A Web Framework to Develop Computational Thinking through Music Coding

Adriano Baratè[1], Luca Andrea Ludovico[1], and Giuseppina Rita Mangione[2]

[1] LIM - Laboratorio di Informatica Musicale
Dipartimento di Informatica
Università degli Studi di Milano
{adriano.barate,luca.ludovico}@unimi.it
[2] INDIRE - Istituto Nazionale di Documentazione, Innovazione e Ricerca Educativa

g.mangione@indire.it

Abstract. Music coding is a new discipline aiming to develop a computational way of thinking through music experience even in musically-untrained subjects. In this work we will show how music can represent both a valid learning tool and an engaging reinforcement technique to approach coding at an early stage of children education, i.e. in the primary school. This work starts from the discussion of the educative guidelines that make music coding suitable to the current context. In fact, music coding can foster analytical processes, but it is able to encourage artistic expression, creativity and collaborative learning, too. Educational goals are reinforced by peculiarities such as a prompt feedback of user actions in terms of music performance. After analyzing different types of scaffolding to support computational thinking processes, a publicly available Web framework for music coding will be presented. Finally, the paper will discuss some applications to enhance analytical skills and support creative processes.

Keywords. Coding, Computational Thinking, Education, Music, Web

1 Introduction

School is traditionally considered as the depositary of literacy processes [1]. Nowadays new pedagogical challenges are emerging, and consequently new skills have to be transmitted by educators. Current research is focusing on those psycho-pedagogic practices that emphasize the potential of new technologies and their use to motivate children through active learning. Reference [2] states the need to define new and engaging learning experiences for children – defined as *tynker* – with particular reference to job opportunities related to information and computing. Many scientific papers that propose the introduction of coding recall the need to develop computational thinking since primary school [3–6].

During a coding activity, students are exposed to *computational thinking* [7], namely a form of reasoning oriented to problem solving which involves *abstraction, debugging, remixing* and *iteration* processes [8–10]. Computational thinking is in line with

many aspects of 21st century skills, such as creativity, critical thinking and problem solving [11,12].

Coding is strictly related to tinkering abilities. According to Resnick, the tinkering approach is characterized by a playful, experimental, iterative style of engagement, in which makers are continually reassessing their goals, exploring new paths, and imagining new possibilities [13]. The challenge is that young students – when they approach the basic concepts and the cognitive strategies related to computer science – can develop logical-cognitive abilities, with positive future effects both in terms of meta-knowledge (self-regulation, peer-seeking, problem posing and solving, etc.) and in terms of digital skills obtained through *playful learning processes* [14]. A detailed discussion of the educational pros and cons of coding for young students is beyond the scope of this work; for further details please refer to [15].

The current interest in coding has an impact not only on the exploitation of laboratory didactics, but also on the revision of school curricula [16]. This has been recently demonstrated by the education reform plans in Italy, the UK and the USA that introduced the *Hour of Code* [17,18]. The 2014 report of the European Schoolnet entitled "Computing our future: Computer programming and coding. Priorities, school curricula and initiatives across Europe" has recently described the situation of 20 European Countries, showing that in 13 of them coding has been already proposed in the curricula of primary and secondary school, and in 7 of them it is a compulsory subject (e.g., in Greece and Estonia).

Numerous programming environments for children have been released in order to support the development of three specific dimensions of computational thinking: *concepts*, *practices*, and *perspectives* [19,20]. Such frameworks are called *Initial Learning Environments* [16], and they can be grouped into categories that include so-called novice programming environments, games and challenges, game building environments, and online learn-to-program courses.

Recent studies extend the influence of coding abilities outside the traditional fields of informatics and scientific subjects. For example, Bundy argues that the basic concepts related to coding can be easily integrated into the artistic language, thus facilitating the processes of rule analysis and the observation of recurrent linguistic structures [21].

Starting from the mentioned researches and opinions, in our previous works we considered music as an alternative language to stimulate computational thinking through playful, practical and collaborative activities. For an effective acquisition of competence, intended here as the ability to act in – or react to – a given situation within a given context in order to achieve a performance [22], it is necessary to conceive educational proposals able to integrate music analysis and music production. Our approach to *music coding* – an evolution of "traditional" coding defined in [23] – is based both on music performance and on music processes analysis. After analyzing *problem-solving learning environments* (PSLEs) and adopting the evidence-based approach suggested in [24], we propose a music coding environment which combines artistic creativity and analytical skills. In this way, children are stimulated to work on the cognitive aspects of computational practice and perspectives [19].

This paper is organized as follows: next section will discuss the pedagogical value of

music in children education; Section 3 will present a web framework designed to encourage music coding through a playful approach, discussing its goals, gameplay, graphical interface and technological issues; finally, Section 4 will illustrate a number of possible applications to enhance analytical skills and support creative processes.

2 The Pedagogical Value of Music

Providing an in-depth discussion of the pedagogical value of music would go beyond the goals of the present work. In this section we will just list the position expressed in recent works by some experts in the field of pedagogy and education.

First, experts agree that music is able to influence the construction of the child's personality since it promotes the integration of perceptual, motor, affective, social and cognitive dimensions [25] by relating the basic aspects of human life (e.g., physiological, emotional and mental spheres) with the basic elements of music (e.g., rhythm, melody and harmony).

The abilities of listening, exploration and analysis are fundamental for the development of general meta-cognitive skills, such as attention, concentration, control. For example, through music young students can develop the aspects of analysis and synthesis, problematization, argumentation, evaluation and application of rules [26–29]. In addition, as it regards the ability to read and understand, children have the possibility to train their transcoding skills by moving from the musical domain to the verbal language to describe what they heard [30].

In the digital era, new technologies and computer-based approaches can influence music learning and teaching processes. A recent and comprehensive review of this subject can be found in [31], a work that discusses a range of innovative practices in order to highlight the changing nature of schooling and the transformation of music education. Many researchers, experts and music teachers feel a pressing need to provide new ways of thinking about the application of music and technology in schools. It is necessary to explore teaching strategies and approaches able to stimulate different forms of musical experience, meaningful engagement, creativity, and teacher-learner interactions.

The idea of this work is applying the most recent pedagogical theories about coding and music teaching in primary school through a playful approach to music composition, conceived for musically untrained children and designed to encourage the computational way of thinking.

A comparison between coding environments and music-oriented programming frameworks unveils similarities and differences. According to [24], coding environments for children should foster the development of three dimensions of computational thinking: computational concepts, practices, and perspectives. *Computational concepts* are the conceptual entities that programmers use, such as variables and loops, in order to solve a problem algorithmically. *Computational practices* are problem-solving practices that occur during the process of programming (e.g., iteration, reuse and remix, abstraction, and modularization). Finally, *computational perspectives* refer to students' understanding of themselves, their relationship to others, and the world around them. Only the first item is usually well covered by general-purpose learning tools and environments for children, whereas a suitable music-coding envi-

ronment can fulfill also the other goals. Computational concepts will be discussed in depth in the following section. As it regards computational practices, a music-coding environment allows the exploration of music contents through abstraction processes, as demonstrated by the final use cases. Besides, such an approach can encourage reusing and remixing of music materials, for instance through iterative operators. Finally, as it regards computational perspectives, when students play together a music performance becomes a social activity. A cooperative performance extended to a class not only provides children with self-consciousness, but fosters relationships towards other children and the surrounding environment. Moreover, a music coding environment can implement features recalling social-game mechanisms to allow peer seeking (searching for the helping hand of a friend) and peer reviewing (asking for other students' comments). Consequently, in our opinion a music coding framework may provide not only music skills by fostering creativity, but even teach computational thinking better than other "traditional" coding environments.

3 A Web Framework for Music Coding

In order to demonstrate the possibilities offered by music coding in primary school education, we have designed, developed and released a web-based framework presenting a gamification approach. This project is multi-platform and freely available, and it is an evolution of the prototype proposed in [23] obtained through an initial experimentation phase.
The goal of this section is to describe the different aspects of the framework, both from the developer's and from the end-user's point of view.

Music Operators

In order to support a formal and algorithmic approach to music composition and analysis, we have identified a limited set of music operators. Recalling the goals of this initiative, the application domain has been intentionally simplified:

- In our framework a complex score is made of multiple melodic tunes, one per staff, presenting no chords. Harmony can be produced by putting simultaneous notes on different staves;
- Every note is quantized according to the smallest rhythmical value allowed. Longer values can be obtained by tying quanta together. This approach is valid for children's songs and simple tunes, where a coarse quantization is sufficient, but it would fail with scores containing tuplets or other complex rhythmic layouts. In any case, the interface lets the user set the metronomic value, consequently the time duration of quanta can be very small;
- Supported operators are intuitive to use and understand. In fact our goal is not to cover all possible music processes, but to provide untrained children with an intuitive tool set to build basic music performances.

The operators we have identified can be roughly classified into two categories: *Melodic Operators* (*MOs*), i.e. the operators that set or alter note pitch, and *Rhythmic Operators* (*ROs*), i.e. the operators related to rhythmic aspects of notes. All operators are applied to a given quantum, and their effect is limited to this scope.
In our framework *MOs* include:

- *Set(p)* – This operator sets the current pitch to *p* and starts playing the note. The execution is stopped at the end of the current quantum, unless a *Tie()* operator is invoked in the next quantum;
- *Transpose(v)* – This operator modifies the previous pitch according to a number of ascending or descending steps expressed by signed integer *v* and start playing the note. As explained below, in our implementation we deal both with scale grades of a diatonic scale and with halftones of a chromatic scale;
- *Unset()* – This operator unsets previous pitch information. Since from a logical point of view a score quantum cannot be empty, this operator corresponds to the concept of rest. In a certain sense, *Unset()* belongs both to *MOs* (since it unsets the pitch) and to *ROs* (since it ends the sequence of quanta of the previous note event, thus determining its aggregated rhythmic value).

In our framework *ROs* include:

- *Tie()* – This operator extends the duration of the last pitch (or rest) to the current quantum;
- *Unset()* – As mentioned above, this operator means absence of sound. It is the default when a quantum has no operator associated.

In order to foster computational thinking, other operators which do not belong to the mentioned classes have been implemented. For instance, *Repeat(m,n)* makes the *m* previous steps be repeated for *n* times. In music notation this corresponds to symbols such as bar-repeat signs, repeat barlines, text indications such as “Da capo”, etc.
Within the interface each music operator has been assigned to a graphical representation, as shown in Figure 1.

Fig. 1. Graphical representation of music operators:
Set(C3), *Transpose(+1)*, *Transpose(-1)*, *Repeat(1,1)*, *Tie()*, and *Unset()*.

Gameplay

The educational activity is composed by two different steps: *score coding* and *interactive listening*.
During the first phase, score is produced by placing a number of cards (which implies the invocation of music operators) on the game board. The set of available cards is contextualized to the current music instrument: an unpitched instrument will enable only rhythm-related operators, consequently it could constitute a good entry point for musically untrained children. This phase aims to produce a traditional music score in either a traditional or a non-traditional way, namely by using either the declarative *Set(p)* operator or other operators respectively. Needless to say, the latter option is closer to the goals of music coding, as demonstrated by the complete example shown in Section 4. Score composition can be performed collaboratively by many children together, for instance by assigning an instrument or a quantized step to each student, like in turn-based games.

The second phase, called interactive listening, occurs when the *Play* button is pushed, thus running a timed playback, or the *Back*/*Forward* buttons are activated, thus performing only the music events of the previous/next step. The latter possibility somehow recalls code debugging, where actions can be parsed one by one to see if the desired result is being achieved. At any moment music performance can be paused, stopped, and rewound. This phase is "interactive" for a number of reasons. First, music performance can be influenced by adjusting and rearranging music operators: modifying the score brings back to the first phase, in accordance with a spiral model that converges to the desired result [32]. Besides, children can experience the results of coding not only in a passive way, namely by listening, but they can also interact by playing real music instruments. Please note that this operation can be performed not only through traditional instruments: thanks to MIDI support, ad hoc electronic music instruments can be used inside the web environment as input controllers for coding activities.

Interface

The interface has been designed as a set of panels containing different tools, as shown in Figure 2. The upper panel lets the user choose a music instrument to play. Instrumental sounds have been selected among General MIDI patches, in order to be widely supported by software and hardware in use. Both pitched instruments (e.g., cello, clarinet, etc.) and unpitched instruments (e.g., cymbals, drums, etc.) are available.
After the selection of an instrument, the panel of music operators is enabled and filled with the cards corresponding to contextualized music operators, i.e. rhythm-related operators for unpitched instruments and the full set for pitched ones.
Another area contains standard media-player controls, such as *Play* and *Stop* buttons, and a BPM (Beat per Minute) selector. This panel hosts two additional buttons to advance one step forward and back, thus supporting an asynchronous exploration of music code; these buttons are disabled during the playback phase. Finally, the *Diatonic/Chromatic* button is a mode selector that makes music operators work in a diatonic or chromatic context. For instance, the operator *Transpose(n)* – i.e. "transpose the previous pitch *n* steps up" – implies the transposition of *n* scale grades up when in diatonic mode, and the transposition of *n* halftones up when in chromatic mode. Switching the current mode can deeply modify the contour of a piece, above all when pitches are calculated through sequences of *MOs* rather than fixed by *Set(p)*.
Please note that step-by-step retrograde performance and diatonic/chromatic changes can be very useful didactic tools to teach and test the meaning of music operators.
The main area of the screen is a sort of game table. Its purpose is to contain cards that are graphical representations of the music operators chosen by the user. The game table is dynamically dimensioned as it regards both its horizontal axis (i.e., the score length) and its vertical one (i.e., the number of staves).

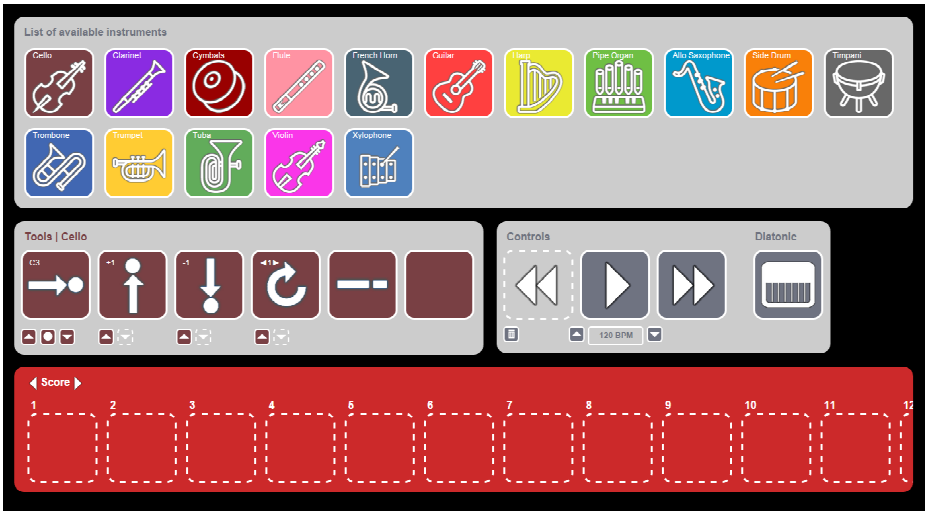

Fig. 2. The game interface.

An automatic and real-time transcription of the current score into common Western notation could be easily performed, and it would definitely be a valid tool to teach "traditional" music notation, but the interface area on most devices is limited, so the presence of another score panel – potentially large – would be confusing.

The framework can be freely accessed at the following URL: http://www.lim.di.unimi.it/coding/player.php. As specified below, it requires a browser compliant with Web MIDI API specifications and a MIDI output device for sound synthesis.

Technological Issues

The languages and formats adopted in the prototype include HTML5, CSS, PHP and JavaScript, in compliance with the World Wide Web Consortium (W3C) recommendations. Consequently this framework is a cross-platform multimedia environment, freely available on browser-equipped and network-attached devices.

A novel aspect is the adoption of Web MIDI API to produce sound and to interface this framework with input/output MIDI-compatible devices. When cards over the table game are parsed, standard MIDI messages are generated and sent to MIDI output for processing.

Web MIDI API – officially described in a W3C Working Draft [33] – currently is still under development. This specification defines an API supporting the MIDI protocol, enabling web applications to enumerate and select MIDI input and output devices on the client system and send/receive MIDI messages. It is intended to enable music as well as non-music MIDI applications by providing low-level access to the MIDI devices available on the users' systems.

We chose to adopt Web MIDI API for the following reasons:

- Provided that either a hardware or a software MIDI-capable synthesizer is available, the user can choose any pitch, instrumental sound and effect supported by

General MIDI (GM) specifications. In other case, any sound (i.e. any pitch for any instrument) had to be associated to a file storing the corresponding audio bitstream;
- Web MIDI API lets us connect to the framework any input and output MIDI-compatible device, thus extending its functionalities. For instance, different controllers such as MIDI keyboards and drums can be used as input devices. Similarly, outputs can be redirected not only to synths, but also to MIDI lighting controllers or other hardware suitable to reinforce music feedback;
- The hardware MIDI chain necessary to produce sound can be substituted by low-cost software devices, e.g. virtual synths available for free. Moreover, by equipping a software synth with sound fonts, the quality of resulting sounds can be very high;
- Even if they are not a W3C standard at the moment of writing, these specifications will be probably supported by all browsers in the near future, due to the interest of the scientific and technical community towards web-based MIDI applications [34]. Currently Web MIDI API is fully implemented only in Google Chrome.

4 Use Cases: Analytical vs. Creative Process

Music coding is a way to foster computational thinking in young students. Usually the idea of computational thinking is related to the ability of solving problems in an algorithmic way. On the other hand, an activity based on music composition and performance within a playful environment seems to be far from the mentioned meaning. How can we combine these apparently conflicting approaches? The answer is in the different ways we can use the framework presented in Section 3.

Let us start from the most straightforward use, namely the instinctive and unmediated creation of a music pattern by untrained students. Note pitches can be set in two ways:

1. A declarative one – e.g., *Set(C3)* – namely by explicitly stating the note to be produced. Please note that quantization forces rhythmic aspects to be treated in a declarative way, nevertheless the operator *Repeat(m,n)* is a way to soften this imposition as it regards rhythm;
2. A relative one – e.g., *Transpose(+2)* – namely by applying an algorithmic modification with respect to the previous value.

After the phase known as *score coding*, *interactive listening* can occur. At this point, audio feedback makes the meaning of music operators emerge. According to the spiral model mentioned above, the educator can invite students to make changes to the score, for instance by setting new pitches or varying operator parameters. Another educational experience is advancing step by step, or changing from chromatic to diatonic mode and vice versa. In other words, the initial unmediated process produces results to be analyzed and improved a number of times.

A completely different approach supported by this framework starts from the analysis of a known piece instrumentally performed, whistled, sung or written in common Western notation. The goal is to use game cards (i.e. music operators) in order to reconstruct the original tune. Once again, the spiral model can be used to iteratively improve the solution.

Finally, let us underline that even this simple set of operators allow different encodings of the same music tune, as shown in Figure 3. This image compares a declarative approach, where each note pitch and duration are explicitly identified, to an extensive use of relative operators. Of course, a great number of "mixed" alternatives could be identified, too. A comparison among heterogeneous strategies can be another educational activity.

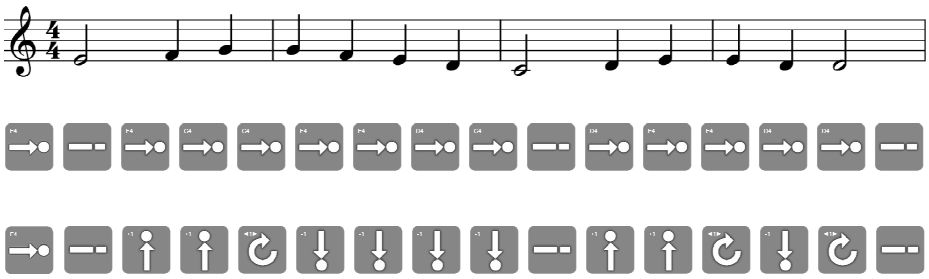

Fig. 3. An excerpt from the 4th movement of the Symphony No. 9 in D minor, Op. 125 "the Choral" by Ludwig van Beethoven. The main theme – proposed in common Western notation (line 1) – has been reconstructed using both a declarative approach (line 2) and a relative approach (line 3).

5 Conclusion

In this work we have underlined the importance that music education can have in developing computational thinking at an early age. Music can provide a playful and creative background to improve analytical mind processes thanks to reinforcement techniques based on sounds. This effect can be amplified through the adoption of gamification, thanks to the association of vivid colors and characteristic shapes to music instruments and operators. Music coding activities can occur in a collaborative context such as a classroom, where both musically trained and untrained students can take part into a unique process under the guidance of an educator. Everyone can contribute by placing cards on the game table during score coding or by interacting with sound through traditional as well as electronic instruments during listening phase.

This proposal is an evolution of former works and stems from the analysis of the problematic aspects detected in previous releases. For instance, during an early experimentation phase we discovered that reinforcement techniques based on visual representations of playing instruments were not effective, since they distracted young users from coding activities.

The current prototype can be improved under many points of view, ranging from interface customizations for portable devices to the introduction of new coding constructs such as conditional structures.

Besides, we are planning a new experimentation phase in Italian primary schools in order to test the effectiveness of our approach.

References

[1] G. Tompkins, R. Campbell, D. Green, and C. Smith, *Literacy for the 21st century*, Pearson Australia, 2014.

[2] J. O'Dell, "Why your 8-year-old should be coding", http://venturebeat.com/2013/04/12/why-your-8-year-old-should-be-coding, April 2013.

[3] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Computers in Human Behavior*, 52, pp. 200-210, 2015.

[4] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and J. T. Korb, "Computational thinking in elementary and secondary teacher education," *ACM Transactions on Computing Education (TOCE)*, 14(1), 5, 2014.

[5] B. Sabitzer, P. K. Antonitsch, and S. Pasterk, "Informatics concepts for primary education: preparing children for computational thinking," in *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, ACM, pp. 108-111, 2014.

[6] N. Smith, C. Sutcliffe, and L. Sandvik, "Code club: Bringing programming to UK primary schools through Scratch," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ACM, pp. 517-522, 2014.

[7] J. M. Wing, "Computational thinking," *Communications of the ACM*, 49(3), pp. 3-35, 2006.

[8] K. Brennan, and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Annual American Educational Research Association meeting*, Vancouver, BC, Canada, 2012.

[9] A. Ioannidou, V. Bennett, A. Repenning, K. H. Koh, and A. Basawapatna, "Computational thinking pattern," in *Annual American Educational Research Association meeting*, New Orleans, Louisiana, United States, 2011.

[10] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), pp. 3717-3725, 2008.

[11] K. Ananiadou, and M. Claro, "21st Century skills and competences for new millennium learners in OECD Countries," *OECD Education Working Papers*, 41, 2009.

[12] M. Binkley, O. Erstad, J. Herman, S. Raizen, M. Ripley, M. Miller-Ricci, et al., "Defining twenty-first century skills," in P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and teaching of 21st century skills*, Netherlands: Springer. Resnick & Rosenbaum, pp. 17-66, 2012.

[13] M. Resnick, and E. Rosenbaum, "Designing for tinkerability," *Design, make, play: Growing the next generation of STEM innovators*, pp. 163-181, 2013.

[14] A. S. Lillard, "Playful Learning and Montessori Education," *American journal of play*, 5(2), pp. 157-186, 2013.

[15] L. A. Ludovico, and G. R. Mangione, "Music Coding in Primary School," in *Smart Education and Smart e-Learning (Smart Innovation, Systems and Technologies)*, Springer International Publishing, pp. 449-458, 2015.

[16] C. Duncan, T. Bell, and S. Tanimoto, "Should your 8-year-old learn coding?," in *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, ACM, pp. 60-69, 2014.

[17] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, "Learning computer science concepts with Scratch," *Computer Science Education*, 23(3), pp. 239-264, 2013.
[18] M. Armoni, and J. Gal-Ezer, "Early computing education: Why? What? When? Who?," *ACM Inroads*, 5(4), pp. 54-59, 2014.
[19] S. Grover, and R. Pea, "Computational thinking in K-12: A review of the state of the field," *Educational Researcher*, 42(1), pp. 38-43, 2013.
[20] Y. Kafai, and Q. Burke, "Computer programming goes back to school," *Phi Delta Kappan*, 95(1), pp. 61-65, 2013.
[21] A. Bundy, "Computational thinking is pervasive," *Journal of Scientific and Practical Computing*, 1(2), pp. 67-69, 2007.
[22] G. Le Boterf, *De la compétence, essai sur un attracteur étrange*, Les Editions d'organisation, Paris, 1994
[23] A. Baratè, L. A. Ludovico, G. R. Mangione, and A. Rosa, "Playing music, playing with music. A proposal for music coding in primary school," in *Proceedings of the International Conference E-Learning 2015, Las Palmas De Gran Canaria, Spain July 21 - 24, 2015*, pp. 3-10, 2015.
[24] S. Y. Lye, and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for K-12?," *Computers in Human Behavior*, 41, pp. 51-61, 2014.
[25] E. Willems, *Las bases psicológicas de la educación musical*, Editorial Paidós, 2011.
[26] R. Berkley, "Teaching composing as creative problem solving: conceptualising composing pedagogy," *British Journal of Music Education*, 21(3), pp. 239-263, 2004.
[27] M. Kaschub, and J. Smith, *Minds on music: Composition for creative and critical thinking*, R&L Education, 2009.
[28] P. Burnard, and B. A. Younker, "Problem-solving and creativity: Insights from students' individual composing pathways," *International Journal of Music Edu-cation*, 22(1), pp. 59-76, 2004.
[29] A. E. Major, and M. Cottle, "Learning and teaching through talk: Music composing in the classroom with children aged six to seven years," *British Journal of Music Education*, 27(03), pp. 289-304, 2010.
[30] D. Branca, "L'importanza dell'educazione musicale: risvolti pedagogici del fare bene musica insieme. Studi sulla formazione," 15(1), pp. 85-102, 2012.
[31] J. Finney, and P. Burnard, *Music education with digital technology*, Bloomsbury Publishing, 2010.
[32] B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, 11(4), pp. 14-24, 1986.
[33] World Wide Web Consortium (W3C), *Web MIDI API*, W3C Working Draft, http://www.w3.org/TR/webmidi, 17 March 2015.
[34] *Proceedings of the 1st Web Audio Conference (WAC 2015)*, CEUR, to be published.