Cooperative Live Coding as an instructional model

Antonio Deusany de Carvalho Junior
Instituto de Matemática e Estatística
Universidade de São Paulo
dj@ime.usp.br

***Abstract*—The advances on technologies have provided many tools that inspired new instructional models. Learners and instructors are experiencing a diverse environment where everyone can participate from anywhere in the world and share the same learning platforms. Although we already have some manuals, tutorials, and also MOOCs that can be useful for people who wants to learn Computer Music languages, the musical interaction is not offered in these solutions. In this paper we present an instructional model for computer music and live coding based on a cooperative live coding environment where participants can teach and learn through distributed pair programming. We also discuss the fundamental ideas and the tool used on this work during the first experiments.**

I. Introduction

New instructional models have been provided by the advances on technologies and they are becoming popular even without a long evaluation regarding their advantages. Most of them are web based and can be accessed at any time by thousand of students through the Internet. This structure for instructional model takes advantage of the Internet as it is and these models requires a basic setup to some extent. From this point of view, the Internet permits the dissemination of hypermedia to the whole society through the use of digital technologies for information and communication [1].

The proposal of an instructional model self-contained on the Internet also rises from the interaction or interactivity aspects behind the model. One can say that the interaction implies the transmission of information in unidirectional point of view, while the interactivity rises from the two-dimensional participation of the learner [2], or that the interactivity idea is to focus on the observed object that is modified from time to time [3]. Furthermore, the interactivity also includes many features that permits participants to modify and intermediate between the information received and the resultant knowledge.

In this work we are going to present a new instructional model for computer music and live coding based on a tool for cooperative live coding that provides an online environment with many ways of interactivity. We are also opening a discussion about cooperation and collaboration practices throughout the use of this tool, initially proposed for collaborative live coding by the authors. Even though cooperation and collaboration terms may be interchangeable used, we will consider that the cooperation does not imply mutual benefit, and that a collaboration assumes contribution of all participants with mutual goals [4].

The next sections include a discussion about the advantages of new instructional models that involve online systems. We will present how users can learn cooperatively and which environments can be used to provide online interaction. The tool created by the authors is also presented, and we will discuss the advantages of its use for teaching computer music languages with musical interaction.

TABLE I. Sites with MOOCs

Name	Website
edX	https://www.edx.org
Coursera	https://www.coursera.org
NovoEd	https://novoed.com
Udacity	urlhttps://www.udacity.com
MiriadaX	https://www.miriadax.net
Futurelearn	http://futurelearn.com
OpenUpEd	http://openuped.eu
P2PU	https://p2pu.org
UoPeople	http://uopeople.edu

II. Online instructional models

An instructional model is a set of instructions or directions that provide a way to acquire some knowledge, improve capabilities, or extend the practice. Although we have a variety of instructional models available for classroom, in the past two decades many technologies became available and inspired new instructional models.

A new contemporary instructional model is the Massive Open Online Courses (MOOCs). This model offer online courses that can be attended by thousand of people at the same time. The systems often offer syllabus, videos, and forum as the main features for their participants. The idea of MOOCs has been a new tendency in many areas of teaching. People around the world are engaging in online courses alone or in groups and learning anything at anytime. The instructors provide updated materials through the syllabus and propose exercises with online evaluation. The evaluation is automatic in most of the time due to the unlimited number of students that can sign up for the courses. We have many examples of sites with MOOCs in Table I.

The interaction between participants (professors and students) in these sites happens in the forums where anyone can post, answer, and discuss questions at any time. Although the students have this open channel with the professor, the feedback is not always immediate and the professors may take some time to answer questions from all students. It is also noticeable that some instructors have a Teaching Assistant (TA) responsible for the forums. Although the addition of a TA may increase the distance between the students and the instructor, the TA will be more attentive to the forums while the instructor

978-1-4673-9143-6/15/$31.00

TABLE II. MOOCs WITHOUT SCHEDULE

Name	Website
Khan Academy	https://www.khanacademy.org
Udemy	urlhttps://www.udemy.com
Pluralsight	http://www.pluralsight.com
Code School	https://www.codeschool.com
Digital Tutors	http://www.digitaltutors.com
Three House	https://teamtreehouse.com
Veduca	http://www.veduca.com.br
Acamica	https://www.acamica.com
Codecademy	http://www.codecademy.com

will be responsible only to organize the learning objects and syllabus.

The MOOCs are very helpful but some courses are not offered all the time. In case someone has an urgency in learning a specific topic, the MOOCs presented on Table I may not be the best option as these sites present courses on a predefined schedule. However, autodidact (self-taught or self-learner) students have other options of MOOCs sites where the courses are always open. A list of these sites is presented on Table II. In this case the instructors keep the courses open and update the materials from time to time.

The MOOCs have full courses and predefined materials with fixed structure. This instructional model follows the traditional teaching method applied at the classrooms where the student will need to learn everything without any adjustment for his/her own difficulties. While some students will abandon the course due to the lack of basic concepts, other students may find the course tired when they already have previous background and the course does not go further than expected.

In terms of specific content about any topic or special necessities (e.g. learning step-by-step with variable intensity and extra content), the MOOCs may not be the best option. Students interested in technological topics, like programming languages, would probably try online tutorials or specific forums.

Instructors and students are sometimes interested in teaching or learning specific topics that don't need a full course, and the online tutorials are an alternative that can suit better their needs. The tutorials are distributed online mostly in textual formats and can explain the same topic in many levels.

Online tutorials about many subjects and areas can be found through online search tools. Still, the technological tutorials are probably the most distributed through Internet users. The official sites of many programming languages and applications have tutorial sections on their website or include the tutorials inside contents downloaded by users. A list of sites with tutorials is presented on Table III, and these sites may present video tutorials and also courses based on their tutorials.

The W3Schools is one of the most famous site with tutorials about web technologies. All tutorials are deeply detailed including the description of the programming language paradigms, recommendations, attributes, statements, options, and advantages. Additionally, this site also introduces some tools to run code online and execute data base statements through the web interface, that we are going to talk afterward.

TABLE III. SITES WITH TUTORIALS

Name	Website
W3Schools	http://www.w3schools.com
Vogella	http://www.vogella.com
Tutorials Point	http://www.tutorialspoint.com
Tuts+	http://tutsplus.com
TechTutorials	http://www.techtutorials.net
Home & Learn	http://www.homeandlearn.co.uk

Some web technologies has great communities that build online tools in order to help new learners. An example is the Try Ruby[1] website. This is the recommended online tutorial for everyone interested in learning how to program in Ruby language. The site tries to talk with the users like a real instructor and go step-by-step covering the basics of the language. Try Ruby is a short tutorial but it demonstrates how a computer system can interact with a learner during its initial practices. Next we will discuss some solutions where real users (the learners and professionals) can work together in a collaborative and cooperative way.

A. *Cooperative learning*

The use of sites with focus on question and answer (Q&A) extends the instructional models supported by online environments. Although the users may need at least a basic knowledge beforehand, Q&A sites provide search tools for questions related to a specific topic that the users want to learn or to discuss. Some discussions can grow up to many answers and diverse point of view from users around the world. While some users take advantage of these sites only to solve personal problems, many others have the position of supporting other users and expending some time in more detailed discussions and explanations.

The StackOverflow[2] is one example of these sites and it does accept questions about many programming languages. This site is part of StackExchange[3] community which includes other kind of Q&A sites from diverse topics including life, arts, culture, recreation, and science. It is also possible to propose ideas for new Q&A sites at Area 51[4] creation zone from StackExchange.

The StackOverflow site provides some rules that guide initial users in order to help other users to cooperate in solving some technological questions. For programming languages, one of the main rule is to present a minimal, complete, and verifiable example (MCVE). The MCVE will help any other online user to have a detailed idea of the context of the question on most situations. In case the user wants to fix some code regarding Javascript, HTML, or CSS, the recommended procedure is to create a MCVE at JSFiddle[5]. If the user wants to write a specific database statement using SQL, one option to write the MCVE is the SQLFiddle[6]. Both options implement the same idea presented at W3Schools and permits the users

[1]Try Ruby: http://tryruby.org

[2]StackOverflow: http://stackoverflow.com

[3]StackExchange: http://stackexchange.com

[4]Area 51 - The Stack Exchange Network staging zone: http://area51.stackexchange.com

[5]JSFiddle: https://jsfiddle.net

[6]SQLFiddle: http://sqlfiddle.com

TABLE IV. TOOLS FOR DPP

Name	Website
collabedit	http://collabedit.com
CodeShare	http://www.codeshare.io
Cloud9	https://c9.io
Squad	https://squadedit.com
Floobits	https://floobits.com
MadEye	https://madeye.io

to modify the code on the site and see the results without any other specific tool or even the necessity of creating a database for the latter option.

A special feature available at JSFiddle is the collaboration option. The user can share the link of the 'fiddle' created at this site and invite other users to work together on the same code. This idea of having many users working on the same code is also known as pair programming and will be discussed below.

B. Pair programming

There are many ways of working and practicing with programming languages in order to apply agile methods. One famous practice is the Pair Programming (PP), that involves two programmers working on the same piece of code at the same time. This practice suggests only two programmers but it is not restricted to this structure. Although PP is most used by developers that work in companies, research shows that PP can be also used as tool for practicing and also teaching programming [5]. The advances on the technologies have permitted this practices being extended to distributed places.

In the same way as JSFiddle, there are many tools that provide an ambient to share code and allow many users to program together through the Internet. This practice is named Distributed Pair Programming (DPP) and considers users on different machines and locations. There are tools available for DPP and some of them also integrate audio, video, and chat features. A list of DPP tools is presented on the Table IV.

The idea of sharing the code and running online with the results being presented to more than one user at the same time has many advantages. Users can apply the PP concepts where one user only observes while the other one is coding. It is possible to have each user programming a different method on the same file in order to finish the program faster and collaboratively. Some companies can also use this environment to evaluate programmers online from different places and observe how the candidate can solve a problem during an interview. Recent research about DPP literature shows that 'there is a strong trend towards the use and research of the empirical effects of DPP in teaching programming' and that 'there is an opportunity to investigate DPP with other types of collaborative programming' [5].

The fundamentals discussed in this section serve as a base to cooperative live coding practices that are going to be discussed in the next sections. We will start presenting the *live coding* that is a musical practice based on writing code lively and evaluating this code without compilation.

III. LIVE CODING

During the 80's and 90's, composers and performers would write piece of codes in some languages like Csound and wait some time to have the result. [6] discuss that after learning Csound from online documentation in 1996, a specific file with tens of thousands interacting instruments took 20 hours to build, and it sounded dreadful in the end [6, p. 81]. During an interview, Judy Klein remember that during the 80's she would start a compilation of a short piece of sound at night and hear the result in the morning, when no errors had occurred [7]. Nowadays even cheap computers like Raspberry Pi[8] can be used as sound processor with new Computer Music languages, and we can say that the practice of live coding is built upon the technological advances that permit high processing of codes and sound synthesis in a matter of milliseconds.

The idea of writing codes lively is similar to the paradigm behind interpreted languages where the user writes line by line and have the results right after the evaluation has been done. One can cite Javascript as an interpreted language on client side, where the code is not compiled before being executed. Ruby and Python are also normally taken as interpreted language, but this condition depends on the compiler implementation and may differ between versions.

In Computer Music, we have lots of programming languages and some of them are interpreted languages that can be used for live coding. The performers can write pieces of code and hear the results while they continue to write new lines or to modify the same code. No deep programming skills are required for most of the languages used by live coders and the practice have been diffused between people from many areas, specially musicians and artists.

The practice of live coding in music is mostly used on laptop performances and can be achieved with many languages [7]. Research shows that the use of interpreted languages resulted into the rising of live coding movement and that it has become a common practice in festival calls that include electronic music [8]. This art specialization is basically a variation of improvisation and composition using algorithms, and can be briefly defined as 'a form of musical performance that involves the real-time composition of music by means of writing code' [9].

Live coding can also include technical mistakes as any other musical practice [10]. An error on the code can result in an unexpected sound and can generate an undesirable noise or an undesirable traditional pattern depending on the music style. The system can also crash during a performance, and other software or hardware can intercept the sound processing without any control from the performer side. The practice and comfortableness with the language used can give confidence to the performer, and some instructional exercises for this aim are presented on the literature [10].

The live coding practice can also be done by many users at the same time. In this case, the collaboration during a musical performance has a structure similar to a traditional orchestra or band, but the coders may not have a specific

[7] Interview with Judy Klein: http://ias.umn.edu/2013/03/01/electronic-music/#Klein

[8] Raspberry Pi: https://www.raspberrypi.org

instrument, sound, timbre, or function. The performers can also interconnect themselves using some kind of network and share sound or codes, what we would call network music.

Experiments with network music dates back to the 1970's with The League of Automatic Music Composers [11]. The setup of this group of composers was based on desktop computers connected to a local network. Each member of the group was responsible for some musical feature during the performance and the interaction between them would take place through the local network, and later, through phone lines. Many other network pieces have been attempted in the context of laptop orchestras, including the Princeton Laptop Orchestra (PLOrk) [12], the Stanford laptop orchestra (slork) [13] and Linux Laptop Orchestra (L2Ork) [14]. All of these orchestras focus on local network solutions for communication and have an infrastructure for small ensembles. Additionally, we have some works where the performers use local network to share data and communicate within the ensemble [7], [15]–[18].

Following these ideas, one can cite the Republic [9] quark package that is used to create synchronized network performances. In this case, the live coders would have to start a server and have all users connected in order to start sharing code and interacting. Another work that have similar functionality is the extramuros [10], a system for network interaction through sharing buffers of any kind of language. The last solution needs to be configured depending on the language and at the time of this paper it does not present any easy way to stop synthesis on SuperCollider. Both solutions requires a server on one computer to receive connections from clients, and additionally it would be necessary to open network ports or change firewall settings before starting any interaction.

The collaboration in live coding is also presented on the Gibber[11] library for WebAudio. The users can synthesize code on the browser and talk through a chat room in other to have a collaborative online session. Although this solution permits users to share code and synthesize online, there is no way to share the same code environment at the same time, and the users will probably need to share the code through the web chat if they want to try a live session. In the session we will then present SuperCopair, a tool created by the authors that can fulfill all the blank spaces for collaborative live coding and also inspire the cooperative live coding practice.

IV. SUPERCOPAIR

SuperCopair is an application created as a package for the Atom.io[12] IDE. This IDE has numerous packages for many programming languages and presents some solutions for coding, debugging, and managing projects. Atom packages are programmed in CoffeeScript[13], which is a programming language that can be converted to Javascript and can also integrate its libraries. The developers can install Atom packages to enable various functionalities in the IDE such as: communicate through chats, use auto-complete in certain programming language syntax, interact with desktop and web applications, integrate with the terminal command line, and have many options based on other packages. In the same way, the users can just search for SuperCopair and install without many steps.

SuperCopair is based on two Atom packages: atom-supercollider and atom-pair. The first package turns Atom.io into an alternative IDE for SuperCollider programming language and permits users to openly communicate locally with SuperCollider audio server through OSC in the same way we can do on SC-IDE, the default IDE. Additionally, the users can take advantage of packages from Atom and quarks together in the same interface. The latter package is used for pair programming through the Internet. The atom-pair package is based on Pusher cloud service and its default configuration is based on the community free plan, but a user can modify the settings and use his/her own keys. We decided to merge both packages to add new features for collaborative live coding, and finally had dubbed it the SuperCopair package. The main idea is that all participants evolve into a collaborative practice and performance.

The IDEs for SuperCollider have shortcuts to evaluate a line, a block, and to stop all sound process that is running. In addition to these options, the SuperCopair package includes methods and shortcuts that can broadcast the events cited and execute them on all users connected at the same pairing session. Through the shortcuts, one can decide to evaluate selected code either only in the local machine or in all computers on the same session. One can also turn on and off a broadcast alert option in the settings in order to be asked or not before evaluating every broadcast event sent by another user in the same session. These options allow each individual to have control over the code that can be evaluated in the local machine.

The broadcast events are diffused to all connected users through the cloud service and the package evaluates on arrival each event message received. The message includes the code to be evaluated and the user identification. A representation of a session using SuperCopair package is shown at Figure 1.

Before inviting others, users need to start a new session following an instructional step-by-step setup presented on the package page. Once a session starts, the session ID string needs to be shared between collaborators that want to join the same session. The shared session ID is based on the channel created at the cloud service and it contains the user's keys. Every user who joins a session will see the most recent version of the shared code. The users can identify each other by different color markers on the left side of the shared file representing the line or lines in edition. A pop up provides information about users joining or leaving the session. Furthermore, a message including user's identification and the code evaluated appear at SuperCollider post window right after each broadcast event is evaluated. In case the broadcast alert option is on, a dialog will appear whenever the user receives an event message from another participant and this dialog asks if the user would accept or reject the code evaluation. The alert dialog will have the sender's id and the code sent via broadcast.

To summarize, this tool permits both participants to share a distributed audio environment and listen to the same sound when synthesized in a broadcast manner. SuperCollider live

[9]Republic quark: https://github.com/supercollider-quarks/Republic
[10]extramuros: https://github.com/d0kt0r0/extramuros
[11]Gibber: http://gibber.mat.ucsb.edu/
[12]Atom.io: http://atom.io
[13]CoffeeScript: http://coffeescript.org

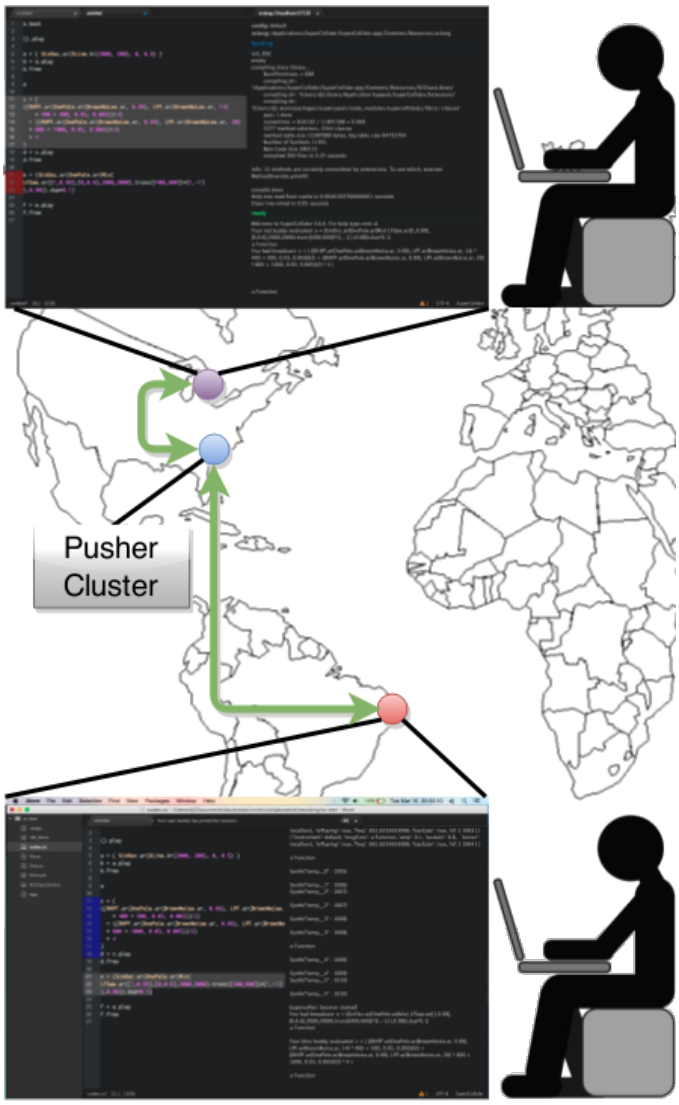


Fig. 1. Session with two users from different countries.

coding sessions through SuperCopair will take advantage of the easy setup and the minimal requirement of an Internet connection, instead of the network and server configuration required by the other solutions cited. Although the main idea behind this project is to provide a tool for collaborative live coding, new usability is proposed in this paper as a result of reflection after some experiments. In the next section we are going to discuss the possibilities of learning program languages related to music and the new paradigms focusing on cooperation and musical interaction.

V. Cooperative Live Coding

The cooperation and collaboration terms are similar regarding the idea of working together, but they differ in terms of benefits offered and tasks equilibrium. Inside a collaborative environment, we have all participants focusing on the same goal and working together to achieve same objectives. One can suggest that the players of the same team on any sport game are working collaboratively in order to win the game. On the other hand, a cooperative environment provides different benefits for the participants and they may have dissimilar objectives. Following the last example, two teams might cooperate during a match following the rules of the game in order to have a fair play, but they will not share the same goal (because one has to win) and each team probably benefits from the other's (failed) activities.

The cooperation can also be discussed in other ambit like: biology, when some species interacts in mutualism; business, when one company offer a service to the other, even if they share the same market; and personal life, when a relationship end up in divorce due to the unsuccessful collaboration between the participants or to the absence of shared goals. These examples illustrate the many characteristics of the cooperation and how it can be applied in different areas. In this section we will discuss the aspects of cooperative live coding in computer music education.

Networked live coding environments have been focused in collaboration as we presented on Section III. All participants are working together to execute a musical piece and share the same final result as a common goal. In most of the cases, sound is the only artifact that is shared, but the performers can also share codes. Some tools also enable chatting, audio, and video interaction between the participants, whether they are sharing the same physical space or they are in different locations. Although the experiments with SuperCopair have been conducted as distributed collaborative live coding sessions, the cooperative environment has emerged during some moments.

During the sessions using SuperCopair, we had participants from different levels of experience on live coding and also some new users that decided to learn how to code during the session. The collaboration on the final sound was similar to other live coding sessions, but the experience added another way of interaction in live coding: the cooperation.

The cooperation aspect emerged from some events that came into view from the live coding sessions. Experienced users are faster and have written lots of code from the scratch without any problem, while the apprentices start from basic structures or from portions of codes available on the file. As all users were trying to synthesize the codes on all computers, it became easy to perceive if something was going wrong because everybody was sharing the code, evaluation, synthesis, and errors. The errors coming from novice programmers were often fixed by the experienced users, and they also discussed the solutions between themselves using comments on the file. Some tricks from the live coding practice were introduced by advanced users and the initial learners rapidly became instructors for new users and these instructors were following the same cooperative practices of the experienced users. These events inspired the instructional model that is proposed and described in the next section.

VI. The Instructional Model

The discussion about the instructional model starts with an example of a cooperative live coding session that is presented on Figure 2. In this session we have two users working on the same file through the SuperCopair package on Atom.io IDE. Both users can see the line that is being edited by the other user following the color mark at the line number column. The comments presented on the file explain the following codes.

In this live coding session the users have the same benefits of any online environment for PP or DPP. However, this session presents more benefits and advantages due to the cloud service and IDE integrated through SuperCopair.

While some solutions presented at Subsection II-B have cloud services on their background, the cloud service used in our solution offer an adaptable and easy-to-setup environment for multiple users. Users can pay for dedicated infrastructures in order to avoid the free service limitations, and it is also possible to select the preferable cluster to connect. The network administrators from the service will take care of data distribution system which has a high level of abstraction from the user point of view. A key from the service is all the client need in order to use the service requested and it encourages the adoption of this service by novices and advanced users.

The IDE adopted includes many features that will hardly be available at the solutions discussed at Session II. There is a community working on packages for this IDE, making lots of options available, and supporting the development of packages like SuperCopair. Atom.io IDE offers common features like code completion, many shortcuts, and code highlight, but it also presents almost three thousand packages available with other features at the time of this paper.

A. Educational aspects

Although the cloud service and IDE bring many advantages for this live coding session, the combination of their features imply a new educational concept for computer music and live coding that we introduce below. We also have a discussion regarding educational values from the instructional model proposed herein. All the instructions and directions can be applied to local or local network structures as the instructor and learner can share or not the same classroom during the process. We can also consider a mixed environment where a group of participants share the same place while other participants are remotely connected to the same session.

1) Use comments for discussion: In this instructional model, the learner and the instructor can write comments with questions or instructions on the file. Comments are useful in this case because they will not be interpreted or cause errors even if the whole file is evaluated. The communication through comments is important if the instructor and the learner share the file from different locations, but the comments may be avoided whenever all users are in the same room.

2) Assist and fix codes: The cooperative live coding is also an advantage for collaborative live coding performances. An experienced user of the language can audit the performance and help to fix codes while other are performing. This special user does not need to participate in the performance as musician, and can also act just as a conductor writing comments in order to guide the live coders during the performance. The users of forums and mailing lists for SuperCollider questions can schedule meetings to code together. The sessions are not restricted for two users and the final code can be posted again in the same place where the discussion has started.

3) Interact during classes: Programming classes can include more interaction if all students can access the same file as the instructor. The code can be synthesized on all computers at the same time, and the students can hear the sound at their own computers without great speakers on the classroom connected to the computer of the instructor. Some students can make use of headphones when preferable, and the number of students connected is not a problem due to the advantages of the cloud computing behind the system. Both students and instructors can participate on the class in this model even if they are at home or traveling to a conference. Additionally, the students can interact directly and instantly on the code learned, as they only need to add code or comments from their own computers during the class. This model can also provide benefits to the instructor, which can ask questions for the students and see the code being written lively on the file shared.

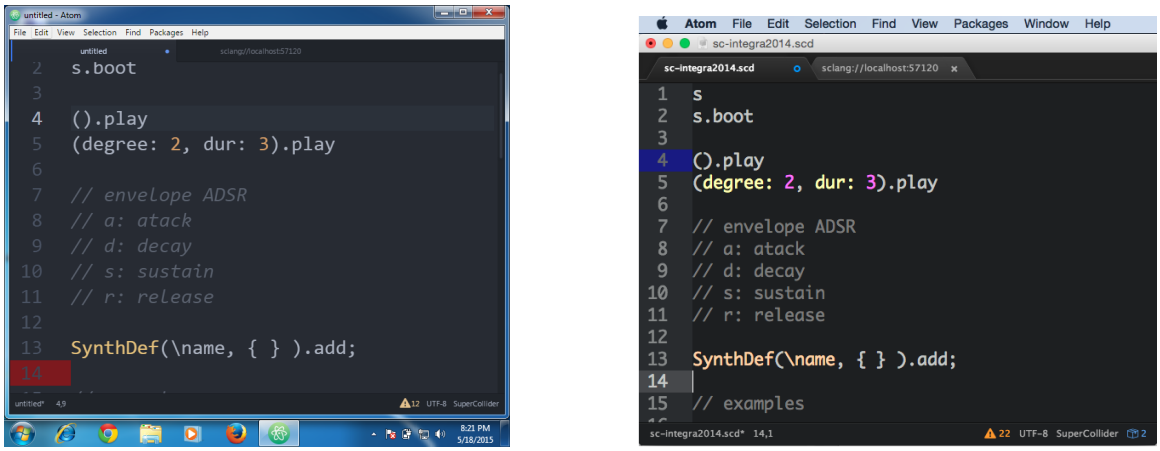
4) Offer online remote tutoring: There are students that prefer personal instructors as a supplement to the learning process, and some instructors prefer to work from home or want to have students from different locations. In this case, both can have benefits from this model. The instructors will not need to move between the students houses and can schedule more classes without intervals between them taking advantage of the interaction through the Internet, and the students can have instructors from different countries. The best students can also offer assistance to others after classes even without an specific place to meet together. On the other hand, experienced live coders can share their knowledge at workshops without physical participation, while the audience can join the same session, share the code, and ask questions on the file The audience can also be distributed, and the proposal of workshops will make the most of the new technologies exploring the functionality of the tool presented.

Although it is not an exhaustive list of directions, we intend to consider this discussion as a starting point for this instructional model for computer music and live coding through a cloud service solution. This instructional model have the cooperation in favor of the instructor or the learner depending on the situation. The participants may not have the same benefits or goals during a cooperative live coding, but they have to evolve in a helping behavior with or without reward in order to agree in a cooperative live coding environment. An open cooperation movement can also arise from this practice and confront some limits of learning.

VII. Conclusion

Many instructional models have been proposed since the first advances on communication technologies. We have instructors using letters, magazines, calling, videos, and including occasional meetings to the teaching process, or at least using one of these channels for receiving feedback from learners. What is important is the link between the learning material and the learning process to ensure the nature of the distance learning models [19].

In this paper we described a new model of distance education that can make use of the distributed live coding and introduce the cooperative idea. In this model we propose that instructors and learners can share codes and audio synthesis from distant places and also in the same room through the advantages of SuperCopair package. The cloud computing behind the solution allows participants to join a session and interact through the file where the code is being written almost



(a) Windows user

(b) Mac OS user

Fig. 2. Cooperative session with two users. The users share the same file and they can see where the other pair is working following the colors markes visible at the line number column. They can also write comments on the file in order to communicate.

instantly, and without problem with the number of participants. As for example, the free plan used at the cloud service behind SuperCopair accepts up to 20 users, but one can have 10 thousand participants when assigned for a paid plan.

The cooperation of the participants is assumed during the live coding session, although new features can be added in the future in order to add more control from the instructor's point of view. We suggested many opportunities for the practice of cooperative live coding following our experiences, and one can conclude that the results can be gradually presented. During our sessions, the learners became instructors of basic topics in a short time. The interaction through the comments was valuable like a chat with many rooms, as we had many discussions in many portions of the file happening at the same time.

The tool is proposed to SuperCollider, but it can be extended to languages like Csound, Chuck, and Processing. Interpreted programming languages are more suitable for this environment due to the fast output without compilation. However, the idea of cooperative live coding can be used with any programming language, considering that the compilation on the learner side needs to be done depending on the operating system.

The authors wish that the cooperative live coding instigate many instructional models, and that more solutions for interaction between distant learners and instructors emerge from the advances on cloud computing. As the Internet is the only requirement for this practice using , we need to focus the attention on bandwidth, timezone, and system requirements in order to have more cooperative live coding sessions.

References

[1] C. V. A. P. da Silva, C. B. L. Neto, and M. R. Petrucci, "Hipermídias educativas: a aplicabilidade de objetos de aprendizagem em planos de aula nos espaços virtuais," in *IV Encontro Nacional de Hipertexto e Tecnologias Educacionais*. Universidade de Sorocaba, 2011.

[2] M. Silva *et al.*, *Sala de aula interativa*. Quartet, 2000.

[3] Â. Á. C. Dias and H. Chaves Filho, "A gênese sócio-histórica da idéia de interação e interatividade," *Tecnologias na educação e formação de professores*, 2003.

[4] S. M. Hord, "Working together: Cooperation or collaboration?." 1981.

[5] B. J. da Silva Estácio and R. Prikladnicki, "Distributed pair programming: A systematic literature review," *Information and Software Technology*, vol. 63, no. 0, pp. 1 – 10, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584915000476

[6] P. Ford, "Processing processing," in *The Best Software Writing I*. Springer, 2005, pp. 79–93.

[7] N. COLLINS, A. McLEAN, J. ROHRHUBER, and A. WARD, "Live coding in laptop performance," *Organised Sound*, vol. 8, pp. 321–330, 12 2003. [Online]. Available: http://journals.cambridge.org/article_S135577180300030X

[8] N. Collins, "Live coding of consequence," *Leonardo*, vol. 44, no. 3, pp. 207–211, 2011.

[9] T. Magnusson, "Algorithms as scores: Coding live music," *Leonardo Music Journal*, vol. 21, pp. 19–23, 2011.

[10] C. Nilson, "Live coding practice," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, ser. NIME '07. New York, NY, USA: ACM, 2007, pp. 112–117. [Online]. Available: http://doi.acm.org/10.1145/1279740.1279760

[11] G. Weinberg, "The aesthetics, history, and future challenges of interconnected music networks," in *International Computer Music Conference*, 2002, pp. 349–356.

[12] D. Trueman, "Why a laptop orchestra?" *Organised Sound*, vol. 12, no. 02, pp. 171–179, 2007.

[13] G. Wang, N. Bryan, J. Oh, and R. Hamilton, *Stanford laptop orchestra (slork)*, 2009.

[14] I. I. Bukvic, T. Martin, E. Standley, and M. Matthews, "Introducing l2ork: Linux laptop orchestra," in *New Interfaces for Musical Expression*, 2010, pp. 170–173.

[15] J. Rohrhuber, A. de Campo, R. Wieser, J.-K. van Kampen, E. Ho, and H. Hölzl, "Purloined letters and distributed persons," in *Music in the Global Village Conference (Budapest)*, 2007.

[16] A. R. Brown and A. C. Sorensen, "aa-cell in practice: An approach to musical live coding," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 2007, pp. 292–299.

[17] S. Wilson, N. Lorway, R. Coull, K. Vasilakos, and T. Moyers, "Free as in beer: Some explorations into structured improvisation using networked live-coding systems," *Computer Music Journal*, vol. 38, no. 1, pp. 54–64, 2014.

[18] D. Ogborn, "Live coding in a scalable, participatory laptop orchestra," *Computer Music Journal*, vol. 38, no. 1, pp. 17–30, 2014.

[19] D. J. Keegan, "On defining distance education," *Distance Education*, vol. 1, no. 1, pp. 13–36, 1980. [Online]. Available: http://dx.doi.org/10.1080/0158791800010102