AUTHOR	Samuli Laato, Sampsa Rauti, Erkki Sutinen
TITLE	The Role of Music in 21st Century Education-Comparing Programming and Music Composing
YEAR	2020
DOI	10.1109/ICALT49669.2020.00088
VERSION	Final draft

The Role of Music in 21st Century Education - Comparing Programming and Music Composing

Samuli Laato
Dept. of Future Technologies and Dept. of Education
University of Turku
Turku, Finland
sadala@utu.fi

Sampsa Rauti
Dept. of Future Technologies
University of Turku
Turku, Finland
sjprau@utu.fi

Erkki Sutinen
Dept. of Future Technologies
University of Turku
Turku, Finland
erkki.sutinen@utu.fi

***Abstract*—21st century skills are being added onto K-12 educational curricula globally, often via integrating them into existing subjects such as math. Simultaneously music teaching in K-12 education is losing relevance and popularity. Yet, music theory contains logical structures which are in many regards similar to program code. Additionally the digitization of music production requires composers to effectively use digital music production tools and associated technology. We investigate the opportunities technology-assisted music composing offers for teaching 21st skills and programming in K-12 education through expert interviews with professional music composers (n=4) and programmers (n=5). Analysis of the similarities and differences in the thought processes between creating software and composing music revealed the latter to have potential for teaching the following thinking skills present in K-12 educational curricula: modularity, loops and conditionals, data structures, input/output and software design. Additionally implicit learning benefits on increasing technical know-how, cooperative skills and design thinking were discovered.**

***Index Terms*—21st century skills, music composing, programming, K-12 education, expert interview**

I. Introduction

Population-level statistical analyses have demonstrated participation in school music to have a strong positive correlation with academic achievement [1], [2]. These types of studies are often criticized for failing to control for parental influence among other significant factors. Yet, the positive impact music hobby seems to have is worth investigating further. The problem is the complexity in the ways in which playing, hearing and producing music influence various areas of life from emotions [3] and thinking [4] to learning to learn [5]. In order to investigate how music supports thinking skills needed especially in software engineering and the 21st century, we observe the similarities and differences between the processes of music composing and programming. Accordingly, we propose the following research questions:

1) What are the similarities and differences in the thought processes of programming software and composing music?
2) What opportunities does technology-assisted music composing provide for teaching 21st skills and programming?

II. Background

A. Music Composing and Programming

Music notations share similarities with computer program code. Classically trained musicians are able to read sheet music i.e. musical code and execute it accurately based on how the composer intended [6]. Sheet music still leaves room for interpretation in terms of, for example, note velocity, type of vibrato, timbre etc [6]. In programming the computer executes program code, however arguably doing less errors and interpretation in the process compared to human musicians playing a score.

Perhaps more interesting than the execution phase is the creation process. Both, programmers and music producers, work with digital tools to create executable artifacts. Both processes involve design thinking and constant evaluation of the artifact. Furthermore, software as well as music consists of small pieces which form larger constructs, which then eventually come together to form the artifact: software or music composition. This is illustrated in Fig 1, which shows a rock song split into verses, which are then further split into the individual instrument tracks that make up the verse. Composers need to constantly refer to the relationships between instrument tracks and how they relate to other tracks in the same verse as well as how the verse relates to the rest of the song. Similarly software engineers need to think about algorithms inside methods, in which classes the methods belong to, what interfaces parts of the software offer each other and how they communicate together.

Despite the above mentioned similarities between music and programming, music notations are not Turing-complete [7] and cannot be used as such to write software. The process does work the other way around, as demonstrated by several programming tools aimed at learning computer science via music creation. For example, the *Sound Thinking* program offered a tangible interface (PicoBoard) to create music and learn programming [8]. Peng on the other hand, suggested that computer science concepts could be learned via a computational drumkit [9]. Because of the logical structures that are present in music, it has also been suggested that introductory computer science courses could contain exercises where stu-

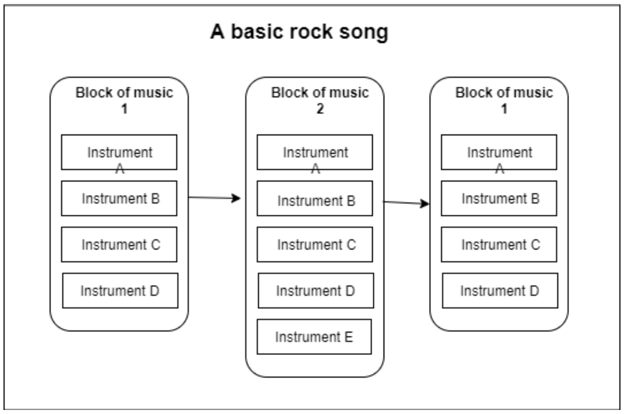

Fig. 1. Visualizing songs by dividing them into verses or blocks

dents use Python or other programming languages to generate music [10]. This has been taken further in environments such as JythonMusic trough which both programming and music theory can be learnt [11]. These experiments further prove that there is a call for conceptualizing the multitude of implicit learning benefits which music composing offers.

B. 21st Century Skills in Music Composing

In order to connect the educational benefits of music composing to concrete thinking skills, we review relevant work on 21st century skills, which describe the kinds of skills that are needed to succeed in the society and working life in the 21st century [12], [13]. Several overlapping definitions exist as to what these skills are in particular [12]–[15]. A systematic literature review found that ”21st century skills” was used in academic papers to describe broader and a more abstract set of skills compared to digital skills [15]. The 21st century skills being higher level thinking and collaboration skills also pose challenges on how to assess and evaluate their learning [14]. Therefore, in designing education, it is paramount to isolate measurable learning goals amidst the 21st century skill umbrella term, such as design thinking [16], critical thinking [17] and computational thinking [18].

Many of the so called 21st skills have already been added into K-12 national educational curricula in several countries [19]. In most cases these skills are being taught through existing subjects during, for example math lessons, with traditional teaching methods [20]. However, scholars have also proposed alternative methods for teaching these skills such as project-based learning [21] and phenomenon-based learning [22]. To ensure a seamless integration of 21st century skills into existing subjects, a careful analysis of affordances of these subjects to teach the new skills is required. In this case, we investigate the opportunities that are provided by music composing.

Music is one of the natural school subjects in K-12 education through which 21st century skills can be taught due to its inherit logical structure [13]. Shuler argues in his work that music in particular affords learning the so-called four Cs: (1) Creativity, (2) Critical Thinking, (3) Communication and (4) Collaboration [13]. This list, however, is needlessly narrow, as music composing has been shown to have affordances for teaching, for example, various kinds of mathematics [23] and multiple thinking skills [24]. In fact, Hanna presents a revised version of Bloom's taxonomy with examples of cognitive processes present in music composing, many of which are applicable in programming as well, including (1) interrelationships among the basic elements within a larger structure, (2) evaluating musical compositions (or program code) and improve it and (3) Planning and creating songs (or program code) [24].

III. Research Design: Expert Interviews

This study invokes the qualitative empirical research methodology of expert interviews [25] for probing the thought processes and knowledge structures of professional composers and programmers. Both professional music composers (n=4) and programmers (n=5) were interviewed with semi-structured questions. The interviews were carried out during holiday season 2019-2020 and lasted between 30-60mins. A researcher and an observer taking notes were present during the interviews.

In analyzing expert interview data, it is crucial to link obtained information to existing evidence to avoid reporting a representation-driven knowledge base that is only loosely connected to reality [26]. For this reason, 21st century skills in K-12 national educational curricula in seven countries as identified by Lindberg et al. [19] were used as a backbone to which the expert interview findings were related and linked to.

We extracted meaningful expressions from participants replies and further categorised them thematically following the qualitative research method of open coding [27]. Two authors were present during the analysis. Inter-rater reliability was not obtained as coding and theme decisions were made cooperatively, instead of individually and then compared. With the resulting thematic categories we aimed to reach concrete thinking patterns that could be connected back to previous literature.

IV. Results

A. Thought Processes of Professional Music Composers

Four professional music composers and or producers were interviewed (29M,29M,31F,27M) with differing background and experiences. All had received formal music education with one having a Master's degree in music and others at least a secondary degree. Three of the experts were active recording artists. The composing software actively used by the experts included *Bitwig studio*, *Renoise*, *Pro Tools*, *Cubase* and *Sibelius*. All four composers used multiple approaches and strategies when composing and producing music, depending on their goals for the specific project they were working on.

The composers mentioned to use several composing techniques and approaches. Depending on the project they mentioned to alternate both the outcome they aimed at, or whether the aimed at one at all, as well as the actual process of creating a song. Individual compositions could also use multiple techniques. The most experienced composer said he recently enjoyed using guitar for composing as he had no idea what he was playing and had to rely on his hearing to know what was good. He continued: *"Knowledge of music theory can be limiting when coming up with new ideas. When later turning my guitar compositions into notes I have often been surprised by the weird key choice or chords I've used"*. All experts further stated that the technical interfaces through which they visualize music have an impact on the composing process.

Three of the four interviewed composers reported to have at some point experienced trouble interfacing their musical hardware with software, getting certain sounds working on their music software and having to have tweaked operating system level settings to, for example, get a new sound card working. This suggests that especially when using composing technologies, music producers gain general technical skills and know-how of computers. Furthermore, one of the composers mentioned to regularly use Renoise, a digital audio workstation including a scripting API (in Lua) for music creation. This is an example of how music production is already integrated with programming and consequently software development.

Collaborating and communicating was mentioned by all as a major part of the process of music producing and band activity. However, less so for the actual music composing process, despite the existence of technical tools for collaborative music composing [28], [29] and some bands such as Dream Theater coming up with majority of their music that way. Finally, one composer stressed that he usually spent at least ten times more fine-tuning his compositions and sounds compared to the actual composing process.

B. How the Interviewed Programmers Conceptualize Software

Five programmers participated in focus group interviews (44M, 29other, 27M, 36M, 29M). The participants had programming experience from various languages, including C++, Python, Java, Haskell, COBOL and JavaScript and several technologies such as desktop and mobile operating systems, development environments, version control tools (git, Mercurial) among others. Two participants were masters of software engineering, one was a bachelor and the two others undergraduate students. In addition to the semi-structured questions which were also used in interviews with music composers, the programmers were asked to draw with pen and paper how they visualize software and how they would represent a musical song with code.

Upon asking programmers to visualize the architecture of musical songs in program code the more experienced programmers split the song into smaller blocks which they further split into method-like structures. However, the more inexperienced programmers had unique ways to visualize software: The first one stated they preferred to visualize a user interface first and then think how all code can be connected to that. The second on the other hand, started thinking about the program execution from the main-method onward and visualizing the blocks as they approached where the execution was. This type of visualization often relates to flowcharts and is depicted in Fig 2. The flowchart view is the closest to music where the execution is always linear with no dynamic calls and only in one place at a time.

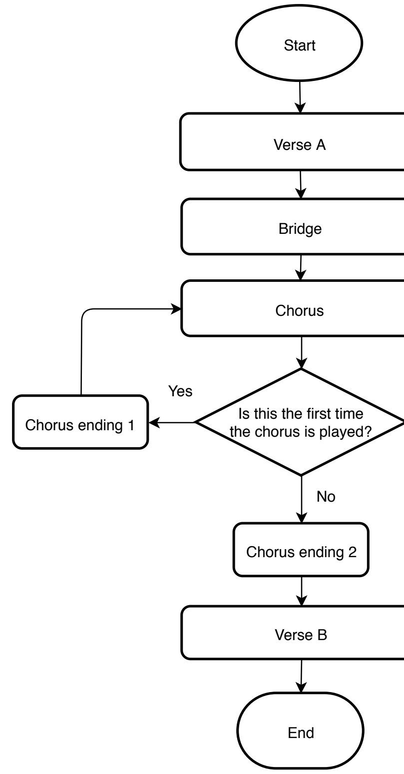

Fig. 2. An example of using a flowchart to present a song.

When discussing whether programmers need creativity in their work, the majority replied to the positive. However, the most experienced programmer noted that "creative code" is difficult to read and annoying, meaning they prefer to write as simple and effective software as possible. When asked about the need for technical know-how the programmers stated that it was mostly required in the early stages of projects at most, and the majority of their work related to actual programming. Social skills and cooperative skills were mentioned as useful especially in projects involving people outside software engineering, but for software engineers themselves, simple well written code with comments and documentation was mentioned to be the most useful cooperative element. Altogether the expert interviews with programmers gave rise to seven key thinking skills and processes which were useful during programming and/or software development:

- *Modular thinking*. Dividing programs into smaller parts and perceiving the interrelations between the components of a program was seen as an essential skill by all interviewed experts. Most often this was achieved by visualizing the structure of the program with different diagrams such as class diagrams. In object-oriented programming, this means dividing the program into distinct sections (such as classes and methods) with clear and separate purposes.

- *Visualizing the execution of the program*. Much like when composing a piece of music, it was found that visualizing the intended execution of a program, for instance by using flowcharts (see Fig. 2), is an important skill when designing software, although programs usually have a more complex structure involving conditional jumps, and possibly non-deterministic and parallel execution.
- *Choosing the correct tools*. Using an appropriate programming language, technology, framework or code editor was seen as an important skill, as it creates different restrictions and possibilities in the design and implementation process. This is similar to choosing instruments and composing tools for a piece of music. Using digital composing tools was also found to be helpful in gaining technical know-how and solving technical problems.
- *Recalling and applying the basic programming concepts.* Understanding and applying the basic concepts such as conditional structures, loops and variables was found to be essential. In many ways, these bear a resemblance to musical symbols, note values etc.
- *Aiming for reusability.* Several programmers indicated that they aim for reusability in their implementations. Although reusability does not play such a big role in composing music, recognizing specific styles of music, time periods, and elements associated with them is a central skill [24].
- *Evaluating quality attributes and refactoring.* Especially the more experienced programmers saw reviewing the quality attributes (such as performance and security) [30] as an important part of implementing software. This also pertains to the iterative process of evaluating and critiquing the code and architecture design and improving it by refactoring and optimization.
- *Avoiding needless creativity.* Programmers did not want to exercise needless creativity. Reusing earlier solutions and writing simple, even boring code was preferred over creating new and different approaches. As one of the interviewed programmers put it: "You should avoid getting creative when writing software. The simplest and most efficient solution is often the best. Once I master a programming language, I look for new challenges by moving on to new technologies".

C. Opportunities of learning these skills via technology-assisted music composing

To ground the findings from experts groups into existing literature as guided by the methodology [26], we compare the expert group findings with programming topics in K-12 curricula in seven countries as identified by Lindberg et al., [19]. The thinking skills required can be summarized to be: (1) algorithmic thinking, (2) computational thinking (3) loops, conditionals, variables, (4) data structures, (5) digital logic, (6) input/output and (7) software design/modularity. The comparison revealed significant overlap as shown in Fig I. Yet, we notice that some aspects identified by our expert groups are currently not present in K-12 educational curricula, such as quality attributes and refactoring, visualizing program execution and avoiding needless creativity.

TABLE I
SIMILARITIES BETWEEN MUSIC COMPOSING AND PROGRAMMING

Curricula	Programmers	Musicians
algorithmic thinking	yes	some
computational thinking	yes	no
loops, conditionals	yes	yes
data structures	yes	yes
digital logic	some	some
input/output	yes	yes
software design/modularity	yes	yes

Additional analysis is required to see what further possibilities music composing provides for learning 21st century thinking skills useful in programming beyond current K-12 educational curricula. A comparison between the two expert groups reveals six thinking skill constructs which are displayed in Fig II. Despite interviewing experts, even this is not an exhaustive list as scholars have previously suggested thinking skills including planning (design thinking) to be learnt via music creation [24].

TABLE II
SIMILARITIES BETWEEN MUSIC COMPOSING AND PROGRAMMING

1. Use a defined syntax to create logical structures
2. Read syntax to understand what is meant by the creator
3. Hierarchical and modular structures
4. Refactor or adjust syntax to make it better
5. Iterative process. Constantly check the artifact for visual or audio feedback.
6. Understand how parts of the artifact interface with each other.

V. Discussion

The results of this study revealed significant overlap between key skills needed in the work of professional programmers and music composers. Furthermore, when mapping the skills to programming content seven K-12 educational curricula, four primary concepts that could be learnt via music composing emerged: (1) loops and conditionals, (2) data structures, (3) input/output and (4) software design and modularity. In addition, cooperative elements, technical know-how, planning and design thinking, evaluation and refactoring and working iteratively on an artifact were identified as skills needed in both music composing and software development. These skills can be further mapped to knowledge structures which are important for both composers and programmers.

A. Implications of Findings

Music teaching is being sidelined in several countries or even excluded from educational curricula [31]. One cause for this might be that music in itself lacks relevance to working life and more important subjects take its place. However, building off previous work [10], [11], [13], [23], [24], this study provides further evidence towards the power of music creation to teach skills that are necessary in the 21st century. The findings also offer some explanation to the reported implicit learning that takes place when practicing music [1], [2].

B. Limitations

The chosen research methodology of expert interviews has been criticized for being subjective [32], similarly to focus groups and ethnographic methods, but it remains an effective way to obtain expert knowledge on fields were still great deal of conceptualization and evidence is needed [25]. The small number of participants (N=9) and participants being recruited from a geographically small area bring into question whether the final list of similarities between programming and music composing was exhaustive.

VI. Conclusions Future Work

This study found six categories of thinking skills that are present in both music composing and programming. Furthermore, music was found to have affordances to teach four categories of programming knowledge currently found in K-12 educational curricula in seven countries. Additional skills bridging technology-assisted music composing and programming were also discovered such as technical know-how, cooperation, ability to evaluate and improve artifacts and planning (designing) artifacts. The results highlight the following three venues for future studies:

1) Empirically verifying the translating effect of music composing to programming.
2) The results can be used in designing music composing technologies that aim to teach programming concepts on the side.
3) The pedagogical strategies used in teaching music theory could be adopted for programming education and vice versa.

References

[1] C. dos Santos-Luiz, L. S. Mónico, L. S. Almeida, and D. Coimbra, "Exploring the long-term associations between adolescents' music training and academic achievement," *Musicae Scientiae*, vol. 20, no. 4, pp. 512–527, 2016.

[2] M. Guhn, S. D. Emerson, and P. Gouzouasis, "A population-level analysis of associations between school music participation and academic achievement.," *Journal of Educational Psychology*, 2019.

[3] K. S. McFerran, "Contextualising the relationship between music, emotions and the well-being of young people: A critical interpretive synthesis," *Musicae Scientiae*, vol. 20, no. 1, pp. 103–121, 2016.

[4] S. M. Ritter and S. Ferguson, "Happy creativity: Listening to happy music facilitates divergent thinking," *PloS one*, vol. 12, no. 9, p. e0182210, 2017.

[5] S. J. Havre, L. Väkevä, C. R. Christophersen, and E. Haugland, "Playing to learn or learning to play? playing rocksmith to learn electric guitar and bass in nordic music teacher education," *British Journal of Music Education*, vol. 36, no. 1, pp. 21–32, 2019.

[6] M. Puurtinen, "Learning on the job: Rethinks and realizations about eye tracking in music-reading studies.," *Frontline Learning Research*, vol. 6, no. 3, pp. 148–161, 2018.

[7] A. Hodges, "Alan turing and the turing machine," *The Universal Turing Machine a Half-Century Survey*, pp. 3–14, 1995.

[8] A. Ruthmann, J. M. Heines, G. R. Greher, P. Laidler, and C. Saulters II, "Teaching computational thinking through musical live coding in scratch," in *Proceedings of the 41st ACM technical symposium on Computer science education*, pp. 351–355, ACM, 2010.

[9] H. Peng, "Algo. rhythm: computational thinking through tangible music device," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pp. 401–402, ACM, 2012.

[10] A. Misra, D. Blank, and D. Kumar, "A music context for teaching introductory computing," *ACM SIGCSE Bulletin-ITiCSE'09*, vol. 41, no. 3, p. 248, 2009.

[11] B. Manaris, B. Stevens, and A. R. Brown, "Jythonmusic: An environment for teaching algorithmic music composition, dynamic coding and musical performativity," *Journal of Music, Technology & Education*, vol. 9, no. 1, pp. 33–56, 2016.

[12] L. C. Larson and T. N. Miller, "21st century skills: Prepare students for the future," *Kappa Delta Pi Record*, vol. 47, no. 3, pp. 121–123, 2011.

[13] S. C. Shuler, "Music education for life: The three artistic processes—paths to lifelong 21st-century skills through music," *Music Educators Journal*, vol. 97, no. 4, pp. 9–13, 2011.

[14] P. Griffin and E. Care, *Assessment and teaching of 21st century skills: Methods and approach*. Springer, 2014.

[15] E. Van Laar, A. J. Van Deursen, J. A. Van Dijk, and J. De Haan, "The relation between 21st-century skills and digital skills: A systematic literature review," *Computers in human behavior*, vol. 72, pp. 577–588, 2017.

[16] R. Razzouk and V. Shute, "What is design thinking and why is it important?," *Review of educational research*, vol. 82, no. 3, pp. 330–348, 2012.

[17] A. J. Rotherham and D. T. Willingham, "21st-century" skills," *American Educator*, vol. 17, no. 1, pp. 17–20, 2010.

[18] A. Yadav, H. Hong, and C. Stephenson, "Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in k-12 classrooms," *TechTrends*, vol. 60, no. 6, pp. 565–568, 2016.

[19] R. S. Lindberg, T. H. Laine, and L. Haaranen, "Gamifying programming education in k-12: A review of programming curricula in seven countries and programming games," *British Journal of Educational Technology*, 2018.

[20] D. M. Gut, "Integrating 21st century skills into the curriculum," in *Bringing schools into the 21st century*, pp. 137–157, Springer, 2011.

[21] S. Bell, "Project-based learning for the 21st century: Skills for the future," *The clearing house*, vol. 83, no. 2, pp. 39–43, 2010.

[22] K. Lonka, J. Makkonen, M. Berg, M. Talvio, E. Maksniemi, M. Kruskopf, H. Lammassaari, L. Hietajärvi, and S. K. Westling, *Phenomenal learning from Finland*. Edita, 2018.

[23] S. Laato, T. Laine, and E. Sutinen, "Affordances of music composing software for learning mathematics at primary schools," *Research in Learning Technology*, vol. 27, Sep. 2019.

[24] W. Hanna, "The new bloom's taxonomy: Implications for music education," *Arts Education Policy Review*, vol. 108, no. 4, pp. 7–16, 2007.

[25] M. Meuser and U. Nagel, "The expert interview and changes in knowledge production," in *Interviewing experts*, pp. 17–42, Springer, 2009.

[26] N. M. Cooke and J. E. McDonald, "A formal methodology for acquiring and representing expert knowledge," *Proceedings of the IEEE*, vol. 74, no. 10, pp. 1422–1430, 1986.

[27] A. Strauss and J. Corbin, "Open coding," *Basics of qualitative research: Grounded theory procedures and techniques*, vol. 2, no. 1990, pp. 101–121, 1990.

[28] M. Biasutti, "Group music composing strategies: A case study within a rock band," *British Journal of Music Education*, vol. 29, no. 3, pp. 343–357, 2012.

[29] M. T. Hopkins, "Collaborative composing in high school string chamber music ensembles," *Journal of Research in Music Education*, vol. 62, no. 4, pp. 405–424, 2015.

[30] F. Losavio, L. Chirinos, N. Lévy, and A. Ramdane-Cherif, "Quality characteristics for software architecture," *Journal of object Technology*, vol. 2, no. 2, pp. 133–150, 2003.

[31] S. Laato, F. Gideon, R. Shivoro, E. Sutinen, and N. Pope, "Identifying factors for integrating math and music education at primary schools in namibia," *IEEE*, 2019.

[32] H. Dorussen, H. Lenz, and S. Blavoukos, "Assessing the reliability and validity of expert interviews," *European Union Politics*, vol. 6, no. 3, pp. 315–337, 2005.