Facilitating Creative Exploratory Search with Multiple Networked Audio Devices Using HappyBrackets

Oliver Bown
Interactive Media Lab
UNSW Art & Design
Sydney
NSW, Australia
o.bown@unsw.edu.au

Sam Ferguson
Creativity & Cognition Studios
University of Technology
Sydney
NSW, Australia
samuel.ferguson@uts.edu.au

Liam Bray
Design Lab
University of Sydney
Sydney
NSW, Australia
liam.bray@sydney.edu.au

Angelo Fraietta
Interactive Media Lab
UNSW Art & Design
Sydney
NSW, Australia
a.fraietta@unsw.edu.au

Lian Loke
Design Lab
University of Sydney
Sydney
NSW, Australia
lian.loke@sydney.edu.au

ABSTRACT

We present HappyBrackets, an audio-focused creative coding toolkit for deploying music programs to remote networked devices. It is designed to support efficient creative exploratory search in the context of the Internet of Things (IoT), where one or more devices must be configured, programmed and interact over a network, with applications in digital musical instruments, networked music performance and other digital experiences. Users can easily monitor and hack what multiple devices are doing on the fly, enhancing their ability to perform "exploratory search" in a creative workflow. We present two creative case studies using the system: the creation of a dance performance and the creation of a distributed musical installation. Analysing different activities within the production process, with a particular focus on the trade-off between more creative exploratory tasks and more standard configuring and problem-solving tasks, we show how the system supports creative exploratory search for multiple networked devices, and consider design principles that could advance this support.

Author Keywords

Internet of Musical Things, Creative Coding, Digital Musical Instruments, Multiplicitous Media, Networked Music

CCS Concepts

•Human-centered computing → Ubiquitous computing; *User studies;* **•Applied computing → Sound and music computing;**

1. INTRODUCTION

The focus of this paper is on how the design of toolkits can best support rapid and creatively fruitful coding practices, in the spirit of environments such as Processing, MaxMSP and Scratch, but specifically targeted at the context where *multiple remote media devices* are being programmed. We aim to enable the designers of digital instruments, interactive experiences and performance systems to be able to quickly and easily realise, explore, modify, reconfigure and extend their creative designs. In outlining an approach to designing creativity support tools, Shneiderman et al [14] call for "improved software and user interfaces that empower [designers] to be not only more productive but also more innovative" [14] (p. 62). They observe that well-designed creative productivity tools do "more than merely speed performance of standard tasks, enabling freer exploration of alternatives and a willingness to probe past limits imposed by existing tools" [14] (p. 69). With its greater complexity, the world of multi-device creative Internet of Things (IoT) presents clear challenges to obtaining this goal.

Figure 1: A scene from the dance development workshop with dancers exploring minute movements.

Helping outline a conceptual framework that supports solving these problems, Blackwell, with various collaborators (we refer here to [5]), have developed cognitive dimensions that describe how software design can support or hinder productivity, particularly in creative work. Using an approach inspired by such concepts, this paper presents practice-led creative research into the use of multiple-device Raspberry Pi systems using the HappyBrackets toolkit. We consider how the design of the toolkit and the affordances of the system influence the creative workflow, with reference to Blackwell's framework. In particular, in our case studies, we identify a series of different tasks performed by the creative team, and the factors that influence what task is being per-

NIME'19, June 3-6, 2019, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

formed at any one time. We then examine how the design of the toolkit can support increased time spent engaging in creative exploratory search in a team-working environment.

We explore these themes through the creative development by the HappyBrackets team of two new pieces, one for interaction with dancers and the other for a multi-element musical art installation. We consider what enhances and what blocks creative exploratory search, and how it is steered by the design of the system and the hardware, as well as by the affordances of the medium.

2. BACKGROUND

2.1 Creative exploratory search

Research into creative coding looks at the language, libraries and tools that support the creative production of digital experiences and products using code. Of particular interest is the design of development environments that best support the creative process, the core of which is the ability to explore, reflect and reconfigure creative designs. For example, creativity researchers have examined the concept of exploratory search in different ways. Simonton [15] grounds a theory of exploratory search in the concept of 'blind variation', which provides the basis for a non-reductive comparison between human creativity and natural evolution. He argues that creative tasks necessitate blind search, since "not knowing the path to the solution" is what makes them creative tasks in the first place, but that blind search does not entail a random unstructured process, but can be entirely systematic and structured to maximise results. This idea that creative tasks require structured trial and error, has been central to the creative coding literature [14, 13], where the more rapidly you can see and understand the effects of your actions and design ideas, the more fertile your creative search will be. The most critical feature of any creative tool, therefore, is rapid access to richly informative results from any creative experimentation.

Church et al. ([5], based on earlier work by Blackwell, e.g., [2]) discuss this in terms of support for progressive evaluation, and consider other factors that describe how easily a person can manipulate the system at hand in response to such trials: viscosity (the effort required to effect change); abstraction (how elements are represented with respect to the ability to meta control or batch-control them); and premature commitment (i.e., how locked-in you are to a design direction once you've started on it); and predicting task complexity and risk, and making decisions about switching tasks [5].

Even in the best of creative systems, creative search is constrained by the natural limits to these factors. For example, there is no best abstraction to suit a range of tasks, and any system that has a complex structure requires some degree of lock-in once you start building it, for example which tools or design pattens you use. Therefore all practitioners are used to working within constraints, and some theorists have proposed that constraints are even actively employed to define sparse and therefore manageable creative search spaces [6, 10].

2.2 Creative coding toolkits for networked, tangible devices

Creative coding for networked, tangible devices is gaining interest with the rise and proliferation of physical computing devices. In this paper we are concerned with creative projects that employ multiple standalone small-format computers or microprocessors. The key affordances that distinguish this context from regular computing contexts are: (i) the devices can be made portable and embedded into small interactive objects; (ii) due to the low cost as well as the small form-factor of these devices, multi-device systems can easily be assembled, with multiples in the 100s and 1,000s not being prohibitively unaffordable [3]. In both cases, wireless network communication, primarily over WiFi on a local area network (LAN), provides the basis for easily configured and programmed communication between devices. WiFi networking can be limited in network latency, reliability and data capacity, but suffices for many applications and is very convenient to work with. More generally, in the music domain, this work is part of a field entitled Networked Music Performance [9]. The concept of the Internet of Musical Things [16] has also recently risen to prominence.

In related work, two of the authors have worked extensively with the media arts organisation Squidsoup to produce creative content for large-scale volumetric LED installations [7, 8]. The most recent instance of this work, *Bloom* premiered at the *Christmas at Kew* light festival at Kew Gardens, London, in 2016, and involved 1,000 WiFi enabled Arduino-based devices with, GPS, IMU sensor, LEDs and small piezo speakers, creating a large distributed audio-visual artwork. Due to its scale the work could not be tested in advance, with limited time on-site to test ideas out. These conditions are a strong motivator for the current research into the ease with which creative exploration can be conducted.

Many programmable embedded audio systems, and other types of programmable IoT systems, now exist. For example the Teensy audio library and boards similarly use the Arduino architecture and provide a high quality, low-latency audio programming toolkit. An advantage of Arduino-based systems like Teensy and the custom *Bloom* system are that they are generally lower power compared to Linux-based systems (as used by HappyBrackets). However, when higher-power built-in audio rendering is being used, the power requirement of the audio might balance out this difference. Bela [12] is another system for audio performance for small form-factor Linux-based devices, and the Bela team have developed a number of convenience tools for monitoring and controlling the device. Bela works with BeagleBoard devices and is compatible with a number of audio programming tools, as long as they are adapted to communicate with the Bela's low latency IO system. Bela researchers have explored how environments such as Bela support divergent creative outcomes through their open-ended design potential [1].

A vast number of tools exist to speed up creative search in interactive music contexts. For example, tools such as libmapper [11] and Osculator (https://osculator.net/) make connectivity and mapping between sensor streams and outputs easy across different software environments. Yet relatively little research has been conducted into how creative efficiency can be maximised in dynamic, networked, multi-device contexts, which is the target of this research.

3. HAPPYBRACKETS

HappyBrackets is a coding toolkit that targets Raspberry Pi computers to make networks of distributed interactive audio device's (DIADs, described in [4]); portable sensor-enabled sonic devices. A typical DIAD unit consists of a Raspberry Pi computer, sensor HAT, mini-speaker and battery. Recently we have started working with the Raspberry Pi Zero and HiFiBerry MiniAmp board, and now our own board which combines power regulator, 3W amplifier, LED, IMU sensor and Grove expansion ports. DIADs can be housed in different device housing designs, but we have been mainly focused on making standalone sonic balls that can be used as instruments, toys, elements in interactive sonic games, and props for performance and dance. Currently the HappyBrackets system is the only system we know of which allows

immediate push deployment over WiFi to multiple running Raspberry Pi (or other Linux) devices. This capability has been key to enabling rapid creative exploration with multiple devices. The current key design features of HappyBrackets include the following capabilities:

- **General purpose language and widely supported IDE**
  - Uses the general purpose and widely supported programming language Java;
  - Incorporated into and makes full use of IntelliJ IDEA development environment, with advanced IDE features;
- **Interface design, Communication and Debugging**
  - Tools to set up GUI controls to remote control multiple devices, including shared variables across devices;
  - Tools to manage, monitor and debug devices remotely over a network;
  - Convenient libraries for network communication between the controller and devices;
- **Inputs and Outputs – Convenient sensor library and creative audio toolchain**
  - Convenient sensor library allowing rapid access to sensors via the Raspberry Pi's GPIO and the PI4J library;
  - Access to the Grove library of arbitrary sensors through the GrovePi add-on board and associated java library;
  - Key synthesis, sample and granular sample playback tools, alongside a suite of audio effects;
  - Generative music libraries for musical control of synthesis;
- **Device Interaction – Inter-device communication and time synchronisation**
  - Convenient libraries for network communication between devices, including shared variables across devices;
  - Tight synchronisation of clocks between devices using NTP over a local network;
- **Live Coding – on-the-fly iteration and updates**
  - Simplified setup process for configuring and working with multiple devices, getting up and running with live coding quickly;
  - Enables live coding audio programs to run on remote networked devices, live compile and send software compositions to devices, run multiple compositions simultaneously, and live update compositions on the fly;

4. TASK ANALYSIS FRAMEWORK

"Multiplicitous media" systems [3] are digital systems consisting of multiple hardware devices connected over a network. Unlike other digital creative contexts, we find that such systems impose a greater demand for the production of code in specific situations where hardware is set up in context (i.e., it cannot always be performed in advance because the ideas must be tested in the final context). For example, when experimenting with dancers using our interactive audio devices, it was necessary to be able to rapidly experiment with different system designs whilst the dancers were on site, performing with the devices. In this context, it is common for the dancers to be waiting around whilst the coders solve problems, but preferable that the coders are making tangible and ongoing contributions to the creative development of the work. When working with multiple devices, it can be slow to deploy code updates to the devices, and deal with hardware connectivity and configuration issues, causing large, creatively debilitating delays.

We therefore identify in our practice a series of different activities that coders may be engaged in during such a creative session (Table 1), and analyse how the design of a creative coding toolkit can help coders focus on those activities that are most productive in time-critical creative phases. The activities we identify are: (i) setting up devices, (ii) connecting to devices, (iii) in-depth coding (developing software code), (iv) debugging and troubleshooting, (v) rapid creative coding (exploring software code), (vi) exploring and testing interaction (e.g., physically interacting with devices), (vii) remote-controlling and remote-monitoring devices via program and state variables, (viii) discussing and planning. We identify items (v) through (viii) as the tasks that we want to focus on in a creative time-sensitive session, minimising time spent on tasks (i) through (iv). The distinction between tasks (iii) and (v) is significant and will be elucidated in the following discussion. As a rule of thumb, task (iii) relates to more involved heads-down coding work that requires planning and concentration, whereas task (v) is rapid and can be done without much thought and planning. Following the discussion of case studies we offer insights into the relation between tasks (iii) and (v) in creative strategies.

Table 1: Tasks in a collaborative creative process.

Task	Context	Participants
(i) setting up devices	hardware setup	coder
(ii) connecting to devices	computer	coder
(iii) in-depth coding	computer	coder
(iv) debugging and troubleshooting	computer	coder
(v) rapid creative coding	creative time-sensitive session	coder+artists
(vi) exploring and testing interaction	creative time-sensitive session	coder+artists
(vii) remote-controlling and remote-monitoring devices	creative time-sensitive session	coder+artists
(viii) discussing and planning	creative time-sensitive session	coder+artists

5. STUDIES

The workflow of creating works with HappyBrackets was studied in two creative development contexts involving the authors. These sessions were documented and analysed to examine the progression through different coder activities during the creative sessions, and the associated factors involved in the distribution of these activities. In each of these cases, the context is given, the creative process and outcomes are discussed, and the task analysis is presented in an associated diagram.

5.1 Development of a Dance Performance

A creative study undertaken by the authors (the HappyBrackets developers and Lian Loke acting in the role of dancer and interaction designer) and another dancer, Kirsten Packham, resulted in a performance, *So Predictable!?*, at the *NOW now Festival* in Sydney in January 2018 and the NIME'18 conference in Virginia in June 2018. For this dance performance, two DIADs were made, housed in soft spherical enclosures, designed by industrial designers *Vert* [1].

The soft spherical designs afforded the possibility to roll and spin the devices, and leave them to rock or wobble on their own, producing specific sensor data patterns that were not present when interacting only through hand-contact with

[1]Video documentation of sections of this creative development session can be found at https://tinyurl.com/ycg2bhcr.

the devices. Given that the potential impact of these physical affordances on the interactive audio design were unknown, the creative development of the piece was grounded in being able to support exploratory search on-site during rehearsal.

A small handful of basic sketches were developed prior to the creative session to explore core interaction paradigms. These were divided into a basic suite of direct mappings from sensor data: linear and non-linear scaling; threshold-based triggers; and discrete state detectors (e.g., is the device pointing up, down, left, right, rolling, spinning, in free-fall, etc.). Other basic mapping functions such as smoothing and trigonometric operations were also prepared for convenience. At the beginning of the session, the coder deployed the initial code sketches to two separate devices simultaneously. The dancers played with the devices, taking some time to become familiar with the interactive sonic behaviour in each case.

After a period of exploration of the pre-prepared compositions the team agreed that threshold-based and orientation-based triggers were of limited interest because they lacked sensitivity, involving only occasional events. Sketches implementing immediate mappings from sensor values to sonic outputs became the focus, and in turn, the dancers became interested in small movements of the device as the motion-to-sound mappings enabled a sensitivity to micro-movement and an interest in improvising on the edge of silence/sound and stillness/motion. A period of more intense co-creative exploration took place around this theme, where the coder manipulated data parameters on-the-fly via wireless data communication to the device, for example, manually changing numbers in the code to alter specific frequencies and amplitudes, and explicitly reconfiguring the system behaviour and synthesis architecture.

In response to the coder's live manipulations, the dancers gave feedback in three forms: a) mark this as interesting, but maintain flow; b) stop, mark as interesting, discuss directions this could go; and c) bored/don't like, request something new. For each of these alternative paths, the coder responded in different ways: a) note good settings, but continue to manipulate parameters in such a way as to be able to go back, either through comments in the code, or where necessary making a copy of the sketch file; b) either engage in a more detailed refactor on the fly (while the dancers take a break), or make notes in code for later refactoring (to be prepared for the next session); and c) pull up another pre-coded sketch or make more radical changes to the current sketch code.

Through this exploration the team rapidly identified relevant thresholds and parameter settings that suited the micro movements being explored by the dancer. In addition, mappings that involved delayed responses and longer-term interaction effects were explored and found to be highly impactful, by allowing a more dialogical interaction between dancer and device. A new composition was coded on the fly, that included a simple hidden state variable that tracked the absolute magnitude of the gyroscope's 3-axis output, with smoothing. This variable was designed to increase rapidly towards the gyro magnitude, but to drop back down slowly, similar to how different attack and release values of an audio compressor can create interesting dynamic effects.

Variations of this delayed variable were used to control LFO modulation depth and rate of a simple synth, as well as the overall volume. Sustained activity by the dancer would work the device gradually into frenzied sound. After ceasing activity, the sound would linger for several seconds before dying down again, with unpredictable pathways back to silence. This stood out as being immediately engaging for the dancers, especially when they interacted with the devices with minuscule movements. In addition, an initially

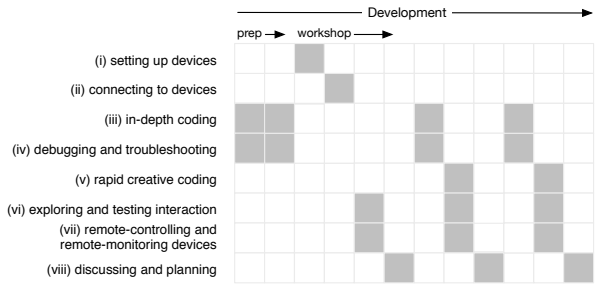

Figure 2: A schematic view of switching between tasks, during the creative development of a dance performance.

unidentified divide-by-zero bug caused the synthesis values to explode after the device was left completely still, resulting in bizarre bursts of noise that only arose after a period of inactivity. This provided a new unexpected basis for interaction and became part of the performance.

To assist with describing the creative development process, a schematic view of switching between the various tasks in this process is provided in Figure 2. In summary, in-depth development took place before the session (tasks (iii) and (iv)). After some setup (tasks (i) and (ii)), the creative session began by the coders and dancers exploring and testing existing candidates (tasks (vi) and (vii)), after which the coders rapidly developed a new program sketch based on time-delayed variables, within the session. This was an involved coding task (iii) and required some debugging (iv), but the program design was kept simple, and rapidly the activity progressed to more creative search again (v, vi and vii). A bug was discovered, but importantly in the context of this creative exploratory task, no debugging was undertaken; instead the bug became part of the creative design. From the coder's perspective, the transition from the (iii) task to the (v) and (vii) tasks was primarily achieved by identifying numeric variables that could be manually explored. This could be done directly in code (v) or using a feature that allowed the coder to rapidly set up remote-controllable variables (vii) manipulated by GUI sliders that could then be further explored in interaction. The latter feature was predicted to be valuable for rapid exploration, but in reality the potential to directly manipulate numbers in the code was more commonly used, as the latter feature took time to set up, added more clutter to code and could complicate the interaction.

5.2 A Distributed Musical Installation

The second case study describes the creative development of a non-interactive, networked musical installation, *Spiral*, at the Powerhouse Museum in 2019. There were no performers, hence the creative development is focused on the work of the creative coders, although a composer, Adrian Lim-Klumpes, was also involved in the creative development, giving feedback to the coders. In this installation, 25 self-contained DIADs were built and hung from the ceiling, communicating with four other elements in the musical work: a Disklavier piano, mechanically actuated drums, a series of cello samples, and an audio-based augmented reality (AR) experience. The Disklavier, drums and cello samples were run directly from a computer running Ableton Live. A Max for Live device within the Ableton Live set sent network broadcast messages to the DIADs and the AR devices. The audio program ran continually at a fixed tempo of 140BPM.

The DIADs were programmed so as to maintain their own

time-synchronised clocks, maintaining time with the Ableton Live session via a local network time protocol (NTP) server, which in our experience kept time to within 20ms. The Ableton Live session then broadcast regular âĂIJ/beatâĂİ messages with a beat number and Unix timestamp attached to each message. A segment of the DIAD code responded to those messages in order to keep the clock in synch, using the known fixed tempo in order to plan the time of the next beat. It was noted that all of the computers' system clocks were prone to drift, so this synchronisation was essential.

The DIADs were then set up with the 25 devices arranged in a spiral, and assigned numbers based on their positions. This made use of a feature in the IDE plugin that allowed a list of known devices to be stored by the controller computer and assigned IDs upon connection. Thus the same code could then be sent to all devices, but with branch points or different actions determined based on the device's ID through switch statements and numeric expressions. In a similar way to the dance scenario, the install hardware, involving Disklavier, mechanical drums and various speakers, was only available at install time, and it was essential to be able to make adjustments to the code of all devices rapidly at this time. However, unlike the dance development, where code was developed on the fly, the code composition was a much more advanced system with complex programmed behaviour. Here the tweaking phase was focused on getting a good sound and making minor adjustments to the composition.

Of particular value in this process was the implementation in HappyBrackets of a simple OSC message broadcast interface, built into the IntelliJ plugin, where the composer could enter and send custom messages and try out different states across all devices. This feature, we found, encouraged us to parameterise elements of the code more early on, knowing that we could then rapidly play with variable values before locking them in, but with the added advantage that our code remained neater and more modular. This included debugging features like turning on a back click on the devices to ensure they were playing in time. In other words, the environment was able to encourage tasks (v) and (vii) (there was no physical interaction in this system, but here task (vi) refers to listening to the system playing in context). In addition to the custom OSC message interface, the remote GUI controls enabled by the HappyBrackets toolkit were used to the same effect, allowing us to vary values across multiple devices from a laptop computer. As well as debugging features, such as turning on and off a click to check timing, we used these features to experiment with tunings, rates, levels of randomness, delays between devices, filter frequency values and other features, before settling on something that was satisfying and suited to the space. For example, the graphical interface was used to tweak a notch filter across individual devices within the acoustic space of the install, for fine acoustic tuning.

A schematic view of switching between tasks is provided in Figure 3. In summary, the pattern followed the same process of moving between cycles of rapid creative development (tasks (v) and (vii)), testing (task (vi)), reflection (task (viii)) and more sustained in-depth development (tasks (iii) and (iv)), in which relevant parameterisations were made that would be useful to speed up the next round of rapid development.

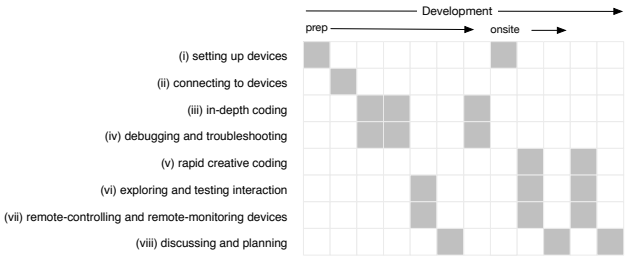

Figure 3: A schematic view of switching between tasks, during the *Spiral* installation development.

6. DISCUSSION

6.1 Task switching

From the two task diagrams (Figures 2 and 3) we see a similar pattern that is familiar from an iterative design perspective (Figure 4): a pair of nested iterations, one inner iteration between rapid coding, parameter tweaking and experimentation, and an outer iteration between this inner iteration and more involved coding stages (sometimes happening on-site, but more often happening at a seperate time to the test sessions). We note two factors in the transitions between the elements in the outer iteration that are of interest from the design of creativity support tools.

Firstly, during in-depth development sessions it is possible for the developer to set up a series of parameters and controls that facilitate more rapid exploration on-site, when transitioning to the rapid prototyping and testing phase. In HappyBrackets we introduced features to manage this process: the ability to easily respond to custom OSC messages sent from a command prompt on the controller computer, and the ability to open up a GUI panel on the controller computer, with GUI elements bound to the devices (both global and device specific). Adding these controls involves some additional overhead and planning during the development session, and importantly, introduces additional code clutter, so although it can be very useful to speed up the creative development sessions there are also reasons not to set up these elements in advance. In comparison, given that sending live changes to the code to multiple devices is extremely easy in HappyBrackets, the need to add such parameters can be alleviated altogether, simply by sending over a new block of code, and we observed that in certain cases it was preferable simply to edit the code and re-send it. The latter is more amenable to open-ended exploration because it does not pre-empt in any way what kinds of parameters are needed. Conditions which may affect how well the code is set up in advance with control parameters include: how refined the design is; and whether the parameters are things that really need to be explored in a rapid interactive manner (an example of which is when performing a filter sweep to find a resonant frequency in a space, as in *Spiral*).

Secondly, during the rapid prototyping and test phase, it can be hard to decide when to commit time to refactoring code (either to generalise it to allow for more variation, or simply to try out a different configuration), given that this demands a shift back into more in-depth coding. The tendency is to make incremental, minor changes (which over time might evolve towards something very different, possibly messy). This impacts how one might then go and engage in more in-depth refactoring later, and also creates situations in which good solutions might get lost and possible branch points missed.

In both cases, whilst HappyBrackets features helped speed up prototyping and enhance exploration, better guidelines are still needed that outline exactly how and when the developer commits to specific coding decisions and designs. In future iterations we will consider additional features in response to this ongoing challenge: making in-code parameters *automatically* exposed to a control surface without having

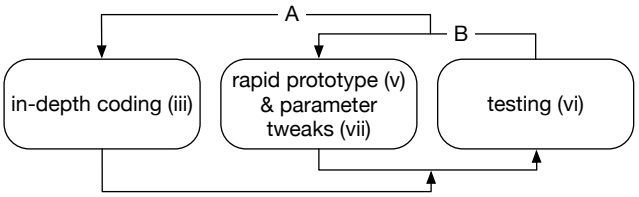

Figure 4: Iterative cycles in the creative process.

to edit the code, which would include being able to store successful parameter values as defaults back into the code; advancing the API to provide objects that are positioned at the correct level to maximise breadth of exploration (e.g., modular synthesisers do this relatively well, many programming libraries are too low level, MIDI instruments are too high level); and providing rapid interfaces for exploratory reconfigurations, such as a simple matrix mapping interface.

6.2 Minimising device configuration tasks

Working with multiple remote devices increases the overhead of tasks that are distracting to productive creative development: setting up devices (task (i)) and connecting to devices (task (ii)), which includes monitoring connections and tracking which device is which in a collection. One of the most productive things a toolkit can do then is minimise the time cost of these tasks. Although we did not take detailed measurements of the time overhead of these tasks in the current case studies, we estimate that anywhere between 10-30% of time was devoted to issues with setup and connectivity. Features of HappyBrackets designed to ease these tasks include having a pre-established network that devices already know to connect to, autodiscovery of devices, being able to ping devices from the controller computer (make a bleep on the device), being able to query devices for hostname and assigned ID, and sensor data streams. It is also possible to directly configure devices substantially via code, such as setting hostnames and boot options. A recent development includes being able to directly flash audio files and code libraries to devices via the IntelliJ coding interface. The difficulty of sending audio files to devices previously slowed exploration of different sample options considerably. From our experience in these practice-based studies we consider this set of features to constitute an essential baseline for functionally working with multiplicities.

7. CONCLUSION

In this paper we have examined how creators working with the HappyBrackets toolkit for programming audio on multiple remote devices switch between tasks and manage creative exploration in an iterative design cycle. We have considered how the features of HappyBrackets support different tasks and the progression between tasks, and how these could be improved, through revised design guidelines to improve this process further.

8. ACKNOWLEDGEMENTS

We acknowledge additional programming contributions by Sam Gillespie and Oliver Coleman, financial support by internal grants (from the University of Sydney, with Phil Poronnik, and a Faculty Research Grant from UNSW), dance development by Kirsten Packham, industrial design by Vert Designs, Spiral artwork commission by MAAS, and composition by Adrian Lim-Klumpes and Tangents.

9. REFERENCES

[1] J. Armitage and A. McPherson. Crafting digital musical instruments: An exploratory workshop study. In *Proc. NIME'18*, Blacksburg, Virginia, June 2018.

[2] A. Blackwell and T. Green. Notational systems–the cognitive dimensions of notations framework. *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science. Morgan Kaufmann*, 2003.

[3] O. Bown and S. Ferguson. Creative media+ the internet of things= media multiplicities. *Leonardo*, (Early Access):53–54, 2017.

[4] O. Bown, L. Loke, S. Ferguson, and D. Reinhardt. Distributed interactive audio devices: Creative strategies and audience responses to novel musical interaction scenarios. In *Proceedings of the 2015 International Symposium on Electronic Art, Vancouver, Canada*, 2015.

[5] L. Church, M. Marasoiu, and A. Blackwell. Sintr: Experimenting with liveness at scale. In *Proceedings of ECOOP 2016*, 2016.

[6] P. Dalsgaard and K. Halskov. Designing urban media façades: cases and challenges. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2277–2286. ACM, 2010.

[7] S. Ferguson, A. Rowe, O. Bown, L. Birtles, and C. Bennewith. Networked pixels: Strategies for building visual and auditory images with distributed independent devices. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pages 299–308. ACM, 2017.

[8] S. Ferguson, A. Rowe, O. Bown, L. Birtles, and C. Bennewith. Sound design for a system of 1000 distributed independent audio-visual devices. In *In proceedings of NIME 2017*, Aalborg University Copenhagen, Denmark., 2017.

[9] L. Gabrielli and S. Squartini. Wireless networked music performance. In *Wireless Networked Music Performance*, pages 53–92. Springer, 2016.

[10] T. Magnusson. Designing constraints: Composing and performing with digital musical systems. *Computer Music Journal*, 34(4):62–73, 2010.

[11] J. Malloch, S. Sinclair, and M. M. Wanderley. Libmapper:(a library for connecting things). In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 3087–3090. ACM, 2013.

[12] A. P. McPherson, A. Chamberlain, A. Hazzard, S. McGrath, and S. Benford. Designing for exploratory play with a hackable digital musical instrument. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 1233–1245. ACM, 2016.

[13] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. Design principles for tools to support creative thinking. 2005.

[14] B. Shneiderman, G. Fischer, M. Czerwinski, M. Resnick, B. Myers, L. Candy, E. Edmonds, M. Eisenberg, E. Giaccardi, T. Hewett, et al. Creativity support tools: Report from a US national science foundation sponsored workshop. *Int Journal of Human-Computer Interaction*, 20(2):61–77, 2006.

[15] D. K. Simonton. Creativity and discovery as blind variation: Campbell's (1960) BVSR model after the half-century mark. *Review of General Psychology*, 15(2):158, 2011.

[16] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. Internet of musical things: Vision and challenges. *IEEE Access*, 6:61994–62017, 2018.