technologies involves exercises more than texts, while the learning of algorithmics is mostly text-based.

It might be assumed that questions of algorithmics and programming, which require a degree of practical application by students, involve exercises to a greater extent, whereas questions of use are likely to be more text-based. However, the results of our analysis of the textbooks paint a different picture, which is why we carried out a qualitative analysis of textbook content.

Transversal analysis of textbooks and curricula

Our second research question aimed at questioning to what extent informatics content referred to social activities of children or needs of future citizens. We examined the content of both th textbooks and the French CS curricula.

As argued before (Baron & Boulc'h, 2012), the French terms "informatics" and "digital" refer to distinct but complementary realities. The term "digital" has largely come to replace the term "informatics" and is "increasingly used as an equivalent and often a euphemism for what once fell under IT and software".

In curricula and institutional texts, "digital' (*numérique*), when used as a noun, is assumed to offer "many opportunities" (Struder, 2018; our translation) and to "compensate for the failings of our education system" (Institut Montaigne, 2016; our translation). These views widely reflect refuted myths associated with the digital in education (Amadieu & Tricot, 2014). These views assume that digital (generally speaking) has potentially transformative effects in schools, whereas research has long shown that this is not the case (Fluckiger, 2019b; Livingstone, 2012). When used as an adjective, the term "digital" refers to a domain (variously described as the "digital domain", MEN, 2018, "digital life" (Studer, 2018), or "digital world" (CNNUM, 2014)) that requires students to develop specific skills since "the daily lives of children are already digital" (Institut Montaigne, 2016; translation ours). In several texts, the term "digital" is used both as a noun and as an adjective, sometimes within the same sentence, as in "teaching and learning relating both specifically to digital and using digital tools and resources" (MEN, 2018).

Lastly, the term "digital" may refer to a supposedly new form of relationship to knowledge that has the effect of challenging the very function of education: "Knowledge is changing, as are its methods of transmission, as well as our relationship to knowledge. The latter now escapes the monopoly of traditional academic institutions" (Institut Montaigne, 2016; our translation). The term "digital" is largely associated with the notion of sudden and significant changes, with terms such as the "digital revolution", "digital transformation", and "digital disruption" found repeatedly in the textual outputs of institutional actors. The idea that emerges is of an evolution that calls for guidance and supervision, with digital presenting both a challenge and an opportunity.

In curricula, what qualifies as "digital" is primarily seen as one tool among others that students are expected to master in order to develop a set of competencies associated with schoolwork. The "common core of knowledge and skills" (known in France as S4C), which all students are expected to master, stipulates that the use of digital tools should enable students to:

- organise their personal work: digital tools should enable students to engage in writing for the purpose of practicing, revising, and memorising.
- cooperate and carry out projects: digital tools should enable students to organise their work, exchange, and collaborate with their class, school, and institution.
- search for and process information: digital tools should enable students to produce, receive, and disseminate information.
- exchange and communicate: digital tools should enable students to create, publish, and transmit documents.

In other words, depending on the curricula, the objective of digital teaching and learning is to enable students to "use digital technologies more pertinently to conduct research, access information, hierarchise it, and produce content themselves" (MEN, 2015b; our translation). In this view, digital education involves teaching students to use computerised tools and to engage in social information processing. In the curricula, where the focus is computer science, what is being described is often just one of its constituent dimensions (e.g., programming or algorithmics, often referred to as "coding"). The term is seldom used to refer to the technological dimensions or social uses of IT.

As for textbooks, only one of the selected volumes makes an implicit distinction between what pertains to computer science in the strict sense and what relates to digital more generally, with a first section entitled "digital tools" and a second entitled "coding and programming". However, no links or bridges are made between the two sections, except for the use of computers, covered in the first section and allowing for the use of the programming software Scratch in the second section.

Almost all the books in our corpus refer to programming on their cover ("discover coding", "I can code", "become a programmer", "algorithmics and programming booklet", "learning to program", "IT programming for beginners", etc.), illustrating the tendency of textbooks to associate, in line with official (prescriptive) guidelines and the *Socle Commun* (S4C), informatics with the learning of a language, coding, and programming. This observation could lead to further research to question the link between this focus on programming and the representations of students and teachers.

Kafai & Proctor (2021) wonder what it does mean for learners to be computationally-literate in the 21st century". Curricula and textbooks seem to have divergent views on that question. This is a possible difficulty for teachers. Whereas textbooks focus on informatics and coding, the main idea in curricula is not the importance of developing a way of thinking, a computional thinking (Wing, 2006) or even digital humanities. The idea in curricula is of using digital tools in teaching and teaching students how to use them. We decided to test these general conclusions, answering the second research question drawn from our corpus on a specific content: the notion of loop.

Analysis of evolving approaches to the notion of loop

The results presented in regards to the first research question point to changes in the content taught from one cycle to the next. Questions of algorithmics are a central feature of textbooks in cycles 1 and 2 (ages 3–8). The question of use becomes more prominent in cycle 3 (ages 9–11). We hypothesise that what is meant by algorithmics varies by cycle in two ways:

- New notions are introduced (e.g., variable, assignment).
- Notions are further refined or placed more explicitly in a computer science context. For example, the same term (e.g., the notion of loop) can refer to different types of content at different levels and even in different textbooks.

A second analysis was carried out with the aim of characterising how the same notion is formulated and evolves across the different textbooks. The aim is to identify the different modes of textualisation associated with the same notion at different academic levels.

The analysis focused more specifically on the notion of "loop". A loop is a programming structure that repeats the execution of a given sequence of instructions. Beyond the notion itself, what type of content is presented to students in different cycles? What associated notions are considered in the different cycles?

For the purposes of this analysis, we selected four of the ten initial textbooks on the basis that these address the notion of loop in at least one text (in which the relevant knowledge is presented to students) and one exercise (in which students are set a task that applies the knowledge in question). One textbook per cycle was selected.

For each textbook, the chapters introducing the notion of loop were analyzed by adapting the categories of analysis of school subjects proposed by Reuter (2014). We focused on the notions examined, the tasks set for students, and the social practices referred to. Following Martinand (1986), we proceeded on the basis that school content is not only derived from academic knowledge but can also be constructed with reference to production practices, engineering practices, and even day-to-day practices in different social spheres. This explains why we set out to determine the extent to which content referred to academic knowledge or to different social practices.

Several types of loops are examined

First of all, we have to clarify that in primary school, it is not the actual informatics concepts that are taught, but rather a first approach to the notions. Therefore, the first point of our analysis concerns the types of loops with which students are made familiar (table 4).

Computer scientists distinguish between different types of loop: repeating an instruction indefinitely, repeating it a set number of times (a "for" loop), repeating it while an event is true (a "while" loop), or until an event is true ("until"). The different types of loop are not introduced in every textbook. We might expect an increasing number of loops to be presented to older students, the assumption being that students are likely to learn about a limited number of types of loop in cycle 1 before learning about other types at a later stage. However, an analysis of the selected textbooks shows that this is not the case:

- In the cycle 1 textbook, students are introduced to the "for", "while", and "until" loops and the infinite loop, although no distinction is made between them, and no practical illustrations or applications are provided.
- In the cycle 2 textbook, only the infinite loop is presented.
- In the cycle 3 textbook, the "for" and "while" loops and infinite loops are examined. Students learn that loops can be nested within other loops.
- In the cycle 4 textbook, "for" and "while" loops are considered.

The loops are also defined differently in different textbooks (Table 4).

Table 4. *Cross-tabulation: How loops are described in the different textbooks*

Cycle	Cycle 1	Cycle 2	Cycle 3	Cycle 4
"For" loop	Repeat a set number of times	/	Repeat a given number of times	Repeat n times
"While" and "until" loops	Repeat the action until	/	Repeat until	Repeat until
Infinite loop	Infinite loop	Self-loop	Repeat indefinitely	/
Nested loop	/	/	Nested loop	/

Cycle 1 textbook

The book is split into two sections. The first section is presented as a children's storybook in which each notion (including the notion of loop) is introduced in a narrative text without being made explicit. The second part is an "exercise book" focusing on the notions examined in each chapter.

The first part of the book presents a situation to introduce the notion of loop (Figure 4). Ruby must recover a stone on a roof. To do so, she builds a wooden ladder, repeating the same operation over and over again.

Figure 4. *Extract from the book Hello Ruby (Liukas, 2016, p. 35). The illustrations show Ruby building a ladder by assembling rungs. The picture suggests that building a ladder requires taking a rung, tying a knot on the left and then on the right, and starting all over again until the desired height is reached.*

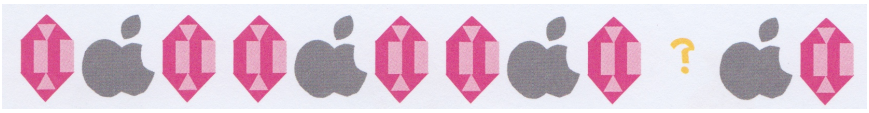
In this narrative, the notion of loop is introduced through reference to an everyday situation that is assumed to be familiar to students by emphasising the repetition of the same procedure.

The second part of the book returns to the narrative and explains the notion in greater detail: "In actual fact, she built one rung of the ladder before repeating the same action five times" (p.87). The aim of this section is to introduce the notions before applying them in exercises. The "for", the "while" and the infinite loops are presented to students through a text describing each loop: "the simplest loops are those that are repeated a given number of times. However, in many cases the action must be repeated until something changes […]" (p. 86).

The book then moves on to exercises involving offline activities. Students must first identify patterns repeated within different visual representations and complete them (Figure 5 [A]). It is worth noting that no visual element (e.g., arrow) is used to indicate repetition.

Students are required to perform an action sequence (Figure 5 [B]) by embodying it themselves. In this exercise, which is similar to the child-robot game (Greff, 1996), students embody a "machine" by receiving instructions and executing them. The aim is for them to perform different loops by repeating a series of actions until a partner issues a signal. This raises the question of the representation that students have of the operation of a "machine" – a fundamental concept in computer science (Dowek, 2012) – since they are not machines.

It should also be noted that the proposed exercises range from a horizontal representation of motifs (Figure 5 [A]) – similar to tasks typically set at preschool level – to a vertical representation of the sequence (Figure 5 [B]), which is closer to representations used in code editors.



A

B

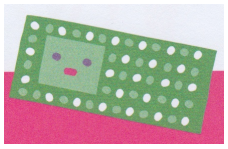
Figure 5. *Representations of loops in a [A] and [B] cycle 1 textbook (Liukas, 2016)*

As well as the elements of content that relate to the notion of loop introduced through stories and applied through exercises, the book promotes values expressed by a character called Léopard, who "likes things to be clear and simple" and who recommends "ignoring any details that make things difficult". His advice points to a recurring theme.

The narrative sections and the exercises refer to situations that are assumed to be familiar to students and appear to make no reference to computers or IT. However, the attentive reader will detect subtle references to IT and digital culture, such as the choice of characters or the graphic representations used as illustrations (see Figure 6).

A

B



C

Figure 6. *Illustrations from Hello Ruby (Liukas, 2016)*
[A] Snow leopard corresponding to the Mac OS X Snow Leopard update[6].
[B] Tapestry motif corresponding to the Apple logo [7].
[C] Illustration corresponding to a motherboard.

Cycle 2 textbook

The cycle 2 textbook is split into seven workshops – themselves divided into eight activities – and four challenges. A humanoid robot operating on the surface of Mars guides the reader throughout the book. Each activity includes a scenario setting out a problem, a "user manual" introducing the key notions examined, and an exercise by way of practical application.

In the activity devoted to loops, the "looping procedures" are introduced as procedures that "enable actions to be performed without stopping" (p.33). In other words, the book examines infinite loops. The graphic representation uses cells displayed horizontally in which symbols are inserted, setting out a set of instructions.

Figure 7. *Representations of loops in a cycle 2 textbook (Croq et al., 2015a)*

Moreover, the loop is approached through the notion of recursion: the last instruction of the algorithm named "procedure 1" (P1) leads to the realisation of procedure 1.

6 https://support.apple.com/kb/DL1860?locale=fr_FR

In this textbook the computer science unplugged approach is adopted. The aim is for students to guide a character (a "robot") across several chequerboards based on a set of instructions to be written down on paper. Rather than being provided to them, the instructions are written by the students themselves, establishing a relationship more akin to programming.

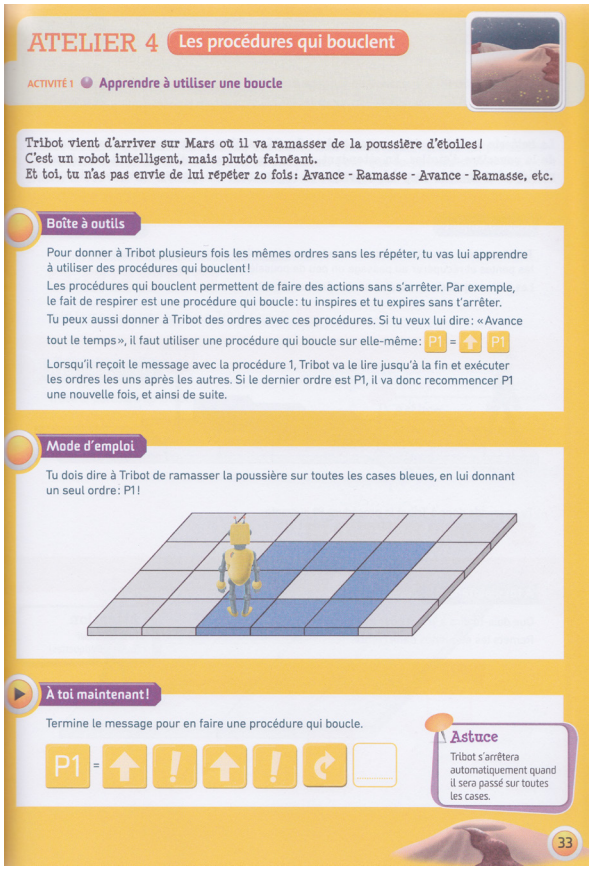


Figure 8. *Extract from the textbook J'apprends à programmer tout seul (Croq & al., 2015, p. 33). The illustrations show Tribot on a chequerboard picking up stardust on the blue squares.*

In this offline activity, explicit reference is made to a digital environment. By analogy with educative robotics, a "robot" character receives instructions given by the students, and moves across a chequerboard (figure 9).

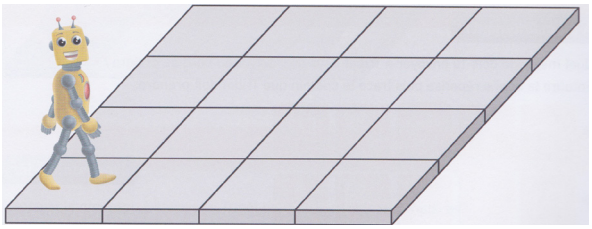


Figure 9. *Representations of a humanoid robot (Croq et al., 2015).*

This offline activity is supplemented by two "workshops" that make explicit reference to social practices in which procedures and loops are used, describing an assembly line in a car factory. This is the only textbook that elucidates the relationships between the learning activities and the social practices.

Figure 10. *Representations of an assembly line in cycle 2 (Croq et al., 2015).*

Cycle 3 textbook

The cycle 3 textbook is divided into three parts. Part one focuses on the use of the block programming language Scratch, part two introduces students to the use of Python, and part three tackles what it terms the "world of IT". Key notions are introduced by learning the Scratch or Python languages. Students learn how to use sprites[7] in Scratch as well as the different functions of blocks, among other things. Notions relating to loops are introduced with the programming language Scratch. They are represented using Scratch instruction blocks, with each block including a series of instructions.

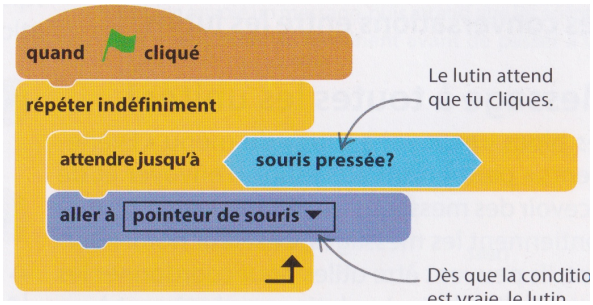


Figure 11. *Representations of loops in a cycle 3 textbook (Vorderman et al., 2017)*

In this book, "for" and "while" loops and infinite loops are presented. Students also learn that loops can be interlinked.

The tasks set are programming projects carried out using Scratch. These take the form of guided exercises in which students are required to follow a pre-defined procedure.

7 A sprite is an object or a character in Scratch.

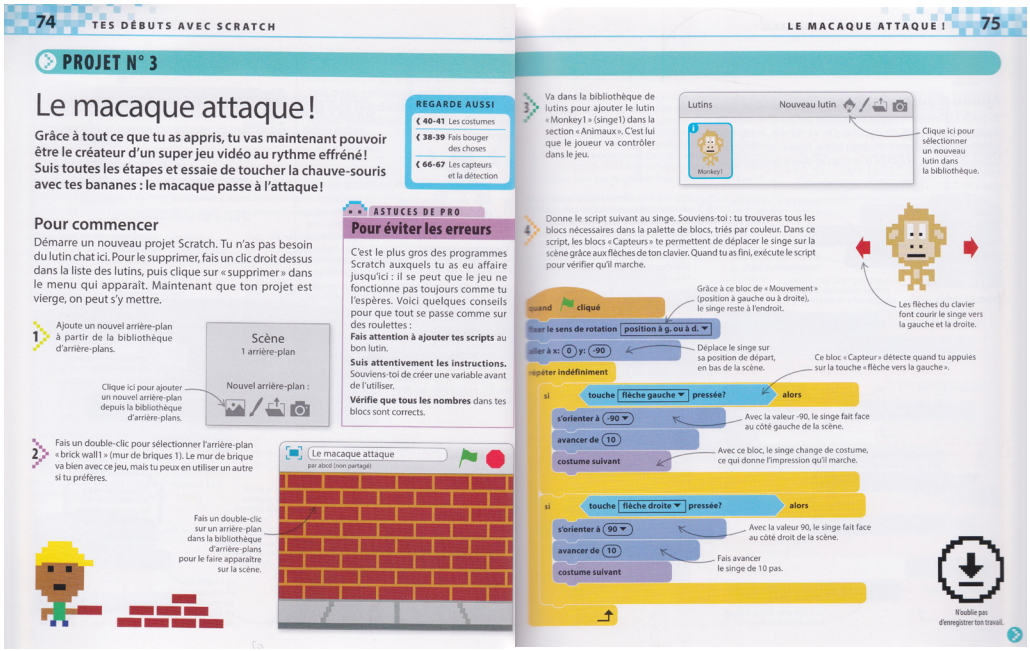


Figure 12. *Example of a "project" exercise page in the textbook À vos marques, prêts ? Codez ! (Vorderman et al., 2017, pp. 68-69). In this case, students are required to create a video game by following the four steps described.*

Cycle 4 textbook

The cycle 4 textbook comprises five sections, including 4 levels and a "complements" section. Each level is subdivided into two activities and five projects. Each activity is subdivided into three sections: an "I discover" section that puts the child in a problem situation, an "I understand" section that consolidates the key notions, and an "I apply" section that provides an opportunity for students to put the notions learned into practice. The level is concluded by a "recap" page that summarises the notions learned and provides an MCQ test.

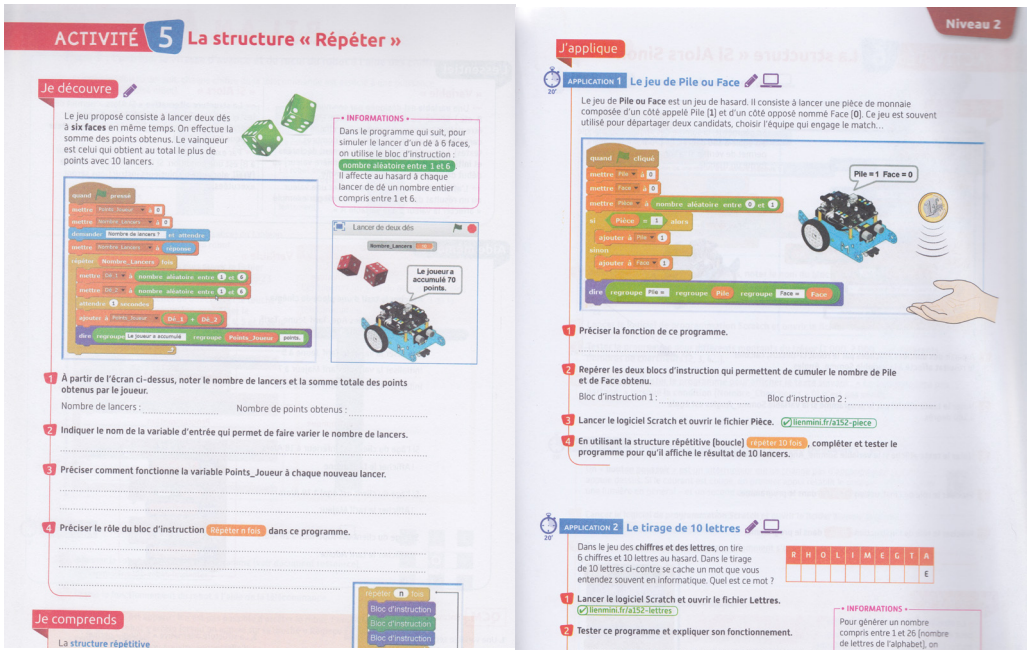


Figure 13. *Example of an activity in Cahier d'algorithmique et de programmation (Anguenot et al., 2016, p. 32-33). In this example, students learn about the notion of loop by programming a heads or tails game.*

As in the cycle 3 textbook, the loop is represented by the algorithm and by the Scratch blocks (figure 13). The "for" and "while" loops, defined as "repeat" (p.45) and "repeat until" (p.59) loops, are examined through various mBot programming exercises.

The aim of these exercises is not to get students to build programs including loops in Scratch so much as to get them to work on mathematical concepts. For example, students are encouraged to perform a series of calculations relating to oil drilling and then to verify their calculations based on the Scratch program using the "repeat until" structure.

During cycle 4 (secondary school), where lessons are delivered by STEM teachers (e.g., mathematics teacher, design and technology teacher, etc.) computer science is approached as an instrument in the service of other disciplines rather than as a discipline to be learned for its own sake.

Discussion on the notion of loop

From the above description of the textbooks, three key points merit discussion: the nature of the tasks set, the increasingly specific vocabulary, and the reference to social practices.

Nature of the tasks set

A point worth considering is the nature of the tasks that students are required to undertake.

In the cycle 1 and cycle 2 textbooks, programming is approached through offline activities similar to activities commonly undertaken in class or activities undertaken in pedagogical robotics. By contrast, in the cycle 3 and cycle 4 textbooks, the learning of programming notions involves programming exercises on computers, usually with Scratch. In these books, this visual programing language provides a basis for both defining, representing and applying the notion of loop.

Increasingly specific vocabulary

The analysis of the vocabulary used in the textbooks shows that it tends to become increasingly specific and connected to other areas of knowledge.

The terminology associated with loops varies in different cycles. During cycle 1 textbook refers to "things that recur" and "procedures that loop". In the cycle 2 textbook, recurring elements are given specific names, i.e., "orders" or "actions" – in other words, terms that are not specific to computer science. In the cycle 3 textbook, the terminology is even more specific. Recurrences are described as "a part of a program". Here, the notion is restricted to the domain of computer programming. In the cycle 4 textbook, the notion is no longer referred to as a "loop" but as a "repetitive structure" or "repeat structure", defined as "allowing for an instruction sequence to be repeated n times". In this cycle, in which Computer Science is delivered by mathematics teachers, the notion of loop is defined in mathematical terms, referring to the concept of recurrence.

In other words, to understand vocabulary choice, one key feature is terminological precision, another is the reference to other knowledge domains (such as mathematics).

It is worth noting that the notion of loop is invariably introduced in each of the textbooks examined by combining a definition of the notion and one or more semiotic representations, with the latter varying from textbook to another. The representations are horizontal in the cycle 1 and cycle 2 textbooks but vertical in the cycle 3 and cycle 4 textbooks.

Reference to social practices

Finally, these analyses highlight the multiplicity of social practices to which textbooks refer in relation to the notion under study. In cycle 1, the notion of loop is introduced with reference to "everyday prac-

tices", using the example of the wooden ladder. This approach reflects the idea that school exercises should be "close to real life", which underpins major international assessments such as the PISA programme (Bart & Fluckiger, 2015).

What are the effects on learning? Computer scientists will likely have no trouble recognising a loop in this situation since they are already familiar with the notion. However, children are very unlikely to recognise a loop since they have yet to acquire the relevant knowledge. Nobody builds a wooden ladder while thinking that they are performing a programming loop. As is often the case (Bart & Fluckiger, 2015), the "real-world" situation presented to students is in fact only an artificial academic construct.

Similarly, when students are engaged in the child-robot game, executing instructions, it is not a machine that executes the instructions but the students themselves. There is a risk of obscuring, for younger children, the concept of "machine", a fundamental concept in computer science (Dowek, 2012), since children are evidently not machines. Our research prompts further investigation of this issue. In textbooks for older children, instructions are executed by the program, involving fictional characters. The "world of IT" in the broad sense involves specific graphic choices. In other words, these textbooks refer to social practices like video games or industrial robotics (Martinand, 1995).

In other words, students are not always in a computer science context when learning about computer science concepts. While it is clear that the notion of loop originates from computer science and, more specifically, from algorithmics, it is striking to find that computer science seems to emerge only gradually as the context in which the notion of loop is articulated and "put into words" for students.

Conclusion

As this literature review has shown, in the absence of consistent teacher training, textbooks and curricula are seen as vital resources for designing classroom activities. Our analysis of curricula and textbooks highlighted several key points. In regard to the first research question (how CS content in textbooks can be described and categorised), the transversal analysis shows that algorithmics is more frequent and technology is minored, as it is in the curricula. Programming and coding is more likely to be taught by mans of texts rather than exercises, which raises questions regarding the need for practicing informatics.

Regarding the second research question (to what extent informatics contents refer to purposes, social activities or the needs of future citizens), the analyses of both the curricula and the textbooks bring some understanding of the wider educational goals of computer science teaching in primary French school; the in-depth qualitative analysis showing that the notion of loop is introduced as a non-digital notion and is gradually related to the digital world.

Regarding the third research question (how the notion of loop evolves across the curriculum), we showed that the notion is gradually defined with more specific vocabulary and that the tasks suggested for the students vary from offline activities to online activities.

The findings of this study suggest that primary school teachers are expected to teach aspects of computer science and, therefore, to oversee computer science learning activities without, however, always being able to clearly identify relevant content and learning objectives (Spach, 2017).

Our results show that the aims and objectives of this content tend not to be clearly set out in the curricula, torn as they seem to be between the internal objectives of the discipline (such as students' cognitive development), academic objectives (such as acquiring skills to use academic computer science in a school context), and extracurricular objectives (such as preparing students for using IT for professional purposes).

In our analysis of both curricula and textbooks, computer science is conceived and presented to students as forming part of a broader and less restrictive field referred to as "digital". The stated objective of computer science education is to introduce students to aspects of scientific and technical culture with a view to equipping them to understand the digital world around them.

However, our findings suggest that this objective can only partially be achieved. The first reason for this is the limited amount of content referring directly to the specific technological dimension, whether in curricula or textbooks. In school, students are only provided with very partial access to the tools needed to understand how diverse the digital world actually is and the different ways in which it operates. The second reason is that whenever computer languages and algorithmics are examined, they tend to be linked more closely to questions and problems in areas other than digital proper. Therefore, it is legitimate to question the extent to which students are likely to understand that the elements of algorithmics to which they are introduced fundamentally underpin the digital tools that they use on a day-to-day basis.

Biographical notes

Isabelle Vandevelde
is a PhD student of Educational Sciences at the the CIREL laboratory (EA 4354) of the University of Lille. Her research focuses on children's digital culture and computer science learning.
Address: Théodile-CIREL, University of Lille, France. Email: isabelle.vandevelde@univ-lille.fr

Cédric Fluckiger
is a professor in educational sciences at the University of Lille and a member of the CIREL laboratory (EA 4354) at the University of Lille. After a PhD (2007) on the appropriation of digital technologies by middle school students, he has recently worked on computer contents in primary and secondary schools, and on the uses of digital technologies in an educational context, by students. He is currently director of the Théodile-CIREL team.
Address: Théodile-CIREL, University of Lille, France. Email: cedric.fluckiger@univ-lille.fr

Sandra Nogry
is Associate Professor in Educational Psychology. Her research topics include technology use in education and computer science education at primary school. She studies learning processes in a situated perspective.
Address: Paragraphe Lab, Cergy-Paris University, Paris, France. Email: sandra.nogry@u-cergy.fr

References

Amadieu, F., & Tricot, A. (2014). *Apprendre avec le numérique. Mythes et réalités*. Retz.

Baron, G.-L., & Boulc'h, L. (2012). Les technologies de l'information et de la communication à l'école primaire. État de question en 2011. *EpiNet*.

Baron, G.-L., & Drot-Delange, B. (2016). L'informatique comme objet d'enseignement à l'école primaire française? Mise en perspective historique. *Revue Française de Pédagogie, 195*. http:// rfp.revues.org/5032

Bart, D., & Fluckiger, C. (2015). *Évaluation, fabrication des contenus et disciplines d'enseignement*. In B. Daunay, C. Fluckiger, & R. Hassan, Les Contenus d'enseignement et d'apprentissage. Approches didactiques (p. 91–102). Presses Universitaires de Bordeaux.

Brennan, K., & Resnick, M. (2012). *New Frameworks for Studying and Assessing the Development of Computational Thinking*. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vol. 1, Vancouver, 13–17 April 2012, 25 p. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Bruillard, É. (2009). *Place de l'informatique dans l'enseignement secondaire, réflexions introductives*. In G.-L. Baron, É. Bruillard, & L.-O. Pochon, Informatique et progiciels en éducation et en formation (p. 21–27). INRP.

Daunay, B., Fluckiger, C., & Rouba, H. (2015). *Les contenus d'enseignement et d'apprentissage. Approches didactiques*. Presses Universitaires de Bordeaux.

Delcambre, I. (2007). *Contenus d'enseignement et d'apprentissage*. In Y. Reuter, C. Cohen-Azria, B. Daunay, I. Delcambre, & D. Lahanier-Reuter, Dictionnaire des concepts fondamentaux des didactiques (p. 43–48). De Boeck Supérieur.

Dowek, G. (2012). Les quatre concepts de l'informatique. In G.-L. Baron, É. Bruillard, & V. Komis, *Actes du quatrième colloque international DIDAPRO 4—Dida&Stic* (p. 21–29). New Technologies Editions.

Drot-Delange, B. (2013). Enseigner l'informatique débranchée: Analyse didactique d'activités. *Actes du Congrés de la Recherche en Education et Formation (AREF - AECSE), 1* 13.

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for Computer Science in the School Curriculum. *Educational Technology & Society, 19*(3), 38–46.

Fluckiger, C. (2019a). *Une approche didactique de l'informatique scolaire*. Presses Universitaires de Rennes.

Fluckiger, C. (2019b). Numérique en formation: Des mythes aux approches critiques. *Education permanente, 219,* 17–30.

Greff, É. (1998). Le « jeu de l'enfant-robot »: une démarche et une réflexion en vue du développement de la pensée algorithmique chez les très jeunes enfants. *Revue Sciences et Techniques Éducatives, 5*(1), 47–61.

Hong, T. N. T., Tat, K. Q., & Bruillard, É. (2010). Informatique et partage de ressources au Vietnam. *Revue de l'EPI* (Enseignement Public et Informatique), (125).

Joris N., Henry J. (2014). L'enseignement de l'informatique en Belgique francophone: état des lieux. *Bulletin de la société informatique de France, 2*, 107–126.

Joshua, S., & Dupin, J.-J. (1993). *Introduction à la didactique des sciences et des mathématiques*. Presses Universitaires de France.

Kafai, Y. B., & Proctor, C. (2021). A Revaluation of Computational Thinking in K–12 Education: Moving Toward Computational Literacies. *Educational Researcher,* 0013189X211057904

Lin, J.M.C., Lin, K.Y. & Wu, C.C. (1999). A Content Analysis of Programming Examples in High School Computer Textbooks in Taiwan. *Journal of Computers in Mathematics and Science Teaching, 18(3),* 225–244. https://www.learntechlib.org/primary/p/8829/.

Livingstone, S. (2012). Critical reflections on the benefits of ICT in education. *Oxford review of education, 1*(38), 9–24.

Martinand, J.-L. (1986). *Connaître et transformer la matière*. Berne, Peter Lang

Martinand, J.-L. (1995). The purposes and methods of technological education on the threshold of the twenty-first century. *Prospects, 25,* 49–56.

Means, H. W. (1987). A content analysis of six introduction to computer science textbooks. *ACM SIGCSE Bullein, 19(1)*, 403–413. https://doi.org/10.1145/31726.31796

Mirabail, M. (1990). La culture informatique. *ASTER, 11,* 11 28.

Otero, M. R., & Baron, G. L. (2010). Informatique et TIC en Argentine. Éléments d'analyse et de comparaison avec la France. *Revue de l'EPI (Enseignement Public et Informatique),* (123)

Papadakis, S. (2018). Gender sterotypes in Greek computer science school textbooks. *International Journal of Teaching and Case Studies, 9*(1), 48–71.

Reuter, Y. (2004). Analyser la discipline : Quelques propositions. In *Actes du 9e colloque de l'AIRDF*.

Reuter, Y. (2007). Discipline scolaire. In Y. Reuter, C. Cohen-Azria, B. Daunay, I. Delcambre, & D. Lahanier-reuter, *Dictionnaire des concepts fondamentaux des didactiques* (p. 85–89).

Reuter, Y. (2014). Construire la catégorie de discipline scolaire en didactique(s). *Linguarum Arena, 5,* 79–95.

Slot, M. F., Lorentzen, R. F., & Hansen, T. I. (2021). Hvordan integreres teknologiforståelse i dansk?. *Learning Tech, 6*(10), 21–46. https://doi.org/10.7146/lt.v6i10.122751

Spach, M. (2017). *Activités robotiques à l'école primaire et apprentissage de concepts informatiques. Quelle place du scénario pédagogique?* Sorbonne Paris Cité.

Vandevelde, I. (2019). *Culture numérique et apprentissages scolaires de l'informatique* [Journées jeunes chercheurs du Gis2IF].

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

Corpus of textbooks and other books used as textbooks

(*books selected for the analysis of the notion of loop)

Cycle 1:

* Liukas, L. (2016). *Hello Ruby!* Glénat jeunesse.

Cycle 2:

* Croq, A., Farnet, P., Payet, N., & Trannoy, G. (2015a). *J'apprends à programmer tout seul!* Bordas.

Croq, A., Farnet, P., Payet, N., & Trannoy, G. (2015b). *Mon cahier pour apprendre à programmer.* Bordas.

Lyons, H., & Tweedale, E. (2017). *Mon atelier code. Tout pour faire ses premiers pas en programmation informatique.* Fleurus.

Cycle 3:

Cohen, A., & Marcialis, J. (2018). *Comprendre les outils numériques et programmer.* Hatier.

Morgan, N. (2017). *Javascript pour les kids.* Eyrolles.

* Vorderman, C., Woodcock, J., & McManus, S. (2017). *À vos marques, prêts ? Codez !* Larousse.

Cycle 4:

* Anguenot, G., Launay, J., Corne, R., Vogt, O., & Sauzeau, D. (2016). *Cahier d'algorithmique et de programmation.* Delagrave.

Bassette, T. (2017). *1, 2, 3, je code avec Scratch.* Larousse.

Plumel, D. (2017). *1, 2, 3, je construis avec Minecraft.* Larousse.

Corpus of curricula

MEN. (2015a). *Programme d'enseignement de l'école maternelle.*

MEN. (2015b). *Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4).*

MEN. (2015c). *Socle commun de connaissances, de compétences et de culture : Décret n°2015-372 du 31-3-201*