**Computer mediated music
production:
A study of abstraction and
activity**

by

Matthew Duignan

A thesis
for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2008

Abstract

Human Computer Interaction research has a unique challenge in understanding the activity systems of creative professionals, and designing the user-interfaces to support their work. In these activities, the user is involved in the process of building and editing complex digital artefacts through a process of continued refinement, as is seen in computer aided architecture, design, animation, movie-making, 3D modelling, interactive media (such as shockwave-flash), as well as audio and music production. This thesis examines the ways in which abstraction mechanisms present in music production systems interplay with producers' activity through a collective case study of seventeen professional producers. From the basis of detailed observations and interviews we examine common abstractions provided by the ubiquitous multitrack-mixing metaphor and present design implications for future systems.

Acknowledgements

I would like to thank my supervisors Robert Biddle and James Noble for their endless hours of guidance and feedback during this process, and most of all for allowing me to choose such a fun project. Michael Norris and Lissa Meridan from the Victoria University music department were also invaluable for their comments and expertise. I would also like to thank Alan Blackwell for taking the time to discuss my work and provide valuable advice. I am indebted to all of my participants for the great deal of time they selflessly offered, and the deep insights they shared into their professional world. Thank you to all of the members of the Elvis research group for the support and motivation your provided, particularly to Pippin and Rilla who helped me carry this thesis to completion. My thanks go to the members of the Carleton HOT Lab for hosting me so generously in Canada, and to Judy in particular for sharing your knowledge of activity theory. The members of my band Attik — Mike, Christian, Duncan, and Rob; you fuelled me with the passion and motivation to undertake this research project. Thank you for encouraging me to experiment with new technologies in rehearsals, and for understanding my move to Canada. Additional thanks to Duncan from Experimenta Ltd. for providing your experience and examples from your graphic design work. Thanks to Richard Procter for taking such an active interest in my work and for our long conversations. My thanks go out to my family for your love, and for creating in me the thirst for learning that has taken me this far. Lastly, thank you to Juls for your unwavering care and love.

This research was made possible through the assistance of Victoria University postgraduate scholarships, and a Royal Society of New Zealand travel grant.

Contents

1	Introduction	1
2	Background	7
2.1	Introduction	7
2.2	Computer mediated music production	8
2.2.1	Recording, reproduction and synthesis of sound	9
2.2.2	Capture and reproduction of musical gesture	11
2.2.3	Editing audio and gesture	11
2.2.4	The producer	12
2.2.5	The Digital Audio Workstation	13
2.3	Human Computer Interaction	16
2.4	Digital craft and notational systems	17
2.5	Abstraction	19
2.5.1	Types of abstraction	21
2.5.2	Conceptual models	24
2.5.3	The cognitive dimensions of notations	26
2.6	Activity theory	28
2.6.1	Activity theory in HCI	30
2.7	User-interface metaphor	32
2.7.1	Usability and metaphor	34
2.8	Summary	38

3	Models and Abstractions for Music	39
3.1	Introduction	39
3.2	Models of music	40
3.2.1	Psychological perception of music	41
3.2.2	Common Music Notation and theory	45
3.2.3	Digital representations of music	47
3.2.4	Representations in software for experimental and gen- erative music	51
3.3	The multitrack-mixing metaphor	51
3.3.1	Multitrack tape recorders and mixing consoles	52
3.3.2	From physical hardware to user-interface metaphor	56
3.3.3	Multitrack-mixing model critique	64
3.3.4	Abstraction mechanisms atop the multitrack-mixing model	66
3.4	Summary	72
4	Methodology	75
4.1	Introduction	75
4.2	An introduction to case study	76
4.3	The researcher	78
4.4	Research tools	80
4.4.1	Activity theory	80
4.4.2	Cognitive dimensions	86
4.5	Data collection and analysis	89
4.5.1	Ethical considerations	89
4.5.2	Data collection and recording procedure	89
4.5.3	Data analysis procedure	93
4.5.4	Case selection and participants	96
5	Activity Analysis	103
5.1	Introduction	103
5.2	Computer mediated music production	104

5.3	Contradictions	105
5.4	Subject, tool, and object	108
5.5	The subject and their roles	113
5.6	Activity hierarchy analysis	118
5.6.1	Actions to generate material	120
5.6.2	Actions to refine material	122
5.6.3	Actions to arrange material	123
5.6.4	Actions for live performance	123
5.6.5	Actions for housekeeping	125
5.7	Internalisation and externalisation	126
5.8	Abstraction and activity	128
5.8.1	The four aspects of abstraction	129
5.8.2	Rendering considered harmful	132
5.8.3	Option dilemma	134
5.8.4	Conceptual burden	136
5.9	Summary	137
6	Process Abstraction	139
6.1	Introduction	139
6.2	Processing power	140
6.3	Viscosity caused by processing	142
6.3.1	Moving on the time-line	142
6.3.2	Moving to non-processed context	146
6.4	One-off effects	147
6.4.1	Effect on a single track	147
6.4.2	Temporary diversion of shared effects	149
6.5	Editing effected results	149
6.6	Rendering for live performance	151
6.7	Hiding edits	152
6.8	Real-time processing	154
6.9	Summary	155

7	Voice Abstraction	157
7.1	Introduction	157
7.1.1	Stream perception	158
7.2	Voice duration	160
7.3	Organising voices	161
7.4	Amalgamating voices	162
7.5	Separating grouped material	173
7.6	Grouped voices and context	175
7.7	Summary	177
8	Temporal Abstraction	179
8.1	Introduction	179
8.2	Hierarchical temporal grouping	179
8.2.1	Ambiguous temporal structure	188
8.3	Linearisation	189
8.3.1	Forming arrangements	190
8.3.2	Material in arrangement context	192
8.3.3	Flexibility	195
8.3.4	Lead-ins	198
8.4	Rhythmic structure	200
8.4.1	Tempo and Time Signatures	200
8.4.2	Rhythmic effects	203
8.4.3	Event and metrical structure relationships	204
8.5	The editing / performance divide	209
8.5.1	Performance gesture	210
8.5.2	Editing notation	212
8.6	Summary	214
9	Reuse and Versioning Abstraction	217
9.1	Introduction	217
9.2	Copying versus referencing	218
9.2.1	Copying	218

9.2.2	Referencing	221
9.3	Variation	223
9.4	History	226
9.4.1	Versioning history	227
9.4.2	Take management	232
9.4.3	Material history	233
9.4.4	Dependencies	236
9.5	Library management	237
9.5.1	Archiving previous work	237
9.5.2	Locating material	239
9.5.3	Local libraries and working space	243
9.6	Summary	247
10	Evaluation and Discussion	249
10.1	Introduction	249
10.2	Evaluation of systems	249
10.2.1	Apple Logic 7	250
10.2.2	Ableton Live 6	257
10.3	Implications for design	265
10.3.1	Designing for processing abstraction	265
10.3.2	Designing for voice abstraction	270
10.3.3	Designing for temporal abstraction	273
10.3.4	Designing for reuse and versioning abstraction	276
10.4	Wider applications	282
11	Conclusion	291
11.1	Thesis contributions	292
11.2	Generalisability	295
11.3	Future work	297
	Appendices	298

A	Activity Checklist to Interview	299
A.1	Checklist critique	299
A.2	Developing the activity interview	301
A.3	Activity interview in practice	304
A.4	The unified activity checklist	312
B	Activity Interview	317
C	Evaluation Questions	321
D	Additional sources for figures	327

List of Figures

2.1	Recording technologies mediating performers and audience. . .	9
2.2	The Moog synthesiser.	10
2.3	The Korg SQ-10 analogue sequencer.	11
2.4	The Fairlight CMI sampler-sequencer.	14
2.5	The Fairlight CMI's sequencer user-interface	15
2.6	3-dimensional facial models.	21
2.7	3-dimensional facial animation abstractions.	22
2.8	Design, User, and System Image models	25
2.9	Folder and trash can user-interface metaphor.	33
2.10	Metaphor in volume controllers.	34
2.11	Metaphor in the combinator.	35
2.12	Metaphor in the virtual MIDI keyboard.	36
3.1	Abstraction in Common Music Notation.	48
3.2	The multitrack-mixing metaphor.	52
3.3	A multitrack tape recorder.	53
3.4	An audio mixing desk.	54
3.5	The Pro Tools mixing console.	57
3.6	The Logic mixing console.	57
3.7	The Cubase mixing console.	58
3.8	The Logic transport controls.	59
3.9	The Logic arrange page.	60
3.10	The Cubase project window.	61

3.11	Notator's main screen.	62
3.12	Ableton Live's arrange view.	62
3.13	Ableton Live's mixer.	63
3.14	Three effect units from Live.	67
3.15	An audio effect rack from Live.	67
3.16	Action layers from Soundtrack Pro.	68
3.17	Live's session view.	69
3.18	Lanes in Cubase.	71
3.19	Freezing audio in Live.	73
4.1	Activity checklist to interview transformation matrix.	85
4.2	A software tool for interview analysis.	94
5.1	Activity triangle for music production.	109
5.2	Activity hierarchy for music production.	119
5.3	Activity and abstraction.	129
8.1	Aggregation of audio in Live.	180
8.2	A MIDI keyboard overlay.	182
8.3	A customised MIDI keyboard.	182
8.4	A compilation of a 'megalousal' Live set.	187
8.5	A passage from Mozart's Sonata in A K. 331.	188
8.6	The Notator event list editor.	208
8.7	The Logic piano-roll.	210
10.1	Apple Logic.	250
10.2	Ableton Live.	258
10.3	A prototype for processing abstraction.	267
10.4	A prototype for v-rendered audio.	268
10.5	A prototype for editing v-rendered audio.	268
10.6	Animations for voice abstraction.	272
10.7	A prototype for metrical abstraction.	276
10.8	A prototype for reuse and versioning abstraction.	278

10.9 The track displays from iMovie 7 & 8.	284
10.10The DoView model editor.	285
10.11The DoView slice list.	286
10.12A DoView user's filesystem.	287
10.13A graphic designer's file-system.	288
10.14A graphic designer's Photoshop layers pallet.	289

Chapter 1

Introduction

Much of the music we listen to today is recorded and manipulated with computer systems. In many cases, these computer systems are playing an increasingly large role in mediating the entire creative process rather than simply acting as a recording and post-production technology. This thesis examines activity systems that are towards the computer mediated end of this spectrum, where the entire development process from initial composition and concept formation through to the arrangement, mixing, and performance are heavily informed by computer music systems. This thesis uncovers the activity of professional ‘producers’ who use these sophisticated computer interfaces to create contemporary popular music, and examines the abstraction mechanisms these user interfaces provide that both help and hinder their activity from a Human Computer Interaction perspective.

Human Computer Interaction (HCI) research has a unique challenge in understanding the activity systems of all creative professionals, and designing the user-interfaces to support their work. In these activities, the user is involved in the process of building and editing complex digital artefacts [100] through a process of continued refinement; as is seen in computer aided architecture, design, animation, movie-making, 3D modelling, interactive media (such as shockwave-flash), as well as audio and music production. In digital artefact creation, progress and goals are often relatively undefined and unfold

in response to the artefact as it takes form, typically through the medium of a graphical notation system. This hinders traditional task analysis and user-interface design techniques. The complexity and structure of the developing digital artefact creates many interesting user-interface analysis and design challenges, the most pronounced of which being the discovery of how can users manage this complexity and still remain creative and in control. In order to create such complex digital artefacts in a professional context, the user-interfaces also tend to contain rich and deep functionality, exhibiting their own inherent complexity.

As the philosopher John Locke described in *An Essay on Human Understanding* [98], a principle way in which the human mind deals with complexity is through the process of abstraction: that of creating and manipulating higher level concepts through patterns and generalisations:

“In this faculty of repeating and joining together its ideas, the mind has great power in varying and multiplying the objects of its thoughts” — John Locke, 1690 [98]

This thesis examines the ways in which abstraction mechanisms present in music production systems interplay with the activity of professional music producers. The notational systems employed in music production software are an exemplar of digital artefact creation systems used by creative professionals, making them a rich and complex domain for HCI research. The study of user-interfaces for music creation is also interesting in its own right, with their multi-modal character, collaborative usage, and mature state of development that provide a solid platform for examining the state of the art. Additionally, the design of music production tools is predominantly based on pre-existing machines, and thus exhibits extensive use of ‘user-interface metaphor’ in which the software is conceptually and graphically organised around the same design as some other (typically physical) system. In the case of music production software, their design has been tightly informed by traditional music hardware systems, and in particular the multitrack-mixing

model and its inherent design limitations. Our limited understanding of the abstraction mechanisms allowed by this particular metaphor, and how it interacts with the modern computer based music production activity, provides us with motivation to study these systems also.

The purpose of this research is to learn about the fitness and difficulties of existing music production abstraction mechanisms in relation to the work of producers. This will lead us to a better understanding of how to design abstraction mechanisms, both in music production tools, and for other digital artefact crafting systems. This motivates our central research question:

How is the work of professional music producers affected by the particular abstraction mechanisms in the user-interfaces of music production software?

This thesis presents three major contributions to the field, stemming from detailed interviews and observations which we conducted with seventeen professional producers.

- The first contribution of this thesis lies in our findings, resulting from our principled analysis of four classes of music production abstraction that were uncovered through the fieldwork component of this research. These findings represent a great advance in our understanding of the strengths and weaknesses of existing abstraction mechanisms, and more importantly reveal pressing evidence for the provision of specific new abstraction mechanisms in these systems. Moreover, the methodology and analysis presented here provide a compelling template for application to the many other forms of computer mediated activity in which creative professionals craft digital artefacts.
- Secondly, this thesis presents our activity theory based analysis of participants' music production activity. As the first large-scale activity theory analysis of computer mediated music making based on a detailed

field study, we see the utility of the activity theoretic approach. Additionally, we provide a useful foundation for understanding this type of creative production that had not been examined in such detail previously. This analysis also addresses the current lack of understanding of how commonplace composition and other music production activities are carried out with modern computer tools.

- In our third contribution we present a concrete set of evaluation criteria to act as a tool for the understanding and design of both existing and new systems. Our new evaluation resource is the direct result of our field research findings, and are explicitly derived from specific discussions of observations and interview quotes in our write up. The new evaluation questions can be found in appendix C and can be applied in future to ensure the findings of this research impact the abstraction design in the development of forthcoming music production systems. We demonstrate their utility by applying them to two music production systems, and then present a set of design suggestions and prototypes to address some of the ways in which our extensive fieldwork found current abstraction mechanisms lacking. These new designs flow directly from the analysis and evaluation of music software tools, made possible by our application of the evaluation criteria described previously. Some of these design suggestions would require wholesale redesign of music production systems, while others could be added to the abstraction mechanisms already present in existing software tools.

Additionally, we present the methodological contribution of a new interview script based on the theoretically rich but terminology dominated activity theory checklist [85]. While the checklist does provide a solid basis for understanding human work from the perspective of activity theory, our novel activity interview script provides a more practical and rapidly deployable resource for practitioners not steeped in the activity theory literature. Even without detailed theoretical knowledge, our interview resource allows

practitioners to take advantage of the rich frame of analysis that activity theory provides. This activity interview resource was tested through its use as part of our extensive collective case study with twelve of our professional music producers, and is provided in appendix B.

The organisation of this thesis is as follows:

- Chapter 2: covers essential background. We introduce the domain of computer mediated music production, HCI's study of notational systems, the basis of abstraction in these systems, and introduce user-interface metaphor.
- Chapter 3: introduces various models of music and the abstractions they contain. We detail relevant aspects of the psychological basis of audio perception, the historical development of music theory and common score notation, and digital music representations. We outline the history of the multitrack-mixing metaphor, and the abstractions that have been developed since.
- Chapter 4: explains the methodology we employed in our collective case study and analysis, and introduces our participants. We introduce the cognitive dimensions of notations questionnaire, and describe the systematic process we used to create our new activity interview resource.
- Chapter 5: serves as our "task analysis" by presenting our analysis of our participants activity systems through the lens of activity theory. This chapter also provides concrete examples of how producers described their own work, and as they were observed working in their production studios.
- Chapters 6-9: expose in detail the ways in which abstraction mechanisms in participants' tools were observed and described to be interacting with their real world activity. Each chapter covers one major

class of abstraction (processing, voice, temporal, as well as reuse and versioning abstractions), and our comprehensive analysis of the data from the field.

- Chapter 10: presents a final high level evaluation of the significant aspects of two popular music production tools using our evaluation criteria developed through the proceeding chapters. The chapter then outlines potential user-interface designs to address the abstraction issues uncovered through our findings, and the relevance of our research to related domains.
- Chapter 11: draws final conclusions and outlines future work.

Chapter 2

Background

2.1 Introduction

In order to identify and analyse the tensions between real world music production activity and the abstraction mechanisms in production software, we must introduce the relevant background. Systems that allow creative professionals to build and edit complex digital artefacts have much in common with programming systems for end users. In particular, they both rely on task specific notational systems. Nardi described the necessity of designing notational systems to support a specific task domain as follows:

“Task-specificity has two advantages: (1) it affords users ready understanding of what the primitives of the language do (because they already know them from their task domain), and (2) it eases application development because users can directly express domain semantics in the high-level operations of the language — there is no need to string together lower-level operations to get the desired behaviour.” — Nardi [105, p.39]

Each aspect of this quote relates to various disciplines and established theory that we introduce in this chapter:

- The study of how to design user-interfaces to better support users' tasks is called Human Computer Interaction (HCI). Nardi herself supports the use of 'activity theory' as a theoretical tool for HCI researchers studying end user notational systems [105, p.54].
- Exploiting users' language and understanding of domain knowledge requires an analysis of users' conceptual models; and user-interface metaphor has become a common technique to leverage this existing expertise. The multitrack-mixing music production metaphor is a prime example.
- Higher-level aspects of domain specific languages are otherwise known as 'abstractions', and are a central component of experts' conceptual models.

2.2 The domain of computer mediated music production

The relationship between western music and technology is so strong that it has been argued that music is “at the core of modern world techno-culture” [76]. Tools to sequence and automate musical performance date back to ancient Greece and Arabia [77, p84]. Examples such as Heron of Alexandria's self-playing organ pipes, and the Mûsà brothers' automatic flute-player from Baghdad show how the desire to automate musical systems dates back to the early days of mechanical technology.

Automated music technologies are dependent on both the ability to record musical gestures, and to independently recreate or synthesise musical sound as a result. For our purposes, a musical gesture is physical movement that directly expresses musical meaning (such as hitting a drum or cueing a verse in real-time) and can be recorded in some representational form. This can be seen in contrast to movement simply involved in editing an abstract notational system (such as drawing a note, or typing in a volume value) where

the actual movement conveys no musical meaning in its own right. In many ways the invention of mechanical intermediaries between human gesture and musical sound was an important precursor to the ability to automate music. This use of technology to separate gesture and the resultant sound can be seen in early mechanical instruments such as carillons and organs.

Figure 2.1 shows how the addition of these types of recording technologies, both for capturing gesture and sound, would begin to mediate the traditional performer-instrument-audience relationship that we discuss in the following sections.

Figure 2.1: Here we show how recording technologies de-coupled the relationship between the performer, their instruments, and their audience. Gestures (shown as broken lines) and audio (full lines) could be captured and edited by producers through recording technologies. These gestures could then be used to automate instruments for performance, and recorded audio played back to audiences.

2.2.1 Recording, reproduction and synthesis of sound

The recording of sound began in 1877 with Thomas Edison’s invention of a ‘talking tin foil’ device [38, p.1]. He named this the “phonograph”, meaning voice writer. Following the development of the gold master record technique in 1887 it became possible to separate the act of recording and reproduction.

Synthesis originated with Elisha Grey in 1874 when she invented the first electronic musical instrument [78]. Grey called the device the ‘musical telegraph’, and the first real synthesiser (the Telharmonium) followed in 1900. This instrument could not only produce electronic sounds, but was able to apply various effects and sound-shaping to the generated signal.

Later developments in synthesisers led to flexible sound generation systems that allowed users to control signal processing with patch cables [56]. These cables carried the audio signal between various sound processing devices and the final signal was sent for amplification. These devices featured knobs and buttons to control each device. An example of this is shown in figure 2.2.

Figure 2.2: A Moog synthesiser that is controlled with patch cords and knobs.

Another example of the further separation of musical gesture systems and sound triggering systems was voltage control. Synthesisers gained standardised voltage control systems through cables that allowed them to be manipulated via a variety of different hardware, with high and low voltages controlling the sound characteristics of the sequencer.

2.2.2 Capture and reproduction of musical gesture

The mechanical separation of musical gesture and sound production is the key advance that allowed the development of modern automated music systems [133, p.199]. This development paved the way for the musical keyboard interface to be replaced with other technologies that triggered musical events. The result was that it was no longer a requirement for a human performer to be controlling every performance parameter and event in real-time. Early examples of these technologies were built into grandfather “flute” clocks that played back tunes at hourly intervals [77]. In 1904 the German firm Welte demonstrated their new player-piano technology [77]. This allowed a piano performer to have their performance recorded as perforations on a paper-roll. These types of systems would eventually led to analogue sequencers, as seen in figure 2.3.

Figure 2.3: The Korg SQ-10 analogue sequencer.

2.2.3 Editing audio and gesture

The first editable medium for recorded audio was actually associated with optical film technologies. Film could be physically chopped and spliced together to form new arrangements of material. Audio editors used the physical film medium to encode and edit sound [47, p.33].

Cinema also had a pioneering role in another area crucial to the development of modern sequencers — the invention and design of samplers. Samplers are devices that can trigger a number of pre-recorded sound effects, either as one-off sounds or repeating loops. The first samplers were designed as sound effect generators for films and cartoons [38, p.141].

The invention of the modern tape recorder in 1948 marked the key transition into the studio age. The tape recorder transformed the process of recording and playback of sound. Magnetic tape could be re-recorded and copied with ease. Tape also brought the ease of physical cut and paste editing enjoyed by film editors to audio artists and musicians such as John Cage [112].

The recording process was again transformed by further advances in tape technologies. The introduction of stereo and multi-track tape recording in the 1950s allowed further separations of musical gesture and finished performance [134]. Firstly, this allowed the mixing process, where volume and sound equalising was conducted, to be delayed until after the source audio had been recorded. Secondly, it was now possible to over-dub individual instruments or multiple ‘takes’ of the same performance, and selectively merge them with other material.

2.2.4 The producer

This represented a serious milestone in the separation of musical gesture and sound reproduction. It was now very possible for musical performance to be assembled from various clips of audio — usually by capturing sound from performers using traditional instruments, but also from sounds found from the world and from synthesised sources. In 1965 Time Magazine expressed how the roles of music production had changed. The ‘grand designer’ of a recording was “no longer the conductor but producer . . . with a mountain of sophisticated machinery at his command, he has become a space-age sculptor of sound. His raw material is the performer, his workshop the glass-enclosed control room” [38, p.131].

Recording studios were rapidly transformed from a place for capturing musical performance into sophisticated composition tools. Producers began to “use the studio in order to ‘compose’ as they went along” [38, p.144], previewing, overlaying, and perfecting ideas as part of the recording process:

“Tape recording allowed producers and engineers to manipulate performances in the same way that it allowed musicians to manipulate sound Multitrack recording . . . creates new musical possibilities; the new mode of production therefore begins to turn the recording engineer — the mixer — into a musical creator of a new kind.” — Chanan [38, p.147]

2.2.5 The Digital Audio Workstation

Once the analogue recording studio had become established, the final step to the introduction of the modern digital audio workstation was the computer. Although it would take some time, all the pieces were now in place for the computer to become the centre of the studio. This would take form in the Digital Audio Workstation (DAW). These systems would bring together the following advances:

- Digital to Analogue Conversion (DAC) and Analogue to Digital Conversion (ADC) computers could capture and playback audio signals, and replace multi-track recorders. Sequencers would facilitate the recording and playback of audio.
- Digital Signal Processing (DSP) computers could replace mixers, effects and synthesisers. Sequencers would provide the infrastructure for sound processing, and virtual instruments.
- Advances in real-time systems allowed computers to replace and enhance samplers beyond what had been previously possible. The introduction of the Fairlight CMI, the first major digital sampler, in the

1970s marked a significant step in the introduction of DAWs. The Fairlight CMI was a “dedicated computer allowing sound to be fed in and manipulated, the software program it employed became, as one writer puts it, ‘the mother and father of all sequencer software’” [38, p.159]. The Fairlight CMI is shown in figure 2.4, and its sequencer interface in figure 2.5.

- The final component of the amalgamation of the studio around the computer and DAW was the introduction of the Musical Instrument Digital Interface (MIDI) in 1984. This allowed any piece of digital music hardware and software to send gestural data to any other in a standard protocol, including note-on, note-off, volume, pitch and other abstract controller data. This would give the sequencer the power to control any device in the studio.

Figure 2.4: The Fairlight CMI — the first commercial digital sampler and sequencer.

With the unification and advances in these technologies it was now possible to capture, create, manipulate, edit, and trigger any sound you could imagine [133]. With such a huge potential power, the only remaining problem

Figure 2.5: *Page R*, the “Real-Time Composer” from the Fairlight CMI sampler, credited as “the mother and father of all sequencer software” [38, p.159]. The Page R sequencer took the form of a basic pattern sequencer, with each pattern having a multitrack organisation.

was how to provide a user-interface that would bring this to the musician and composer, rather than keeping it solely in the hands of computer experts. For professional producers, this took the form of user-interfaces designed around the existing multitrack-mixing model.

2.3 Human Computer Interaction

The study of Human Computer Interaction (HCI) is concerned with the evaluation and design of user-interfaces to address the needs of the users of those systems [1], and is fundamentally cross-disiplinary, incorporating the fields of psychology, computer science, sociology, anthropology and design among others. As such, the study of digital crafting systems in general, and computer mediated music production in particular, are an interesting area for HCI focus.

The study of HCI has led to the development of methodologies for analysing many types of task. However, techniques such as task analysis and decomposition fall short when applied to particularly rich and dynamic domains. Traditional methods typically fail to account for “the tacit knowledge that is required in many skilled activities, or the fluent action in the actual work process” [13] that needs to be taken into consideration when analysing open-ended and exploratory activities. These methods can also fail to describe the rich context of activities such as music production, which must be taken into account when looking at tool design. Scenarios, use cases, and essential use cases are based on the assumption of ‘structured narrative’ [46, p.103] which is more suitable for use with multi-step aggregate tasks than the creative open-ended composition process. As such, systems that allow creative professionals to build and edit complex digital artefacts provide an interesting problem for HCI research. One area of HCI that has focused on this is in the study of the use of notational systems.

2.4 Digital craft and notational systems

The domain of music production systems is just one example of an important class of software tools used by creative professionals. At their core, computers are powerful symbol manipulation machines [86]. This fact is increasingly exploited in many creative activities that essentially consist of the creation and manipulation of complex information artefacts, of which computer mediated music production is a prime example. Other examples include activities originating from non-computer dominated domains such as architecture, engineering, industrial design, graphic design, animation, and movie-making, all of which have been transformed by the application of computer aided design systems. These types of systems are also at the centre of entirely new forms of artefact creation, such as 3D modelling, interactive media (such as shockwave-flash), and mixed-multimedia.

All of these activities have in common the underlying creation and manipulation of digital artefacts. In Malcolm McCullough's book 'Abstracting Craft' [100], he describes in detail what these activities share with traditional forms of handcraft. Firstly, traditional craft and new forms of digital craft both involve manipulation of a medium with distinctive properties and 'feel'. In both cases, subjects interact primarily through the hand, although this interaction with the medium is often highly mediated by tools. In a crafting activity the tools change the experience of working with the medium. McCullough uses the traditional crafting example of the potter and wood carver: The medium being the clay or wood, and the tools including the lathe, water, or carving instruments. With digital crafts, such as 2D graphic design, the medium emerges from the behaviours and constraints of both the digital artefact that the subject creates and manipulates, and the environment in which it is manipulated. For example, Adobe Photoshop provides 'layers' of graphical elements that create certain interaction possibilities, and the provision of vector versus bit-map facilities further define the nature of the medium. In this context, tools include virtual brushes, colour pickers, selection tools, rulers, cutting, text tools and many others. In some cases

tools can begin to change the feel and character of a medium and provide new handles for manipulating it, blurring the distinction between the two.

In McCullough’s words the digital artefact, which is the object of focus of the digital craft activity, can be described as:

“...a cumulative composition more like a work of architecture.

Like a traditionally made form, it represents the results of many simple repetitive actions, where one move enables the next. We build up digital artefacts one step a time, one piece on top of another, with what is already there affecting what is to come.”

— McCullough [100, p.151]

These digital artefacts require a representational form in order to be accessible to subjects. This is most naturally found in notation systems [107]. Notations are an information visualisation technique created through graphical marks on an underlying spacial substrate [15]. Unlike other information visualisations, each digital notational system has a vocabulary of primitive graphical elements, and a grammar defining possible combinations and their associated meaning; in effect defining a visual language that can describe the structure of a set of digital artefacts [69, 9]. Examples of notational systems that can represent digital artefacts include textual and visual programming languages, 2D and 3D drawing and modelling representations, and various music notations found in our music production systems. In terms of McCullough’s crafting analogy, the notational system takes on the role of the medium in which the craftsperson is working. In contrast to static paper-based notations, the digital equivalent can allow interaction, specifically manipulation, that creates many additional demands on the design of the notation along with new possibilities. Due to the visual nature of notational systems, the persistence of visual objects in the notation and the dominance of mice and other gestural input devices, this takes the form of ‘direct manipulation’ [125]. Direct manipulation facilitates the “rapid incremental reversible operations whose effect on the object of interest is immediately visible”, which fits well with the idea of the crafting of digital artefacts.

2.5 Abstraction

Nardi’s emphasis on the higher-level aspects of domain specific notational systems is echoed by Buxton’s approach to bridging novice and expert performance:

“...there should be as close a match as possible between the structure of how we think about problems and the language or representation that we use in solving them. . . . This can be achieved by engineering the pragmatics of the human-computer dialogue to reinforce the chunking that we believe would used by an expert working in the domain.” — Buxton [34]

This ‘chunking’ Buxton refers to is a form of abstraction. In our efforts to understand the tensions between abstraction mechanisms and the music production activity, we must explain what abstraction entails, and why it is important. Abstraction is the process of managing complexity by creating and manipulating common patterns and generalities:

“The acts of the mind, wherein it exerts its power over its simple ideas, are chiefly these three:

1. Combining several simple ideas into one compound one; and thus all complex ideas are made.
2. The second is bringing two ideas, whether simple or complex, together, and setting them by one another, so as to take a view of them at once, without uniting them into one; by which way it gets all its ideas of relations.
3. The third is separating them from all other ideas that accompany them in their real existence: this is called abstraction: and thus all its general ideas are made.”

— John Locke, 1690 [98]

By wrapping specifics inside abstractions people can work at a higher level without constantly struggling with a vast number of details. Abstraction is a well understood phenomenon in the discipline of computer science where the essential task is dealing with complexity in expression [2], but abstraction also has many applications to all complex information management. One common problem in notational systems for creating digital artefacts is "... a common failing of direct manipulation at too low a level: far too many people work by making one object at a time, without a thought to how this next object might be an instance or transformation of another existing object" [100, p.151]. For example, 3D modelling software allows subjects to create points in space (vertices), join them with lines (edges), and create 3-dimensional surfaces (faces) from them. By combining millions of these fundamental elements, modellers can create entire scenes and manipulate them over time to create animations. However, limiting direct manipulation to this low level of vertices and edges confronts subjects with the complexity of the artefact, and places practical limits on the sorts of actions the subject can master. A subject working on a 3D model of a face (as seen in figure 2.6) would find it extremely difficult to create various facial expressions if they were forced to manipulate the thousands of individual vertices that need to be moved. This problem occurs when users are not provided with an adequate *task specific* high-level language to express their goals.

As an illustrating example, figure 2.7 shows a number of screens where the subject is provided with a task specific abstraction (on the right side) of a face model (on the left side). In this case the abstractions include the various muscle groups and bone structure of the face. The end result is that the subject can now move a single abstract element, such as the eyebrow, and thousands of vertices will move appropriately. The subject's ability to manipulate facial structure, rather than individual vertices, is an example of what we can term 'abstract direct manipulation'; where abstract entities are represented as persistent near-concrete graphical objects that can be manipulated directly via gesturing devices.

Figure 2.6: A 3-dimensional model of a face. *Leftmost:* The underlying 3D points, lines and surfaces making up the face. This is a reduced polygon version of what would be used for most modelling purposes. *Middle:* A graphical texture mapped over the underlying facial model. *Rightmost:* Additional lighting and rendering effects applied to the same model. [124]

Another example of the importance of abstract direct manipulation in digital crafting systems can be seen in architecture and engineering domains. Autodesk’s Revit software includes increasing types of abstraction, allowing a change to a single abstract ‘smart object’ to be reflected across thousands of walls, electrical fittings or other building structures. Interactive media design applications such as Macromedia flash allow a single change to a clip of animation to be reflected wherever it is used.

All of these forms of abstract direct manipulation are dependant on system designers providing means not just for abstractions, but for the right abstractions [114].

2.5.1 Types of abstraction

There are a number of common types of abstraction that have been identified in disciplines such as philosophy through to psychology, and find their way into computer science and other areas that attempt to deal with complexity.

Containment or grouping captures the concept of a single whole wrapped

Figure 2.7: Six frames of a 3D facial animation. Each frame shows the rendered face and a small task specific abstract representation of the face allowing abstract direct manipulation of the model. [124]

around composite parts through modularisation [114]. This is what Locke referred to (quoted above) as “combining several simple ideas into one compound one; and thus all complex ideas are made” [98]. Gestalt psychology describes how people can see a single entity emerge from the sum of its parts [140], and some patients suffering from prosopagnosia are unable to see the abstraction of a ‘face’ as an entity but rather see only the individual features of the face so are unable to operate normally [24]. In computer science, by defining code and data into reusable blocks (functions or objects) the developer can hide and ignore the complexity of the component and therefore comprehend larger projects [2]. Also, the ability to reuse the component across their software project allows a change (such as a bug fix or feature improvement) made in the shared component to be reflected wherever it has been used [16].

Classification of various objects into some form of hierarchy is an important part of how we understand the world. Plato described how various objects in the world are instances of conceptual and eternal ‘forms’, and thus are classified based on their adherence to that form [118]. Aristotle’s refinements to the concept of classification provided an understanding of taxonomies, in particular those from the natural world. This type of thinking is evident in class based programming where ‘car’ objects might all share behaviour and properties of a common car class [131]. From taxidermy to computer science, these classifications can fall into hierarchies of inheriting behaviour and structure, with Ford cars being a ‘kind of’ car, and cars being a ‘kind of’ vehicle. More modern conceptions of classification, such as proposed by the philosopher Wittgenstein [146] often focus on less canonical structures of classification [131]. Resemblance and shared characteristics or traits can be described by the abstraction of prototype inheritance, which can be seen in action in prototype based programming languages [109], or alternatively through specification of shared behaviour with object interfaces. Philosophy has many other theories with various implications for classification, such as those found in the study of universals, natural types, essences,

and bundle theory [118, 80].

Other ways in which abstraction can be represented include *references* of various kinds from one entity to another (Locke’s second characteristic of abstraction quoted previously) [98].

In visual programming notations, there is often a trade-off in the continuum between concreteness and abstractness [31]. Concrete representations are those that depict an instance of an object, and thus have specific properties and behaviours. In contrast with concrete representations, mastering abstract representations can require a larger attentional investment as they describe an entire class of entities in a single description [18]. It is sometimes possible to reduce this difficulty with the use of prototype based models where abstraction is implemented through references between instances of objects, or using approaches such as linked editing [136] where pattern matching allows manipulation of material to be mirrored in similar material.

2.5.2 Conceptual models

The general approach of computer science treats the structure of programming languages as stemming from a vocabulary of primitive entities, the means of combining them, and the means of abstraction [2]. Naturally enough, the first two elements of this are derived from linguistic analysis where language is built from the particles and syntactic structure of a grammar [39]. Means of abstraction then, are a form of higher-level semantic structuring. This same analysis can be applied to notational systems [69]: the language consists of primitive elements, various means of combination and also abstraction (see for example Mitchell’s visual vocabulary for architecture [104]). In the context of notational systems, these parts taken together form the conceptual model of the system.

In HCI, the term *Conceptual Model* has been used and defined in many conflicting ways, which we describe now. There are several different types of model that are related to or can be defined as conceptual models. In general these models consist of a representation of the objects, relationships, and

behaviour of a system. One common understanding of the various types of models that play a role in the design and use of user-interfaces were articulated by Donald Norman [110] and can be seen in figure 2.8.

Figure 2.8: Design model, User's model and System Image, Norman and Draper [110, p.46].

Firstly there is the designers' model of how the system operates (*Design Model*). The designer will take charge of implementing some version of this model into the appearance and behaviour of the system (*System Image*). A distinction is sometimes made in the system image between the detailed model of entities and their behaviours implemented in an interface, and a more general higher level model that may be present — such as a metaphor where the application at some level behaves like an existing artefact. Finally, through interacting with the system image, the user will develop a mental model (*User's Model*) of the system. The users' mental model [48, 84] will typically be incomplete, inaccurate, or include additional domain knowledge. Users' mental models can be divided into two general types as seen by DiSessa [55]. There are *Structural Models* where the user develops a model of the structure and internal behaviour of a system, and *Functional Models* where the user develops an understanding of causality in the system and how they can get the system to perform various tasks without having a deeper understanding of its structure.

Applications tend to have strong ties with existing concepts or systems. For example, applications for managing business processes, controlling physical processes, or simulating real-world situations all relate to other systems of knowledge.

Over time, recurring solutions to common problems have been implemented in system models. These have been crystallised by researchers as ‘pattern languages’. Each pattern consists of a common name, a declaration of problem, the conceptual model of the solution, and examples. Inspired by the architectural pattern language of Christopher Alexander [5], pattern languages for both software engineering [68] and user-interface design [135] have been developed previously.

2.5.3 The cognitive dimensions of notations

Green and Blackwell’s ‘cognitive dimensions of notations’ provides a common vocabulary for describing reoccurring trade-offs in the conceptual models employed by notational systems [21]. Cognitive dimensions have obvious relevance to our enquiry, providing a terminology and framework for approaching the tensions between music production abstractions and the activity of producers. In practice, the cognitive dimensions of a specific notational system arise from the interaction between both the notation itself, and the editing environment in which it is manipulated. For example, powerful pattern matching find and replace tools might compensate for the lack of certain abstractions in the graphical notation itself.

In this thesis where we use technical cognitive dimensions terminology we present them in a bold typeface to make their technical usage clear. The cognitive dimensions are as follows:

- **Hidden dependencies** occur when important links between entities are not visible.
- **Premature commitment** places constraints on the order of doing things.

- **Provisionality** is the degree to which users are committed to actions or marks.
- **Secondary notations** allow extra information to be added using means other than the formal syntax, such as notes and comments.
- **Viscosity** is the resistance of aspects of the notation to change.
- **Visibility** is the extent to which components can be easily viewed.
- **Consistency** occurs when similar semantics are expressed in similar syntactic forms.
- **Diffuseness** of the notation is determined by the verbosity or terseness of the notation.
- **Error-proneness** is defined by how difficult it is for the user to avoid mistakes.
- **Hard mental operations** occur when a high demand is placed on the user's cognitive resources.
- **Progressive evaluation** allows work to be checked at any time.
- **Role-expressiveness** is how easily the purpose of a component can be inferred.
- **Closeness of mapping** is how well a representation fits to a specific domain.
- **Abstraction** is the extent to which higher level abstractions are provided in the notation.

The cognitive dimensions of notations have been used to evaluate programming languages [72], visual programming environments [147], spreadsheets [138], and Application Programming Interfaces (APIs) at Microsoft

[41]. Blackwell et al. briefly applied the cognitive dimensions to music type-setting tools [22, 23], primarily as a proof of concept and validation of the dimensions rather than a detailed examination of the music tools. More recently, Blackwell and Collins applied the cognitive dimensions to textual music programming tools and Ableton Live [19]. This evaluation was from the perspective of “the practice of Live Coding” [44] where textual program code to algorithmically generate music is written and executed live on stage, rather than our domain of professional music production.

2.6 Activity theory

One analytical framework that has strengths in describing context and creative activity is cultural historical activity theory [106]. Activity theory stems from the Russian branch of Psychology, based principally on the work of Vygotsky that emerged in the 1920s [141]. The resulting family of theories focuses on the context of human activity, and explains human consciousness in these terms. The theory uses the notion of an ‘activity’ as its central unit of analysis. Activity is a higher level of analysis than the traditional HCI concept of ‘task’, and includes the meaningful context of individual actions in a group, rather than just the actions themselves. Nardi argues that activity theory has much to offer the process of user-interface analysis and design as it:

“... offers substantial tools for a broadly scoped study of ‘computer-mediated activity’ ...it weaves together, in a single coherent framework, so many interesting theoretical constructs crucial to an understanding of human activity: dynamic levels of activity, mediation, contradiction, intentionality, development, history, collaboration, functional organ, the unity of internal and external.” — Nardi [106, p.375]

Activity theory encompasses and integrates a number of different areas of analysis for any given activity, including structure, context and development.

The structure of the activity is decomposed into its sub-components, and analysed in terms of human motivations. This decomposition includes the full range from understanding the broad activity and its motives, individual conscious ‘actions’ and ‘goals’, through to people’s sub-conscious ‘operations’ and the ‘conditions’ that trigger them [93].

In activity theory, subjects’ consciousness is explained through their ‘internalisation’ of abilities such as mental addition, mental simulation, touch typing, and language. Humans also extend and share their consciousness through ‘externalisation’. This includes note taking and the use of symbolic and abstract tools and artefacts.

Activity theory places a special emphasis on how tools mediate action between any number of ‘subjects’ and the ‘object’ of the activity, which subjects transform into the activity’s ‘outcome’. We argue that this makes activity theory particularly well suited to understanding the use of notational systems in crafting complex digital artefacts, given technology’s mediating role in this domain [100].

The context of an activity is analysed in terms of various influencing forces in the environment in which the activity takes place. The contextual analysis also examines the impact of the surrounding community, explicit and implicit rules, and the division of labour.

As it is otherwise known, ‘cultural *historical* activity theory’ places a strong value on understanding the historical development of an activity. Changes in activity over time determine the current form of activity. Therefore it is vital to examine an activity’s history in order to fully understand it, and how it could be transformed in the future.

These various aspects of the theory provide a broad and flexible framework and terminology for describing human activities and the factors that influence them.

2.6.1 Activity theory in HCI

Much of the comparatively recent applications of activity theory has focused on activity systems in complex work environments [62]. This led firstly to a natural growth in its application to computer-supported collaborative work (CSCW), and later to further use in the more general field of HCI. In 1991 Bødker introduced activity theory to HCI explicitly in *Through the Interface* [25].

Activity theory seems to be a natural fit in the HCI domain due to its basis in the central concept of the mediating role of tools in human work [106]. Activity theory provides a comprehensive framework for many of the concerns of HCI practitioners and researchers:

- The understanding of activity structure as a hierarchic set of “activity, actions, and operations” [93] provides additional conceptual insight to conventional HCI techniques such as hierarchical task decomposition.
- Activity theory’s model of the social environment provides a framework for ensuring a user-interface takes into account contextual issues.
- The study of historical development in activities [106] can provide HCI experts with a model for understanding how computer tools have changed activities in the past, and how changes they make may impact the future activity.
- Activity theory highlights the inherent contradictions in activities [139]. These must be taken into account when designing interfaces to support a particular activity.
- User-interfaces need to facilitate the natural human processes of internalisation of activity, and externalisation for extension of consciousness [141]. Activity theory provides a theoretical framework to account for these processes.

- Finally, activity theory exposes the importance of the transition between conscious and subconscious cognition [93], which is of great importance for learning and mastery of user-interfaces and their system models.

Most importantly, activity theory provides a body of thought and a unified framework for these concerns that might have previously fallen under the term ‘task analysis’. Activity theory can provide this framework in concert with traditional methods such as those associated with user centred design [111]. This allows the HCI specialist to develop a thorough and broad understanding of an activity, and a consistent model to describe it.

However, activity theory’s long history, wide-ranging complex and rich approach, and theoretical focus has not lent itself to easy application in the HCI field. Activity theory provides no step-by-step methodology.

Some HCI practitioners have taken a general activity theoretic approach to HCI analysis by firstly immersing themselves in the varied activity theory literature, and then using other techniques to learn about a specific domain where they wish to develop a new user-interface [106]. Activity theory is then used as a framework for explaining and making sense of their findings. This sort of model can then be used as the basis for further, more systematic analysis. For example, Turner and Turner [139] carefully uncovered activity theory ‘contradictions’ in a domain to drive requirements gathering.

While these general activity theoretic approaches have their place, mastering the entire framework is an onerous task that has likely held back its application more universally [30]. With no clear methodological guide it is difficult for HCI practitioners to take advantage of the framework’s potential. Kaptelinin, Nardi, and Macaulay, some of the chief theorists behind activity theory’s application to HCI, admitted that:

“These general principles [of activity theory] help orient thought and research, but they are somewhat abstract when it comes to the actual business of working on a design or performing an evaluation.” — Kaptelinin, Nardi, Macaulay [85]

Despite these difficulties, activity theory is clearly a good fit with the study of creative professionals crafting complex digital artefacts. In our case, not only does activity theory provide a framework for describing the historical development and structure of computer music production; it also accounts for the mediating role of complex digital representations in music production, the externalisation of composers' actions in these artefacts, and the way in which the structure of the artefact are internalised by the user.

2.7 User-interface metaphor

We have seen the importance of abstract direct manipulation in digital artefact creation, and it is clear that design success depends on appropriate abstraction mechanisms being provided by the system's conceptual model. Unfortunately, the multitrack-mixing model employed in professional music production systems owes more to the design of historical hardware recording solutions than to any systematic model of music. This concept of designing a user-interface to imitate real-world hardware is an example of what is called 'user-interface metaphor' [14].

The introduction of Graphical User-Interfaces (GUI) in the 1980s [82] was a key step in defining the form of the DAW, and a common characteristic of these GUI systems was the use of user-interface metaphors. User-interface metaphor can be defined as "a device for explaining some system functionality or structure by asserting its similarity to another concept or thing already familiar to the user" [14]. The most commonly cited example of this is the desktop metaphor, where the user's computing experience is mediated by the representation of a desktop — complete with folders, files, and a trash-can. Figure 2.9 demonstrates user-interface metaphor in action in the Apple OS X operating system. This icon above represents the 'folder' metaphor where items are stored "in" user-interface objects that in many ways behave and look like folders. The lower icon is an example of the 'trash can' metaphor. Until users 'empty the trash', any material they place here can be reclaimed.

Figure 2.9: ‘Folder’ and ‘trash can’ examples of user-interface metaphor in icons from the Apple OS X Dock. The user-interface objects that these icons represent in many ways behave and look like folders and trash cans from the real world.

Lakoff and Johnson have identified three types of metaphor [90] that have been successfully applied to the analysis of user-interface metaphor [14]. These have traditionally been explained in terms of other more typical user-interface metaphors, but we describe each of these here using examples from computer music applications.

Orientational At a basic level people use *Orientational Metaphor* to understand concepts in terms of spatial orientation. An example from the computer music tool domain is LOUD IS UP. Figure 2.10 shows volume controls from the music applications Reason and Live. Dragging these controls upwards with the mouse increases the loudness of the sound output.

Ontological People also use *Ontological Metaphor* at a basic level in utilising the understanding of physical objects and substances in abstract areas. These metaphors use conceptions such as objects, relationships, and containers. For example, the music application Reason allows the user to hide a number of effects inside a ‘combinator’ device. This

Figure 2.10: Volume controllers from the user-interfaces of Live and Reason demonstrating the orientational metaphor LOUD IS UP.

is shown in figure 2.11 and demonstrates the ontological metaphor THE COMBINATOR IS A CONTAINER. In user-interfaces Ontological metaphors allow us to reuse our knowledge of the general properties of the objects in the physical world.

Structural The most recognisable form of user-interface metaphor is *Structural Metaphor*. Structural metaphors take some complex object or concept and apply it to the target domain. Figure 2.12 shows the structural user-interface metaphor THE APPLICATION IS A MUSICAL KEYBOARD. The user can click on the graphical depiction of the keys to produce musical output.

Metaphors are often multi-levelled and composite. It is typical for structural metaphors to be composed of multiple sub-metaphors of any of the three types. For example, the user-interface metaphor THE APPLICATION IS A MUSICAL KEYBOARD has the sub-metaphors THE RIGHT IS HIGHER PITCHED, A KEY IS A NOTE, and A NOTE IS AN ENTITY among others.

2.7.1 Usability and metaphor

User-interface metaphor is in wide usage and has been recommended by influential companies such as Apple [8] and Microsoft [102] who offer guidelines

Figure 2.11: The ‘combinator’ from Propellerhead’s Reason demonstrating the ontological metaphor THE COMBINATOR IS A CONTAINER. In this case, the combinator holds a number of effects units within itself such as the PH-90 Phaser, reMix device, and NN-XT sampler.

Figure 2.12: The interface of VMK (Virtual Midi Keyboard) demonstrating the structural metaphor THE APPLICATION IS A MUSICAL KEYBOARD.

for its use. Academic analysis has both argued for the usefulness [82, 83] and dangers [37, 111, 17] of using metaphor in the user-interface. The benefits of user-interface metaphor are chiefly that users can reuse existing knowledge of the metaphor, while the primary disadvantage is that to a certain extent “... a metaphor is always wrong, by definition” [111, p.180]. Barr has brought these observations together and developed a number of heuristics for the application of user-interface metaphor [14]. These heuristics use a special metaphor terminology from Richards [123], employing his terms of ‘tenor’ and ‘vehicle’. The concept to which a metaphor is applied is called the *tenor* and the originating concept from which the metaphor stems is called the *vehicle*. The characteristics of the vehicle that apply to the tenor are termed the *metaphorical entailments*. Barr’s heuristics are:

1. *When choosing metaphors based on real world objects consider the real world usability issues for insight into interface usability.*
2. *When using orientational metaphors, make sure they fit into the overall framework defined by that orientation.*
3. *When using new metaphors, carefully indicate the UI metaphorical entailments to the user.*

4. *When using conventional metaphors, be careful to indicate any deviation from the accepted set of metaphorical entailments.*
5. *Aim to use as many metaphorical entailments as possible while still keeping a firm focus on actual functionality.*
6. *Represent only those affordances of the vehicle which have an implemented function in the user-interface.*
7. *As far as possible make sure that metaphors are coherent with one another.*
8. *Ensure metaphors do not inhibit access to any functionality.*
9. *Always address the aspects of the system not explained by metaphors.*
10. *Harness the user's experience to help identifying useful vehicles.*
11. *Allow expert users to circumvent the metaphorical means of achieving tasks where such circumvention would increase task efficiency.*

Heuristic 1 is one of the most important. The central lesson of this heuristic is that user-interface designers should not take the usability problems inherent in real world systems and reproduce them in the user-interface of software. For example, the time setting functions of video recorders are an archetypal example of a real world device that difficult to program [70]. This heuristic dictates that it would be a mistake to use this as a user-interface metaphor for setting the time on your computer where much more elegant and flexible interfaces are possible. The physical and cost limitations of real world devices do not apply to software user-interfaces. Building such limitations into software user-interfaces for historical reasons is a serious error, unless it is absolutely vital to reuse existing knowledge (heuristic 10).

Regardless of this advice, the usability problems of real world systems *do* make their way into the user-interfaces of important software products though the use of metaphor. An example of this problem can be found in the user-interface design of the application 'Reason'. As an initial step in

our research for this thesis we developed and published [57] a systematic evaluation of the user-interface metaphor used in Reason, and discovered many issues stemming directly from it.

2.8 Summary

In this chapter we introduced the domain of computer mediated music production, centred on the Digital Audio Workstation (DAW) where gesture and audio have fallen under creative control of the producer. In order to understand these systems, we turned to the discipline of HCI, and its investigation of notation based computer systems for crafting complex digital artefacts. We found that identifying *appropriate* higher-level abstractions that are domain and task specific is central to enabling these types of creative activity, and that abstractions are a central component of the conceptual models built into these tools and the mental models of their users. We presented the cognitive dimensions of notations as one tool in analysing these systems, and activity theory as a framework for understanding the creative activity engaged in by professional producers. Finally, due to its crucial role in the development of the multitrack-mixing model, we introduced user-interface metaphor, a common approach for developing the conceptual model for computer systems.

Chapter 3

Models and Abstractions for Music

3.1 Introduction

The conceptual model inherent in the multitrack-mixing model, which we describe in this chapter, is the basis of the means of abstraction for much commercial production activity. Understanding the historical development of this model is central to understanding the activity of computer mediated music production. These abstractions have origins in the perceptual psychology of music, music performance practice and theory, as well as the developmental history of recording technologies. In this chapter we identify a number of these abstractions which are crucial to music producers' activities.

Finding a good set of abstraction mechanisms for music tools is of recognised importance, as demonstrated by the following quotes:

“Musicians deal with many levels of abstraction in music. If a conductor says, ‘play the downbeat with more conviction,’ he or she is referencing music structure (a downbeat) and emotional content in the same sentence. It is convenient to think of musical representations at different levels, ranging from the highly

symbolic and abstract level denoted by printed music to the non-symbolic and concrete level of an audio signal.” — Dannenberg [51]

“In a computer music system, the grouping of data into larger units such as a sound-object, event, gesture, distribution, texture, or layer may have a profound effect on the composer’s process of organization. . . . Musical structure can usually be thought of as multilevelled, and therefore a computer music system that can address different levels of musical structure effectively will be a powerful compositional tool. And powerful tools lead the creative imagination in new directions. However, the introduction of each level of hierarchy implements a specific musical-acoustic model and ‘prejudices’ the system with a specific hypothesis about how music is structured.” — Truax [137, p.157]

“A computer music language should provide a set of abstractions that makes expressing compositional and signal processing ideas as easy and direct as possible.” — McCartney [99]

“Decisions made by engineering teams at the early design stages of a processing device can thus have a profound impact not only on the ability to make use of the device but also on musical / compositional practices and concepts.” — Théberge [133, p.200]

3.2 Models of music

In the previous chapter we argued that the ability for direct manipulation of higher level abstractions is a vital part of digital crafting. In this chapter we draw together existing evidence that composers *think* in terms of higher level structures. As such, it is crucial that systems provide appropriate abstraction mechanisms to music producers:

“... composers typically do not think in terms of ‘dots on pages’ or detailed note-lists, but rather in terms of higher-level structures, such as rhythmic motifs, melodies, tone rows, harmonic progressions, orchestration and so forth.” — Hill et al. [75]

The difficulty, however, is that it can be nontrivial to decide on the exact means of abstraction to provide. This is further complicated by the multitude of differing higher level structures that can be utilised for varying purposes:

“Music is a multifaceted thing, so perhaps we should not be surprised that so many different models of musical structure have been contrived over the years, each, according to the theoretical predilection of its author, attempting to shed light on one aspect of the medium or another.” — Ockelford [113, p.129]

These higher level abstractions have a rich history in the study of the perceptual system of the musical listener, the classical music and musical theory traditions, research on digital representations of music, and in a variety of experimental music tools. We outline a number of these below.

3.2.1 Psychological perception of music

Psychological gestalt theorists study people’s ability and tendency to perceive patterns and groupings arising from complex underlying stimuli. While gestalt theory is more commonly associated with visual stimuli, the obvious application to music was not lost on early gestalt researchers, with the first paper coining the term using examples from musical perception [140].

The gestalt properties of sound were further illuminated by Bregman’s theory of ‘Auditory Scene Analysis’ [27] that dealt with the perception of sequential and simultaneous streams. The core of the theory concerns the perception of both ‘vertical integration’ and ‘horizontal grouping’, primarily influenced by psycho-acoustic grouping principles of pitch [88], timbre, and time.

- When multiple simultaneous sounds are perceived as a fusion of sound they create ensemble timbres. This is vertical integration, named for the vertical spacing of distinct parts in a traditional score.
- On the other hand, sequential events can also be perceived to coalesce into a unified stream. Multiple instruments or melody parts can often be differentiated by listeners based on this horizontal grouping, similarly named for the horizontal dimension of time in most notation. Rapid breaks in the consistency of texture or pitch from a single source can create the perception of multiple horizontal streams.

Bregman also described how these perceptions of composers and other listeners could diverge:

“In the process of composing such a piece the composer may listen to a particular sequence of sounds dozens of times and may form a mental schema that describes its form. The composer’s own schema-driven integration may be successful even when primitive grouping processes do not work.” — [27, p.468]

Another model of music perception is the Generative Theory of Tonal Music (GTTM) [95]. While there are other music perception models, the GTTM is the best suited to our purposes due to its focus on hierarchical structure (and hence abstractions) and particular applicability to music in the popular idiom. Moreover, the GTTM is probably the most accepted model of music perception, as argued by Ian Cross, a well known psychology of music and music artificial intelligence researcher:

“This is a theory that has not been without controversy; its predictions are still being empirically tested, and its theoretical tenets are being re-evaluated. However, it is still the most comprehensive theory of the experience of tonal music that exists, good enough for jazz . . . and for El Noy de la Mare.” — Ian Cross [50]

Outlining the basic elements of the GTTM here provides some important insights into the types of abstraction we can expect to be important to composers. The GTTM is a Chomsky type grammar [39], but for musical structure rather than language. It provides a set of weighted preference rules for determining the strongest perceptual structuring present. This underscores the multiplicity of structures that can be simultaneously assigned to a piece, some of which will be perceived by any given listener.

The GTTM is more concerned with complex temporal structure, rather than the stream perception of Auditory Scene Analysis, and explicitly ignores the implications of polyphonic musical structure where more than one part plays simultaneously. The GTTM consists of four core components:

- **Grouping Structure:** "...expresses a hierarchical segmentation of the piece into motives, phrases and sections." One interesting way in which we can identify the use of an abstraction is to identify higher-level domain specific terms used to signify that concept. In the development of popular forms of music many types of groupings have been created. In rock we have the 'riff', jazz the 'lick', in electronic music we have the 'loop', and generically we have the term 'phrase'. Names for larger grouping structures include the 'chorus', 'verse', 'bridge', 'coda', and 'head'. For the purposes of this thesis we will use the generic term *blocks* to refer to all of these types of temporal grouping structures.
- **Metrical structure:** "...expresses the intuition that the events of the piece are related to a regular alternation of strong and weak beats at a number of hierarchical levels." Abstractions of metrical structure are most commonly manifested in the concepts of tempo (the number of pulses over time) and time-signature (the underlying pattern of strong and weak beats), and contrast with the conception of temporal location in terms of time (such as seconds).
- **Time-span reduction:** "assigns to the pitches of the piece a hierarchy of 'structural importance' with respect to their position in grouping and

metrical structure.”

- **Prolongational reduction:** “assigns to the pitches a hierarchy that expresses harmonic and melodic tension and relaxation, continuity and progression.” [95, p.9]

Other dimensions of musical structuring such as timbre and dynamics are excluded by the theory:

“The web of motivic associations (and of textural and timbral associations as well) — let us call it associational structure — is a highly important dimension in the understanding of a piece. But this web is not hierarchical in the restricted sense . . . and it must not be confused with grouping structure. It is a different dimension of musical structure, one that interacts with grouping structure. Because associational structure is not hierarchical, however, our theory at present has little to say about it.” — Lerdahl and Jackendoff [95, p.17]

In ‘Representing Musical Structure’ [79], Howell et al. describe the psychological properties of repetition and variation, all of which are crucial in understanding abstractions for motif development:

- Repetition “. . . may occur over pitch contour, pitch interval structure, rhythmic structure, tempo, or simply duration.”
- Transformation is “. . . when a unit can be derived from another unit in a perceptible way and it is possible to specify the function transforming one into the other.”
- With elaboration and simplification, the basic form of repeated elements are the same, but differ in terms of the inclusion of non-structural units (i.e. ornamentation).

Other models of music perception work at a very detailed level, for example attempting to understand how listeners determine pulse (tactus) [92], or focus on tonal properties of perception [95].

3.2.2 Common Music Notation and theory

As we might expect, these models of music perception are reflected in the scores used by composers and performers in various traditions, as well as the established doctrine of music theory:

“Because much of the pitch and timing information is translated into the two spatial dimensions of the score, many of the perceptual groupings that we see in the score correspond to groupings that we hear in the music.” — Bregman [27, p.456]

Common Music Notation (CMN) is the visual language for scores in the western music tradition [129], and also incorporates aspects of performance practice. The appearance of new notations for abstractions in musical scores has historically had an impact on the theory and practice available to composers:

“By the fourteenth century, French composers of the *ars nova* were achieving new levels of rhythmic subtlety through the adoption of duple time division . . . and the invention and exploitation of new rhythmic and formal complexities in musical notation: new time signatures, colored notes, dots, stems, and flags.” — Paul Théberge [133, p.178]

While other music notations exist, CMN is by far the most common and accepted, and so provides the best basis for examining predominant abstractions in music. CMN visually embodies many abstractions from music theory [73] including:

- The tempo (speed), concept of pulse (beats), and division of pulse into bars through a time signature, are a reflection of the GTTM's concept of 'metrical structure', which allows patterns of strong and weak beats to be expressed. These abstractions also allow notations for various durations of musical events such as triplet, quintuplet, half-, quarter-, eighth-, and sixteenth-note (etc.) divisions of the bar. (See figure 3.1.B).
- Ligature inscriptions denote phrase groupings, one level of the GTTM's 'grouping structure'. Higher level structures are sometimes named on the score, for example, in jazz scores the 'head' and 'bridge' structure can be explicitly written. (See figure 3.1.C).
- Parts are a core aspect of modern music theory and practice. Scores allow depiction of individual staves for different parts, each to be played by individual musicians. These parts can correspond to the horizontal integration of Auditory Scene Analysis. Scores can also represent multiple parts merged into a single staff, providing an overview for the conductor or composer. (See figure 3.1.A).
- Tonal abstractions are provided in music theory and score notation. Most prominent is the codifying of the twelve-tone musical scale through the staff notation, and key-signatures shown as a cluster of sharp and flat symbols at the beginning of a part. (See figure 3.1.D).
- While CMN does not specify the exact details of performance (instead leaving this to the performers interpretation)[145, p.7], there are a number of markings that can be used to provide information on additional performance nuances. For example; accents, staccato (short attack), marcato (loud attack), pizzicato (for stringed instruments), vibrato (pitch wavering), and sustain can be signified by various inscriptions. (See figure 3.1.F).
- Repetition is also represented in CMN. Examples include repeat signs

(to repeat the bars between the signs), simile marks (for repetition of the material immediately preceding the mark), Da capo (to repeat from the beginning), Dal segno (to repeat from a specified point marked by a Segno). Variations of repetitions are facilitated primarily through the limited means of Volta brackets (1st and 2nd time repeat bars) and Coda to jump forward out of a repetition. (See figure 3.1.E).

Of course much varied repetition, such as the development of theme and variation, and motif exploration, cannot be represented by CMN, and can only be found through the direct comparison of musical material. This is also an issue for theory as "...it is difficult to make a substantive theory of associations. Within music theory this remains relatively uncharted territory" [94, p.6].

There are other areas of music theory in which abstractions are well understood. For example, in 'The Elements of the Jazz Language for the Developing Improviser' [43] Jerry Coker details a number of abstractions that reoccur as patterns in Jazz performances, and presents them as practical tools for jazz students. For example, some of the 'patterns' that Coker identified include 'Change Running', 'Digital Patterns' and '7-3 Resolution'.

3.2.3 Digital representations of music

While both the psychological theory of music perception and the development of common music notation and music theory tell us much about basic musical abstractions, they have little to say about aspects of performance practice specific to computer mediated music production. For example, the score alone does not symbolically support descriptions of sound synthesis and effect processing crucial to much computer mediated production:

"In Western music the score represents a structure separated from the sound through which it will be realized. Therefore, it is hardly surprising that the principal model of computer music software is the score editor. The problem is that the score is a description

Figure 3.1: A number of graphical marks from Common Music Notation depicting various abstractions. (Individual images sourced from Wikimedia commons [144])

of the surface level of the resultant composition, not a guide to the musical thinking that produced it.” — Truax [137, p.160]

Moreover, Dannenberg points out that CMN does not have the detail required to fully specify performance without the instrumentalist’s personal interpretation, and that conversely, pure performance representations do not have the symbolic structure of notation:

“One of the key problems is that music notation is not just a mechanical transformation of performance information. Performance nuance is lost going from performance to notation, and symbolic structure is lost in the translation from notation to performance.” — Dannenberg [51, p.6]

Even the symbolic structure of CMN is relatively low-level, with few higher-level syntactic and semantic structures [120] which would be vital for direct manipulation of abstractions.

The problem of finding an ideal digital representation to support a set of desired abstractions is an ongoing research problem:

“How to implement the representation is one of the major unresolved issues in the study of music perception. . . . It has long been realized in AI that even when representation systems systems are functionally equivalent to each other, important aspects of what is being represented can be elucidated by some systems and obscured by others.” — Howell et al. [79, p.6]

For example, one problem that has plagued computer music researchers is devising an adequate model for articulating musical ‘texture’ (also known as ‘timbre’):

“. . . at the note level, composers and performers work with a conceptual model of individual instruments whose boundary of abstraction gives very little room for significant manipulation of the inner workings of the timbre of the notes.” — Miranda [103, p.4]

“...starting with sound, we picked out the two things we understood, pitch and amplitude, and called everything else timbre. So timbre is by definition that which we cannot explain. As aspects of timbre are isolated and understood, such as spatial location and reverberation, these components come to be regarded separately, leaving timbre as impenetrable as ever.” — Dannenberg [51, p.5]

Despite these difficulties, computer music researchers have developed many symbolic digital representations of music, for example see [33, 35, 54, 74, 89, 108, 117, 119, 120, 127]. In general, these combine a number of the traditional abstractions from music perception, CMN, and music theory; along with novel abstractions to reflect the production of digital audio. Although we do not require a full description of these models for our purposes, we do note that each utilises a number of the following concepts, with variations in organisation and details of structure between them: Pitch, intervals, chords, key signature, timing, measures, notes, scores, parts, streams, voices, multiple coexisting hierarchies, trees, nested phrases, relationships, repetition, variation, parameterisation, musical objects, timbre, definition and synthesis, functions, processing chains and waveforms.

Dannenberg’s survey of music representation issues [51] discusses some interesting aspects of these models. One of these concerns the distinction between declarative and procedural representations. In the experimental and avant-garde computer music tradition, *procedural* representations allow the composer to create algorithmic musical descriptions of *how* to create music — for so called ‘generative’ composition methods [145, p.199]. On the other hand, *declarative* representations (like scores, event lists, piano-rolls, and the multitrack model’s timeline) have a more direct mapping from symbol to sound as the musical events are specifically notated. This is analogous to our own taxonomic distinction between data-flow and control-flow approaches to music sequencer design [58] developed and published for this research. Another music representation issue discussed by Dannenberg is the instance

and resource models [52]. In the *resource* model, various objects have a persistent existence over time and receive updates. For example a synthesised saxophone object could be a resource that might receive note events and other state changes during a piece. In contrast, the *instance* model requires every musical event to create an independent sound generating instance.

3.2.4 Representations in software for experimental and generative music

As we would expect, these representations have been implemented in a wide variety of computer music tools. Many of these tools support generative composition methods for the avant-garde and experimental tradition, rather than the commercial production domain on which we are focused.

These tools are dominated by specialist music programming languages [99, 26, 142, 75], music Application Programming Interfaces (APIs) for existing textual programming languages [29, 121], and visual programming environments [122]. In this study we focus on professional production tools, and so we do not introduce experimental music tools further here.

3.3 The multitrack-mixing metaphor

The multitrack-mixing metaphor has become the ubiquitous conceptual model employed in professional music production software. This can be more easily understood when viewed in terms of its historical development

“...it is no simple task for a few people in a few days, or years, to overthrow empirical tradition and attempt to create musical instruments or musical structures which will function as well as the traditional ones, or which will function well at all.” — Keane [61, p.99]

3.3.1 Multitrack tape recorders and mixing consoles

The design of modern DAW systems is strongly based in user-interface metaphor, evolving directly from the recording studios in which they were first employed. This takes the form of structural metaphors based on the vehicles of the multitrack tape recorder, and the mixing console; which together we will here refer to as the ‘multitrack-mixing model’. Figure 3.2 provides an overview of the core conceptual model in the multitrack-mixing metaphor.

Figure 3.2: In this figure we show core elements of the multitrack-mixing metaphor. Audio input runs through the channels of a mixing desk, and can be recorded to tracks. Track playback also runs to the mixing desk, through insert effects, optionally to shared auxiliary effects, and the result is mixed to the master output channel.

The multitrack tape recorder (as shown in figure 3.3) records, rerecords,

and plays back multiple streams of audio synchronously. More importantly, each stream can be independently recorded on its own ‘track’ without affecting any others. These streams of audio are represented in a linear fashion on the tape, with time beginning at the start of the tape and time running through to the end, with all tracks lasting for the entire duration. Editing requires over-dubbing new material over old, and cutting and splicing material with razor blades and adhesive tape [38, 112].

Figure 3.3: The Studer J37 4 track tape recorder, the same model used on the Beatles early multitracked album ‘Sgt. Pepper’s Lonely Hearts Club Band’.

The complementary structural metaphor invariably employed in DAW user-interfaces is the mixing console. This stems from the strong developmental link between two technologies described by Paul Théberge:

“It is important to recognize here that the entire development of multitrack recording, and the practices associated with it, is inseparable from a simultaneous evolution in the design of mixing consoles.” — Théberge [134]

Apart from the playback / recording controls of the tape machine itself, principle human-machine interaction takes place through the manipulation of the ‘channels’ in the mixing console. Mixing consoles allow this manipulation via any number of channel strips as shown in figure 3.4.

Figure 3.4: Channels of a mixing desk.

Each channel acts as a proxy for an audio signal, and lies across the y-axis of the mixer surface, with built in volume, equaliser, panning, mute and solo controls provided via a variety of knobs, faders and switches. The result of summing each of these channels output is run through the master channel and can then be output, typically as a stereo pair.

Each channel typically has cabling facilities to link in any number of ‘insert effects’ that sequentially process all audio running through that channel. Effect processing units to be shared by multiple channels can be placed as insert effects on an ‘auxiliary’ channel. Any other channel needing this shared processing then ‘sends’ its audio signal to the appropriate auxiliary channel. For example, when recording a band consisting of several musicians, multiple instruments might need to be processed by the same reverberation effect. The single reverb component would be cabled into the insert effect signal flow of an auxiliary channel. This can sometimes be set up to either re-route the

entire signal output (so it will not be directly mixed into the master channel to allow for sub-mixing), or more commonly, the signal will be split so the channel's output will be sent to both the main mixer output, and to any auxiliary sends, each with their own independent volume. To complicate things further, some effect units allow 'side-chaining' where additional channels can be routed into an effect that is part of any other channel's effect chain. This allows one channel to affect another. For example, a bass-drum audio signal could be side-chained into a compressor on the bass guitar to ensure the bass guitar is audible only when in sync with the drum, to create a tight punchy result.

Each track on the multitrack recorder is associated with a channel on the mixing console. The signal on each of these channels either comes from the studio and can be recorded to that track on the tape, or alternatively is played back from the tape and runs through the desk. The multitrack-mixing console system allows any combination of pre-recorded tracks to be played back while live studio feeds are simultaneously being recorded on their own channels/tracks, and the aggregated mix being played through the main studio monitors. Other channels can remain independent of the multitrack tape machine for send effect processing and sub-mixing.

Producers are able to change the volume, equaliser settings, balance, and processing effect parameters in real-time by manually twisting knobs and sliding faders as a recording is played back. This process, known as 'riding the faders', could with complicated projects overwhelm a single operator and their ten fingers and require a multitude of studio aids (or even the idle musicians themselves) collaborating together to perform all the required changes at the right times. This led to the development of 'automation', which created the ghostly effect of physical controls moving of their own accord. In early systems this was achieved by allowing producers to record and playback their controller manipulations as a performance that was laid down on tape along with the existing audio data.

3.3.2 From physical hardware to user-interface metaphor

These two vitally inter-linked technologies — the multitrack tape recorder and the mixing console — have become the user-interface cornerstone of the modern DAW:

“Early on, we decided that the most probable road to success was to base the software design on the standard, familiar recording hardware of the time: tape recorders and mixing consoles. Plus, we threw in an overall graphic design that built on the conventions being established by the Mac System software (scroll bars, etc.). We added artistic headers to dialog boxes that Performer users overwhelmingly appreciated. Here was a computer not acting and looking like a computer, but instead acting more like a familiar — and cool-looking — recording device” — Jim Cooper, MOTU Inc. [115]

The most visually striking manifestation of this can be found in the mixing user-interface. Figures 3.5, 3.6 and 3.7 show the radical extent to which not only the structural model, but also the look and feel of real world mixing desks are emulated in popular DAW user-interfaces. This extends to the effects processing devices which also emulate virtual physical knobs, through to imitation LCD displays and voltage meters.

Along with the mixing desk, the multitrack tape recorder has also found its place as a metaphor at the centre of the modern DAW. This begins with the basic tape playback and recording controls, which are now embodied in user-interface elements as seen in figure 3.8. These facilitate the navigation and control of a single linear timeline, where all material is located at a specific time and place on the metaphorical ‘tape’. This creates a single linear timeline where material can be recorded, and as such, systems built on this model are often termed ‘linear sequencers’. These contrast to systems utilising a ‘non-linear’ model that allow the user to build a library of semi-structured material that is either cued in real-time through user interaction,

Figure 3.5: The mixing console metaphor implemented in Digidesign's Pro Tools DAW [11].

Figure 3.6: The mixing console metaphor implemented in Apple Logic [7].

Figure 3.7: The mixing console metaphor implemented in Steinberg Cubase.

or controlled by programmed logic.

Figure 3.8: The design of Apple Logic’s ‘Transport’ controls utilise the metaphor of tape playback controls.

The metaphor of the multitrack tape itself uses the orientational sub-metaphor of THE PASSAGE OF TIME IS TO THE RIGHT (which is derived from other temporal-spacial metaphors [91, p.137]), and with individual tracks stacked from top to bottom. As with the vehicle of the multitrack tape, DAW systems retain the metaphorical entailment of tracks lasting the entire duration of the virtual tape. Figures 3.9 and 3.10 show typical depictions of multitrack in modern DAW systems with audio material laid out from left to right to represent a linear timeline.

Digidesign Pro Tools, Apple Logic, and Steinberg Cubase are three big name Digital Audio Workstations (DAWs) used by several of our participants. All three are built on the ubiquitous multitrack-mixer metaphor, with their two shared primary interface components: A mixing-desk view with vertical channel strips stacked from left to right (figures 3.5,3.6), and a multitrack view with time running left to right and horizontal tracks stacked top to bottom (figures 3.9,3.10). Each of these DAWs supports both built-in and plug-in effect processing units that can be chained into the channels represented in the mixer view. Captured Audio and MIDI data, as well as automated effect processing (automation) is represented in the multitrack view, where it can be modified through cutting, splicing, moving, recording, and editing in addition to other manipulations. These DAWs support a wide range of studio music production activity, including composition, recording, editing, and audio mastering of various forms of music. Sony Acid Pro is a

Figure 3.9: Apple Logic’s arrange page, showing a number of horizontal ‘tracks’ with time running from left to right, and a single ‘channel strip’ from the metaphorical mixing console to the left.

largely similar DAW although it forgoes the Mixer view, with mixing controls embedded in the track view alongside each track.

Notator (figure 3.11) is an older Atari system which later evolved into Apple Logic. Focused on the recording and manipulation of MIDI, Notator is a ‘pattern sequencer’, where users create a series of ‘sequences’ each of which consists of musical material broken into a number of tracks. Since pattern sequencers are non-linear in nature, these sequences stand alone, and can be triggered interactively, but can also be chained into a linear arrangement to form a complete piece.

Ableton Live is a modern DAW which borrows elements from both the linear and non-linear sequencing traditions — providing a multitrack view of audio and MIDI material within a single linear timeline (figure 3.12), and a modified mixing desk view (figure 3.13) underneath rows of sequences (called scenes) that can be triggered interactively or in response to rules. As suggested by its name, the interactive triggering supports a ‘Live’ manner of working for both the studio and stage.

Figure 3.10: Steinberg Cubase’s project window, also showing a number of horizontal ‘tracks’ built from blocks of audio and MIDI. Time runs from left to right, and a global overview of the entire track duration is presented above the main display. [128]

Figure 3.11: Atari Notator’s main screen. From left to right: the ‘Arrange’ pane shows the order of sequences to be played, the sequence pane shows the list of tracks in the selected sequence, and the track pane shows the properties of the selected track.

Figure 3.12: Ableton Live’s arrange view. As with other multitrack DAWs, tracks are stacked from top to bottom with time running to the right, and an overview of the timeline is shown above.

Figure 3.13: Ableton Live's mixer view. Each column represents a channel as it would on a physical mixing desk. Non-linear musical material can be placed in the rows above the mixer and triggered interactively.

The MPC sampling drum machine was also popular with several of our participants. This physical device, with a very small character display, features a 4x4 grid of touch sensitive drum pads for input, and allows recording of rhythm gestures into sequences. As with other non-linear sequences, these can be triggered interactively, but can also be chained into a single linear arrangement. The MPC was originally designed for triggering and recording of drum samples, but has become a mainstay of HipHop and electronic music, where various types of musical material are sampled and triggered to create entire songs and supplement live performance.

There are a number of ways in which modern DAW systems diverge from physical multitrack-mixer systems. Several of these stem from the added possibilities enabled by visualisation of multitracked tape. The most basic element of this is the visualisation of audio as a waveform in 2D space stretching over time. The waveform is a conventional metaphor for computer tools that deal with audio. Originating from oscilloscope displays, this metaphor is based on its component orientational metaphors *AMPLITUDE IS HEIGHT* and *THE PASSAGE OF TIME IS RIGHT*. The characteristics of the graphical

depiction allow the experienced viewer to read the audio to pin-point the location and nature of events visually, and edit them through graphical selection and the application of tools. With material now visually stretched over a spacial timeline, DAW systems can also allow visual markers to be placed at various temporal locations for immediate recall.

While controller automation is another entailment of the multitrack-mixing metaphor faithfully retained in modern DAW user-interfaces, it could now be visually overlaid on the audio representation or placed on a timeline alongside as a graphically editable time-series graph.

Just as with tape, lengths of material can be split and spliced back together, but unlike tape, material lies on a track in blocks and described by terms such as ‘regions’ or ‘clips’. These blocks can contain either MIDI or audio data, depending on whether the containing track is a MIDI or audio track. These blocks can be rapidly dragged and dropped to new locations on the timeline or even to other tracks — actions that while possible with physical tape, had been immensely more difficult to achieve. Some DAWs create the distinction between two types of automation: those that are associated with the track, and stay static in time independent of the movement of audio blocks; and automation that is attached to specific blocks.

Another difference from older multitrack-mixing systems is that the virtual mixing desk has the added task of mapping ‘real world’ hardware inputs and outputs to individual virtual channels in the system.

3.3.3 Multitrack-mixing model critique

An initial critique of the multitrack-mixing model raises a number of issues that can help to motivate our enquiry. Firstly, as we have already described, this model has evolved piecemeal from existing physical systems, and as such embodies many limitations of those systems. This can be a problem as:

“...unconsidered features are likely quietly to impose their consequences on the execution of the tool’s tasks for many years.”

— Keane [61, p.117]

For example, the requirement of tracks lasting the duration of the entire piece is the result of unquestioned assumptions carried forward to the very nature of multitrack tape. Secondly, multitrack-mixing systems have been developed to suit traditional groups of musicians. The mapping of instrument parts to channels and tracks is a natural one. However, in digital production environments there can be an unlimited numbers of virtual instruments, arbitrary audio sampling of sound-bites, and often no musicians to demand they appear for the duration of a whole track. Finally, the uniform reliance on this model raises the question of what other yet unexplored conceptual models and abstractions could make possible. Other researchers have also pointed out the potential limitations of the multitrack-mixing model and further the cause of exploring other potential organisations:

“This [multitrack-mixing] model has little to do with the traditional process of creating music. So, although the multi-track recorder metaphor is appropriate for the mixing and audio post-production stages, it provides little leverage in capturing and developing musical ideas.” — Abrams et al. [4]

“Because multitrack recording had become the dominant process for producing popular music from the 1960’s onward, the designers of software-based sequencers have, since the mid-1980’s, used the multitrack tape recorder both as a model of composition and as the prime metaphor for the sequencer program’s user-interface.... The sequencer can draw the user’s attention to selected aspects of the music-making process that are microscopic in their level of detail, but it does so in a fragmented and disjointed way.” — Théberge [133, p.228]

“We have seen strategies used in the design of computer music facilities that owe more to past experience of musical notation

or *early analogue hardware* than to what could be identified as the relevant needs of the system or its operator. . . . Otto Laske’s statement that designing a computer system for electroacoustic music is more the concern of cognitive than of acoustic engineering is certainly correct.” — Keane [61, p.117] (Emphasis added)

3.3.4 Abstraction mechanisms atop the multitrack-mixing model

While the multitrack-mixing model remains remarkably true to its origins, it *has* seen a variety of conceptual additions to take better advantage of the flexibility enabled by digital implementations. Some of the more important additions are facilities allowing new abstractions to be represented within the multitrack-mixing model.

Object editing

As opposed to the effect processing signal-flow model, the object editing model allows the user to control processing by specifying chunks of material to be processed rather than an entire stream of material. Object editing models can be seen in systems that allow what is sometimes termed ‘inline editing’, where effect processing units are applied to portions of material. In this model processing is applied directly to segments of material and processing units do not exist independently of that material or the parameters controlling them. This equates to Dannenberg’s Instance model [52], although the results of object editing are often cached.

Grouped effects

Several software systems allow for a second order of grouping of effects, beyond that of placing them on shared channels. In such systems any number of processing units can be grouped together and treated as a single unit. This can help mitigate complexity, as well as serve as a unit of reuse. Figures 3.14

and 3.15 show three effect units modularised into a single ‘rack’ object, and figure 2.11 shows a similar abstraction mechanism from another system.

Figure 3.14: Three effect units chained together in Ableton Live.

Figure 3.15: An ‘audio effect rack’ containing the three effect units in figure 3.14, serving to abstract away the details. Each metaphorical ‘knob’ controls a macro — a mapping from the knob’s value to any number of controls contained inside the rack, allowing custom ranges and inverse mappings for each control. Thus a small number of simple knobs can control a number of hidden variables in complex ways.

Layers

Another processing abstraction is found in applications such as Apple Soundtrack Pro, which uses the term ‘Action layer’. This abstraction presents audio processing as ordered layers lying on material. Individual layers can be turned on and off, and reordered without disrupting the underlying representation of material. Figure 3.16 presents an example.

Figure 3.16: Apple’s Soundtrack Pro allows effects and edits to be placed in ‘action layers’ that can be reordered or removed individually.

Temporal regions

DAWs often provide a marker facility to bookmark various points on the timeline that can be recalled. These can be used to divide a piece into various sections by placing named locators at their start point.

Ableton Live provides a ‘scene’ facility that allows a number of clips to be triggered simultaneously by a single mouse click or MIDI event. Figure 3.17 shows a session organised into vertical channels / tracks and horizontal scenes. Triggering a scene such as ‘Verse’, or ‘Breakdown’ will cue all the appropriate clips on that row. This allows the user to work in terms of song section abstractions.

Group and hide channels

Producers can use a group and hide technique to represent the concept of a voice created from a number of component tracks. In these systems, any number of related tracks such as ‘vocals’, are selected and put into a ‘group’. All of these vocal tracks are then routed to a single new channel, which in this case might be called ‘Vocals Master’. The group of vocal tracks can then be hidden with a single click in a group list. The grouping status can be used

Figure 3.17: An Ableton Live session view showing material created by the author in a band rehearsal with collaborating musicians. Various song structures such as ‘Verse’ can be triggered in real-time to synchronise with live instrumentalists.

to ensure that certain types of manipulation to one track of the assembled voice abstraction will affect all the others.

Folder tracks

Some systems, such as Apple Logic and Steinberg Cubase provide ‘folder track’ mechanisms that go a step further than group and hide in representing a number of voices as one. When multiple tracks are placed inside a folder track they can in many ways be manipulated and viewed as a single entity. Like the group and hide mechanisms above, folder tracks provide a *conjunction* function, allowing the multiple sub-voices to all sound together. Folders can be contained within other folders recursively, and also used to organise song sections:

“A folder is a Region that can contain other Regions, much like a folder in the Finder that can contain other folders or files. One way to imagine a folder is as a song within a song. A folder can have as many tracks, featuring Regions, as required. . . . You could use folders to represent the parts of a song (choruses and verses). As in the Finder, you can place as many folders as you like within other folders, and within yet more folders . . . Another possible use might be to store different arrangements of a song in different folders, allowing you to switch between them rapidly.”

— Logic Express 7: Reference [6, 123]

Parts, takes, and playlists

In contrast to the *conjunction* function of folder tracks, many DAWs also provide mechanisms to support *disjunction* groupings, where only one of the sub-voices can sound at any one time. Apple Garageband supports multiple ‘takes’ abstracted inside any block of audio, only one of which will play at once, selected through a drop-down menu. Cubase provides ‘parts’ containing ‘lanes’ (figure 3.18), only one of which will play back at any point in time:

“... an audio part is divided vertically in lanes. . . . Only one event per track can be played back at the same time! This means that if you have overlapping events (on the same lane or different lanes) these will cut each other off” — Operation Manual: Cubase 4 [128, p.200]

Figure 3.18: Two ‘lanes’ in a Cubase ‘part’, only one of which can sound at once. [128, p.200]

Pro Tools provides ‘play lists’. Activating a playlist by group replaces the material on all tracks with other material that has been previously recorded on those tracks. While this abstraction mechanism differs from parts, it also allows distinct takes to be recalled instantly:

“Edit playlists let you take a snapshot of a track’s current arrangement of regions, thereby freeing you to experiment with alternate arrangements, returning as necessary to previously saved playlists. . . . You can create new empty playlists for recording or importing audio and MIDI. This can be useful for creating alternate takes of audio or MIDI, or for constructing alternate arrangements.” — Pro Tools, Reference Guide [12, pp.352–353]

Rendering and Freezing

When folder track mechanisms are not available or prove inadequate, producers can ‘render’ multiple sources of material onto a single track. This

allows producers to work with concrete audio. This technique can be used to give an approximation of abstraction as it does allow producers to edit the results of audio operations as a cohesive whole, without being required to deal with the underlying processing information.

In order to work around CPU limitations DAW systems are increasingly implementing features to allow producers to ‘freeze’ one or more tracks. The resulting audio from any virtual instrument or effects units that are processing the contents of the track are rendered and live processing is disabled. When the track is played back, the pre-rendered audio will be used, instead of needing to be calculated in real-time. This can remove considerable load from the system, allowing a project to contain many more heavily processed tracks than would be possible in a purely real-time environment. Freezing a track takes much less time than playing it back in real-time, but can cause a lengthy delay for heavily processed tracks lasting anywhere near the length of a typical song. Moreover, freezing effect processing locks effect parameters, preventing producers from making live changes when performing or editing.

One of the most sophisticated freezing mechanisms can be found in Ableton Live 6, in which frozen audio on the timeline can be split and moved. Figure 3.19 shows an audio track from Ableton Live with various freeze and edit operations applied. Firstly the original audio is shown. Secondly, a freeze operation is applied and tail audio resulting from effect processing becomes visible as a crosshatch. Thirdly, the audio (including crosshatched tail audio) is split and moved on the timeline. Lastly, the freeze is removed, perhaps to allow sample editing or new effects to be added, but the tail audio edits and exact effected audio results are lost as audio will be reprocessed in its new temporal context.

3.4 Summary

In this chapter we have detailed various models of music with a historical perspective. These stem from psychological properties of audio perception,

Figure 3.19: Four progressive states of freezing and editing in Ableton Live 6.

the historical development of music theory and common music notation, and existing research into digital representation of music. We saw the role user-interface metaphor has played in allowing the historical development of music production technologies to determine the current form of computer music production software, and detailed a number of key abstractions that have since been built on this model. These models and their constituent means of abstraction will be crucial in understanding the activity of professional producers that we explore in the following chapters.

Chapter 4

Methodology

4.1 Introduction

In the previous chapter we examined established models and abstractions for representing music and music production. In particular, we described the multitrack-mixing model that has become the ubiquitous framework now determining the means of abstraction available to professional producers. Our research question, introduced in chapter 1, explicitly states our goal of determining the interaction between these abstraction mechanisms and the work of professional producers:

How is the work of professional music producers affected by the particular abstraction mechanisms in the user-interfaces of music production software?

This chapter details the methodology we employed in addressing this question, including the details of our seventeen participants, and how we gathered and analysed twenty-nine hours of recorded interviews, many of which were conducted in the context of the activity, and over twenty hours of dedicated observation.

We also present a new research instrument. Our ‘activity interview’ [59] is based on the early activity checklist [85], but addresses the problematic

impact of its theoretical terminology.

4.2 An introduction to case study

This research was conducted as a collective case study, with the primary aim of identifying the aspects of abstraction mechanisms encountered by professional producers caused by the abstractions (or lack thereof) in music production user-interfaces. Our case study approach is based on Creswell’s ‘Qualitative Inquiry and Research Design’ [49], and exploits case study enquiry’s emphasis of developing an in-depth analysis of a particular activity. Case studies have a number of characteristic elements that are relevant to our study:

- Case studies are a form of qualitative research designed to gain an understanding of the bounded system that constitutes each case. This makes it ideal for analysing a specific activity such as music production. Case studies have a strong emphasis on acknowledging the context of the case, so research is well grounded in the particular activity in which it is based.
- In instrumental case studies “...issues with the case [are] used instrumentally to illustrate the issue” [49, p.62]. In other words, case studies are rich in examples, and express findings through specifics.
- Purposeful sampling is used to discover a mix of interesting, representative, or contrasting cases to learn from; rather than attempting to obtain statistically balanced or random distribution of participants. While this can impinge upon generalisability, a focus on interesting cases can lead to the most thought-provoking findings.
- Multiple sources of information are utilised, ensuring a fully rounded understanding of the cases, and reducing dependance on any one type of evidence. For example, observation of participants provides verification

of claims made in interviews, while simultaneously suggesting further avenues for interview enquiry. Other sources of research data provide similarly balanced trade-offs to complete the picture and can each have a place in case study research.

- Case study analysis typically consists of a detailed description of the cases examined, followed by a description of themes discovered and evidence-based assertions made by the researcher. Among other things, this provides a good basis for presenting problems or issues that are at work in the bounded system under examination.
- With collective case studies (in which multiple cases are studied), there is the implicit advantage of cross-case analysis. This allows the researcher to triangulate issues and themes discovered, and gain insight on why they occur in more than one case, or how their lack of appearance in some cases can be explained.

As a qualitative approach, case study research also inherits the characteristics that are inherent to qualitative enquiry. These characteristics as described by Merriam [101] are also well suited to answering our research question.

Qualitative research is typically focused on processes rather than outcomes or products. This makes it the preferred methodological framework for understanding how and why activity is conducted in the way it is, and the causes of success or tensions at work. There is a focus on the meanings that subjects give to their experience, and how they structure and understand the world. This is well suited to investigations of the concepts that subjects use in their activity, and how they systematise their world, which can be analysed in the context of the activity and tools with which they interact. Rather than claiming an objective research standpoint, the qualitative approach embraces the perspective of the researcher as the primary instrument of data collection. This allows the researcher to uncover and develop a theory of what emerges in the field, without a dependence on an *a priori* theory of

what will be found, while still ensuring claims are well founded in evidence. This is the preferred approach when trying to develop new understanding of a particular activity.

The fieldwork focus of qualitative enquiry provides a foundation in the authentic situation of the participants, ensuring an accurate understanding of the complexity and interplay inherent in real-world activity. This contrasts with the laboratory approach more typical of quantitative research, which would provide a more limited and often less generalisable picture of creative and open ended activity such as that found in music production. Descriptive accounts, typical of qualitative analysis, provide the best underpinning for an understanding of process and meaning, as described above. The inductive character of this type of enquiry provides ample room for the development of fresh theoretical understanding of the patterns uncovered during the research process, rather than being limited to existing frameworks and models.

This is required to develop a novel understanding of the central issues in computer mediated music production and the use of abstraction mechanisms.

This research will not attempt to uncover every possible aspect of music production. Delimitations that narrow the scope of this research include:

- A focus on expert users rather than new users.
- A focus on producers working in the popular music idiom. Issues in software design for avant-garde music composition were not considered other than to inform our analysis of professional production work and user-interfaces.
- Computer tools for typesetting or composing music for human performance were not examined.

4.3 The researcher

Due to the personal and interpretative nature of qualitative research, the position of the researcher must be acknowledged.

As the researcher I have a long history of engagement in musical activity, but am a newcomer to the world of computer mediated music production. Since the age of nine I have studied a succession of wind instruments, beginning with the recorder, and proceeding through clarinet to the alto saxophone which I have been playing on a regular basis for thirteen years. I received tuition in piano but only reaching a basic level of proficiency, and am a self-taught guitarist. My performance experience includes playing wind instruments in various orchestras, jazz big-bands (including university level), and jazz combos where I was required to sight-read, perform from memory, and improvise solos. I formally studied music performance and theory to the final year of secondary-school, and continued to take professional lessons for a number of years afterwards.

I compose and perform in an all-original five piece jazz, funk, rock and dub fusion band, playing guitar, saxophone and singing. The band regularly plays to crowds of over one hundred. This has given me a detailed understanding of collaborative and individual composition in a live band context. This includes an understanding of the roles that improvisation can play in both pre-performance composition and live performance contexts. It has also provided a view of composition and arrangement that is aligned towards the flexibility and improvisational nature of working with human instrumentalists who each develop a personal understanding of a developing composition. I have also worked as an instrumentalist in an additional band performing funk.

I have been exposed to digital recording technologies when recording a number of pre-composed songs with the various groups I have belonged to. This included observing and making suggestions to an engineer using Pro Tools multitrack-mixing software during both the recording and mixing stages of production.

I have worked as a software developer and usability consultant. I helped design commercial software for creating complex causality models for strategic planning and evaluation [60, 132]. Recently I worked in user research for

Microsoft, performing usability engineering work on Windows Vista.

My personal motivation for this study stems from my early forays into the world of computer mediated music production, and a deep dissatisfaction with what I encountered. I found a contradiction in that the conceptual model missed the richness and flexibility allowed when composing with traditional groups of musicians, while simultaneously being overly focused on supporting the recording of these types of traditional groups. Resolving this contradiction is my personal motivation for this research.

4.4 Research tools

Historically, case studies have used existing theory at various points in the study, and use of theory before data collection is well precedented [49, p.87]. There are two theoretical traditions with associated research tools that we draw from, which we introduced in chapter 2. The first is ‘activity theory’, and the second is the ‘cognitive dimensions of notations’, and in the sections below we describe how we used these as methodological tools. Specifically, interview scripts based on these theoretical models provide the foundation for our interviews, and help structure our analysis.

4.4.1 Activity theory

Activity theory provides a useful terminology, a framework for analysis, and a clear catalogue of the important components of human activity; particularly emphasising the importance of tools in mediating work. This makes it an ideal foundation for understanding the ‘*work of professional music producers*’ component of our research question. Activity theory is ideally suited to use in a case study analysis as it emphasises an understanding of the context of activity. The theory’s focus and definition of ‘activity’ as the unit of analysis also provides a mechanism for demarcating the bounded system constituting each case.

Some musical activities have already been analysed from an activity theoretic perspective to a limited extent.

Recently Burrows [32] used personal reflection to examine his experience of performing free improvisation, and used activity theory to frame the discussion. He argues that in this collaborative activity the music plays an important role as both the object, and also as a mediating artefact between collaborating individuals, which is the role typically provided by a tool. Burrows also discusses how, by definition, performers in improvisational contexts will not know the actions they will perform, or their order ahead of time. Additionally, success conditions such as knowing when to end the piece are only defined in the moment itself, and this decision is necessarily distributed between the participating subjects. More importantly, the performers' actions are determined "not by planning or conscious direction, but through a complex set of conscious and unconscious signals between conversants."

Fagerheim [63] applied activity theory to Jazz, but mainly as a means to also apply activity theory to the ethnographic *study* of Jazz. The motive here was to show how the two distinct activity systems of Jazz *performance* and Jazz *ethnography* are interlinked. Fagerheim looked briefly at Ellington's big-band and focused primarily on the division of labour in this group.

However, there has been no significant work in applying activity theory to the compositional activity of computer mediated music production.

We incorporate activity theory into our data collection through the use of our activity interview instrument, that we based on the activity checklist. We describe the activity checklist and interview in the two sections below.

The Activity Checklist

The activity checklist was presented in 1999 in a paper [85] published in the HCI journal *Interactions*. Its goal was to provide a more accessible formulation of key concepts in activity theory, for application to software design and evaluation. The paper was authored by Kaptelinin and Nardi, two of the leading proponents of activity theory's application to HCI, and

Macaulay, an HCI design practitioner.

The checklist consists of a number of points to consider when analysing an activity. There are two versions of the checklist which separate small variations tailored for use in either evaluation or design. The two checklists contain a large number of items, thirty-seven and forty-three items respectively. Examples of an evaluation and a design checklist item are as follows:

Eval 3.5: Use of target technology for simulating target actions before their actual implementation

Design 3.3: Possibilities for simulating target actions before their actual implementation

The checklist was designed to provide a resource encapsulating the conceptual system at work in activity theory in a concrete form. This was specifically aimed at making activity theory “more useful” for evaluation and design of interfaces. At the same time, the checklist’s authors suggest “the Checklist orients without prescribing”, which could reduce its usefulness in contexts where a more prescriptive resource is desirable.

The checklist is divided into the four categories of *Means/ends*, *Environment*, *Learning/cognition/articulation*, and *Development*, each of which are important higher-level concepts in activity theory.

The paper also includes a number of sample questions divided into the same categories, although not specifically tailored to evaluation or design. These questions are presumably derived from the checklist, and provide examples of analysis working above the level of individual checklist items.

The checklist paper contains a testimonial from Macaulay, who used the checklist as an integral tool in her research, and the following quote illustrates one of the ways in which it can be used:

“It gave me a quick way of relating experiences in the field to AT concepts. It helped me think about the kinds of data I wanted to gather, and the kinds of questions I wanted to ask.” — Macaulay [85]

In the years since the checklist's widespread dissemination it has seen only limited use. Moreover, when the checklist has been used, it has been applied in various ways. Turner and Turner [139] used the checklist as the basis for semi-structured interviews, but gave no details on how exactly they derived their questions. Cluts [42] did the same, although they used it to inform observation. Fjeld et al. [65] used the "sample questions" provided in the checklist paper to fuel their analysis. In contrast to Turner and Tuner, they answered the questions themselves rather than using them in interviews. They did not comment on how limiting their enquiry to the example "sample questions" might have affected their research. Irestig et al. [81] used the checklist as a framework for discussing case studies of prototypes from usage centred and participatory design methodologies. They argued that the checklist might have biased their discussions away from issues surrounding the management of "signs and symbols", which they felt to be of great significance and not adequately addressed in the checklist.

While Kaptelinin, Nardi, and Macaulay claim that the checklist makes activity theory accessible, the reality may be quite different. Brown [30, p.50] points to the disappointing state of the checklist's adoption, and suggests the following explanation for this:

"For all items, the language used is taken from cultural-historical psychology, hence the checklist is of little use to those without a basic knowledge of the activity theory framework, and this may be a reason for its limited adoption."

While this much seems clear, in our analysis the difficulties with the checklist run deeper. In appendix A.1 we identify and explain a number of problems with the checklist as it stands. We should emphasise that our aim here in identifying these issues is to work towards remedying specific weaknesses in the otherwise excellent approach and foundation we find in the activity checklist.

The activity interview

To address the weaknesses of the activity checklist we briefly outline how we developed the activity interview from the checklist for the purpose of this research. Further details were published in [59].

The general method for creating the interview questions consisted of a series of steps as follows:

1. Firstly, we systematically merged the evaluation and design versions of the checklist. It is non-trivial to identify the differences between the two versions without careful reading and note taking. Often Kaptelinin, Nardi, and Macaulay have placed the identical item in both checklists. In some cases the item is reworded specifically for evaluation or design. Some additional items are only included in one of the two versions. The unified activity checklist can be found in appendix A.4.
2. The next step was to reword the resulting items in everyday language that retained important activity theory concepts, and avoided wording from our specific music production domain for future use. The resulting set of questions were then checked for coverage over the original checklist items and that they were ready to be used in interviews.
3. We applied this method to the merged checklist, but excluded the reflexive items that looked at the design activity itself. The evaluation version contains thirty-seven items, and the design component of the design version contains thirty-four. After carefully amalgamating items that were rewordings from the different versions of the checklist we were left with forty-one unique items.

Our work outlined above resulted in reducing these forty-one items to thirty-two unique interview questions, which were tested with a pilot participant. More details and examples of how the activity interview questions were derived are available in appendix A.2 and figure 4.1. The full activity interview script can be found in appendix B.

		ACTIVITY INTERVIEW																																
		Goals & Contradictions							Tool use & Collaboration								Internalisation & Externalisation								Change									
		1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	
		1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	
ACTIVITY CHECKLIST	Means / Ends	1	█																															
		2		█																														
		3			█																													
		4				█																												
	5					█																												
	6						█																											
	7							█																										
	8								█																									
	9		█																															
	10																																	
	11				█																													
	12					█														█						█								
	13							█													█													
	14					█																												
Environment	1							█																										
	2								█																									
	3									█																								
	4										█																							
	5											█																						
	6												█																					
	7													█																				
	8														█																			
Learning / Cognition / Articulation	1																█																	
	2																	█																
	3																		█															
	4																			█														
	5																				█													
	6																					█												
	7																						█											
	8																							█										
	9																								█									
	10																									█								
Development	1																									█								
	2																										█							
	3																											█						
	4																												█					
	5																													█				
	6																														█			
	7																															█		
	8																																	
	9																																	

Figure 4.1: A matrix showing how our activity theory interview questions (appendix B) were derived from the unified activity checklist (appendix A.4).

4.4.2 Cognitive dimensions

In addition to activity theory, we also utilised the cognitive dimensions of notations as a theoretical framework informing data-collection. As introduced in section 2.5.3, the cognitive dimensions framework provides a terminology for various design trade-offs in notational systems, many of which become most pronounced in the interactions between the means of abstraction that a notational system provides. For this reason, the cognitive dimensions are a vital tool for eliciting the characteristics of *‘particular abstraction mechanisms in the user-interfaces of music production software’*, which is the second part of our research question.

The cognitive dimensions provide an ideal tool for our research, as they are specifically designed to account for information artefacts and the systems in which they are manipulated. As we described in chapter 2, professional music production software is a prime example of such systems. The cognitive dimensions of notations “...model all information artefacts in terms of notations and their use” [71, p.8]. As such, it allows us to focus on the notational representation of digital artefacts for music representation, and how their characteristics affect the work of music producers. As the cognitive dimensions have been used in the analysis of other forms of musical representation (as described in section 2.5.3), we have good reason to expect them to prove useful for our purposes. Further, our use of the cognitive dimensions perfectly complements activity theory by addressing its weakness by dealing directly with the issues of “signs and symbols” identified by Irestig et al [81]. From this methodological perspective, the cognitive dimensions complement traditional HCI methods by providing tools for describing an extended set of phenomena:

“...existing HCI had no way to describe the structure of information, and therefore had no way to analyse the interactions between: a) the structure of the information, b) the environment that allowed that structure to be manipulated, and c) the type

of activity the user wanted to perform.” — Blackwell and Green [21]

The cognitive dimensions questionnaire

The cognitive dimensions originated as a design and discussion tool. However, two of the original authors of the cognitive dimensions have since created a questionnaire as a resource for gathering data directly from domain experts [20, 22]. This questionnaire has previously been given to users of music scoring systems [23], and spreadsheets [138].

In order to better fit with our case study approach, we decided to apply the questionnaire in a semi-structured interview format. This allows the interviewer to clarify and rephrase questions, express questions in terms of the participant’s specific situation and software, and ask followup questions.

While this resource was originally designed for use in questionnaire form [20], it has since been used in an interview format by the original authors [22]. However, in even in interview format, it was used as a non-interactive questionnaire with minimal feedback from the interviewer. As we used the questionnaire as an interview tool in a highly interactive semi-structured fashion we were able to elicit highly detailed responses from our participants.

One issue we encountered is that many of the participants had trouble with the initial questions regarding the proportion of time they spent performing different types of action. In our interviews, participants felt that the actions overlapped: for example “playing round with new ideas” could overlap with “adding small bits of information”. Secondly, several participants showed confusion in trying to get the percentages to add up to 100%, despite our reassurances that this was not necessary given the overlap. Lastly, participants expressed that their different phases of activity had radically different biases on the proportions they would ascribe to these, and it was difficult, and perhaps not even useful, for them to give an overall rating. A different approach that might have yielded more meaningful responses would be to ask participants to rank the frequency of the different types of action,

or grade frequency on a scale, and ideally allow them to break this down into different phases of activity. This would work best when used in tandem with an activity analysis such as that enabled through the use of our activity interview.

The most significant benefit of our choice to use a semi-structured interview format (rather than the traditional cognitive dimensions *questionnaire* format) was the apparent advantage this offered in that the researcher could clarify questions and follow extremely productive lines of enquiry. Moreover, conducting the interview with the participant in front of their computer allowed them to provide detailed examples which opened avenues for further exploration. Clarification and examples were very important in helping participants understand the full intent of many questions. For example, terminology was a problem with terms such as “description” that were not defined for the participant. Explaining what was meant by “notation” was best accomplished with specific reference to aspects of the user-interface used by the participant. Even terms that *were* defined could be easily confused: “helper devices” were sometimes mistaken for applications’ help systems, and the questions about “mental effort” (meaning cognitive load) could easily result in answers about the number of actions required. In some cases questions would take unexpected turns — when one participant was asked about when they “don’t want to be too precise” they ironically described the extremely high precision work required to make musical material sound ‘sloppy’ and as natural as a human performance. To learn about other truly imprecise editing required further questioning and refinement on the topic, something that would not have been possible in a non-interview format.

In many cases, participants would give just one answer to a question, when there were many other aspects of the notation which might also be relevant. Asking participants about these gave much more complete results, and was vital for corroborating the reports of multiple participants.

4.5 Data collection and analysis

This research was based on a case study methodology, with theoretical underpinnings in activity theory and cognitive dimensions. This section describes the particulars of data collection and analysis used in this study.

4.5.1 Ethical considerations

This research has been cleared by the Victoria University of Wellington Informatics Human Ethics Committee and is identified by the HEC approval reference number IHEC-20052-020. In accordance with the terms of this approval, the researcher was cleared to collect notes, record audio, capture images and computer output, observe work, and *attribute data to individual non-anonymous participants*. However, this was on the condition that the participant had read, agreed to, and signed the consent form itemising the particular rights they agree to. After the study's completion, participants who have opted to will receive a summary of the study results, and any unpublished data will be deleted after three years. All participants agreed to full attribution except one, who required confirmation of attributed quotes. This additional consent was completed once the quote selection was determined.

4.5.2 Data collection and recording procedure

In the case study and activity theory tradition, we collected data from a range of sources. This section describes each of the sources used, the rationale for collecting from these sources, and the procedure for recording the research data.

- **Semi-structured interviews**

We conducted semi-structured interviews with computer music producers, based on my activity interview as well as the cognitive dimensions questionnaire. The base interview questions were created from a novel set of interview questions developed in a principled fashion from the

Activity Checklist as outlined previously, combined with the cognitive dimensions questionnaire used in interview form. Thereafter the result was a total of one-hundred and three questions, sets of which would be left out as appropriate to particular interview contexts, and depending on the demands of the research. After the first round of analysis was complete, targeted followup interviews were conducted to gather further data on issues lacking sufficient evidence.

Collection rationale

Interview data provides insight into the subjects' understanding of the abstractions and the causes of issues they face. The theoretical underpinning of the interview provides a solid starting point to ensure broad coverage of potentially relevant issues. Semi-structured interviews allow subjects to drive discussion in unpredicted directions important to them, and for the interviewer to follow relevant leads.

Recording procedure

Interviews were digitally recorded and transcribed to computer hard-disk. Where possible, interviews were conducted in participants' studios to provide real-world examples of current and past projects illustrating discussion. Interviews resulted in twenty-nine hours of recorded audio.

• Observation

We conducted observations of professional producers in the studio, asking them to explain interesting actions and operations relating to the research goals as they occurred, or after the session had completed, depending on the circumstances of the particular musical activity.

Collection rationale

Observation acts as a reality check to subjects' claims during interview, and provides an opportunity for the researcher to clarify

these with the participant when they diverge. Moreover, observation exposes actions and operations the subject may not describe in interviews, which can provide a basis for further questioning or be used as direct evidence.

Recording procedure

Observation notes and typed quotes were transcribed to computer and digital photographs of activity and tools were taken during research sessions.

- **Analysis of music production software and documentation**

Collection rationale

The user-interfaces of existing music production software are themselves a form of data for analysis. The presence of various abstractions and their behaviour in the system-model are important for research claims, validating subjects' interview claims, and motivating further questioning. As a complement to studying the systems themselves, the highly detailed manuals explicitly describe both the abstractions present in the user-interface, their behaviour, their intended usage, and in some cases the rationale for their inclusion in the system.

Recording procedure

Our written analysis of the user-interface conceptual models present in various music production systems was recorded over the course of the research for use in the data analysis phase. Relevant quotes from manuals were also recorded for analysis. The researcher's resulting knowledge of systems was also used constantly in semi-structured interview and captured in subjects' responses as a result.

- **Analysis of producers' tool customisations**

Collection rational

Any customisations that subjects make to their tools can expose patterns of use, as well as the limitations of existing tools. Producers' tool customisations found during observations sessions can reveal possibilities for improvement and prompt further dialogue with participants.

Recording procedure

Significant customisations were photographed where possible. Interesting customisations also lead to further interview questions as above, and notes were collected for analysis.

• Participatory observation

Collection rational

Due to the complex nature of the activity system in question, it would be almost impossible to comprehend without a detailed personal knowledge of the domain. This comprehension is vital for ensuring the researcher is able to follow observed behaviour and interview discourse, ask appropriate questions, and perform informed analysis. In order to achieve this personal knowledge I conducted a form of participatory observation in which I started composing, and playing with musicians using the music production software Ableton Live and Propellerheads Reason, with a variety of effect and instrument plug-ins. Through this firsthand experience I was able to directly experience some of the issues that participants dealt with on a daily basis, better comprehend the vital significance of workarounds that professionals developed, as well as discover leads for further avenues of enquiry.

Traditionally, participatory observation involves the researcher participating in the activity *with* the research participants. Differing from this somewhat, I participated in the same type of activity

system as my participants, without working with them directly. This personal experience enabled me as a researcher to strike a healthy balance between insider and outsider.

Recording procedure

Detailed notes were kept of insights generated through composing and playing with the group. A text editor kept open in the background during these sessions allowed ideas to be captured when required for use during data analysis.

4.5.3 Data analysis procedure

Interviews resulted in twenty-nine hours of recorded audio, and coded notes and quotes from the sources described above, totalling over fifty thousand words.

Yin [148] described three possible forms of data analysis particularly suitable for use in case studies. This study employed the first two: the searching for patterns where research data interrelates to theory and literature; and explanation building, where new explanations and theory arise from analysis of the data itself. The third (time-series analysis) was less relevant to our investigation as it was not a longitudinal study. How these two techniques were employed is described in the two following sections.

Research data analysed through theory

As a first attempt to analyse the data, I developed a software tool called ‘Interview Analyser’ (shown in figure 4.2) that allowed the response from each interview question to be compared side by side across multiple respondents. As each interview question stems directly from particular aspects of the theory of cognitive dimensions or activity theory, each response should relate directly to that aspect of theory and submit easily to cross participant analysis. While this method and software tool proved useful for early analysis of responses, as well as evaluating patterns in the success or failure of particular

interview questions; in practice, respondents answers were too unstructured, and discussions relating to particular aspects of theory were spread throughout interview transcripts. Additionally, notes from observation and other sources, as well as unstructured interview needed to be analysed.

Figure 4.2: A software tool the researcher developed to help in the analysis of interview responses.

For this reason, all textual research data was loaded into a qualitative data coding tool called ‘TAMS Analyzer’ [143], and notated with a number of a priori theory driven codes. These included fourteen cognitive dimensions codes (such as **closenessofmapping** and **viscosity**), codes based on computer music ontology theory [52, 27, 95] (such as *voice*, *temporal* and *pitch*), codes relating to activity theory (such as *internalisation*, *externalisation* and *collaboration*), and five codes from the computer music software taxonomy [58] developed for this research prior to field research (such as *abstraction*, *linearisation*, *flow* and *medium*).

Codes emerging from research data

The second aspect of case study data analysis is the creation of new explanations and theory from my immersion in the data. As the data was being coded with the theory driven codes described above; sections in quotes, observation notes and other data were found that were clearly interesting from the perspective of the research goals, but which required new codes. In the qualitative coding tradition, new issues encountered in the data allowed me to add additional codes to the set (such as *rendering*, *optiondilemma*, and *assetmanagement*) and mark-up the data appropriately. Some of these issues became obvious during actual data collection itself, or during preliminary analysis between fieldwork sessions, and so naturally became part of semi-structured interview and followup questions in consequent sessions.

Data organisation and analysis

Once data collection and coding was complete, the TAMS analyser software was used to perform queries on the coded data. Closely related codes were performed as a group query. Each data item (most often a quote or observation note) was sub-categorised under headings and copied into a text file. The end result of this process was four sub-categorised files of organised evidence speaking for and against various issues raised in fieldwork. This was then pared down to the most convincing evidence, with redundant and ambiguous data items removed; and reordered to serve the appropriate narrative flow. The remaining content led directly to analysis and written discussion, framing and describing the issues at play which became the foundation of this thesis's analysis chapters, and sub-categories defining section structure.

Sections with only marginal single-case evidence or insufficient cross-case support were subject to verification by a series of targeted followup interviews with existing and additional participants. Each followup interview question was designed to determine support for or against a particular issue described in the preliminary analysis. As a result, a number of issues were removed

from the analysis as lacking sufficient evidential support, while others were strengthened and so marked for inclusion.

In order to make the most important findings easily available to practitioners, major points were summarised as terse ‘evaluation questions’. These are presented in the main body of our analysis immediately following the relevant evidence and discussion, and the full list is included in appendix C. These summarise the findings in a way that can lead to a better understanding of how these abstraction issues are dealt with when the questions are applied to any particular music production user-interface.

4.5.4 Case selection and participants

Case studies must make a trade-off between the depth of study allowed by concentrating on a very limited number of cases, and the generalisability of findings allowed when working with larger numbers of cases. For the purposes of this research we require both a depth of enquiry to match the complexity and depth of the activity system in question, while maintaining some reasonable degree of generalisability. This more broad applicability is also desirable given our aim of applying our findings to the benefit of the majority of computer music producers, and accordingly, this trade-off was particularly undesirable. To address this, I took a dual approach, with an in-depth focus on five core cases, while adding breadth with twelve auxiliary cases each visited in less depth.

Initial participants were recruited through personal contacts in the New Zealand music industry and cold calling eminent producers. Further participants were then found through recommendations from those primary participants. Additionally, academic contacts provided a number of participants involved in experimental computer music production. As described in section 4.2, case selection employs purposeful sampling. In this tradition, the five core cases were chosen as good representations of a small variety of typical producers engaged in computer mediated production in the popular music idiom:

Andy Morton: Hip-hop producer with credits on popular New Zealand artists including Che Fu, Feelstyle, and Lady-6, as well as producing music for film. Andy is interesting in that he uses a variety of hardware and software music systems from different eras. These include modern Pro Tools and vintage Atari Notator software (which he has been using for sixteen years) along with sampling hardware and large mixing console, the MPC and vintage SP1200 sampling drum machines. Andy is also a DJ, and on occasion employs DJ techniques in his production work. Andy participated in activity theory and cognitive dimensions interviews, extended interview discussions, as well as observations with him composing and editing new and existing material.

Christiaan Ercolano: Hip-hop producer, and artist for the electronic house and funk group House of Downtown, who have released two commercial albums. Christiaan is a long time advocate and user of the MPC sampling drum machine (both legacy and modern versions) having been using it for fifteen years. He runs a training school tutoring young people in how to use the MPC system. In his own work, he uses the MPC for composition in conjunction with the industry standard Pro Tools DAW sequencing software and has a deep understanding of the strengths and weaknesses of each. Christiaan participated in the cognitive dimensions interview, extended interviews, and observation of composing new material.

Emile De La Rey: A primary member of the group Sstimuluss, Emile uses Cubase to compose and accompany live musicians, and records and edits their studio performances to create finished pieces. Composing in a range of electronic styles from drum'n'bass to ambient and processed acoustic works, Emile also records his own piano and vocal performances. Emile is studying electro-acoustic composition at university, although his primary musical output is clearly in the popular idiom. Emile took part in the cognitive dimensions interview, extended inter-

view sessions, and observations primarily editing existing works.

Paddy Free: Producer with production credits with a large number of well known New Zealand bands including Salmonella Dub, Supergroove, and Crowded House. As one part of the electronic music duo Pitch Black, Paddy has released more than thirteen commercial albums and performs live around the world, primarily using Apple Logic and Ableton Live. Paddy is a perfect exemplar of a subject working within the activity system under examination: composing, mixing and performing with computer systems. He has more than twenty years of experience with electronic music systems. Paddy participated in the activity theory interview, cognitive dimensions interview, hours of extended interview discussions, and observations while working on tracks for an upcoming album.

Troy Kelly: Independent producer and musician who owns and operates a recording studio working with up and coming bands. Troy plays upright and electric bass guitar, and also composes his own music using Ableton Live and Reason, as well as jamming and performing via a computer with live musicians. He uses both Cubase and Pro Tools in a studio setting for recording and mixing bands. He has been using various sequencing systems for seven years. Troy participated in the activity theory interview, cognitive dimensions interview, observation sessions and a targeted followup interview.

In order to supplement these five core cases with greater breadth, twelve additional participants were recruited to provide auxiliary cases. Seven of these were relatively typical computer music producers:

Andrew Turley: Member of ChampionSound. DJ who produces his own tracks by resampling other works through Ableton Live and Reason, and works live with an MC, performing the down-mixed tracks from a DJ interface. (Activity interview)

Clinton Bradley: Professional international producer working with a wide variety of artists such as Nine Inch Nails. Clinton primarily uses Apple Logic with a wide variety of plug-ins. (Targeted followup interview via video conference)

Eric Vani: Canadian producer who works under the alias Rise Ashen. Has released a number of original albums featuring a variety of electronic and acoustic sources, and performs live. Eric has a professional qualification as a studio engineer. He uses tools such as Acid, Reason, and others. (Composition and editing observation sessions)

Jeremiah Ross: Known as Module, producing electronic music featuring a variety of sampled, recorded and synthesised elements for commercial album release and live performance. Primarily uses Ableton Live and Acid. (Extended interview and observation)

Jeremy Brick: Jeremy is an electro-acoustic composition student using Pro Tools. (Cognitive dimensions interview)

Jim Frazier: Professional producer working with a variety of groups from his studio. Jim primarily uses Apple Logic. (Targeted followup interview via email)

Simon Rycroft: One of the two founding members of the popular electronic-live band Rhombus, and commercial producer. Primarily uses the MPC, Pro Tools and Ableton Live to compose and collaborate with a large band of instrumentalists and vocalists for commercial album release and live performance. (Activity interview and targeted followup interview)

The remaining five participants were chosen for their contrasting nature, either the tools they were using, or a variation in their activity system that might raise novel ideas for analysis when compared to the main cases. These contrasting auxiliary cases are as follows:

Alexander Jensenius: MAX/MSP developer. Alex creates experimental electro-acoustic works using MAX/MSP visual programming system. His use and expertise of the MAX software system provides a counterpoint to the dominant sequencer systems in our primary cases. (Cognitive dimensions interview)

Nick Collins: Experimental live coding performer. Nick is part of an innovative British live music programming group called TOPLAP. Using the domain specific programming language SuperCollider, Nick creates compositions in real-time by programming in its textual language. This is conducted in night-clubs and other venues where he utilises a real-time beat cutting software library he developed as part of his PhD research at Cambridge university. We can learn much about the limitations of visual systems through comparison with textual music programming systems. (Cognitive dimensions interview)

Alex McLean: British Perl programmer and member of TOPLAP. As with Nick Collins, Alex performs live dance music by programming on stage in the Perl language. Alex uses a live perl interpreter he developed to create musical scripts on the fly, some of which are programmed to edit themselves and thus manipulating their own musical output programmatically. Nick's use of live music text programming provides a second contribution to our analysis in the same vein as Nick Collins' above. (Cognitive dimensions interview)

Tim Prebble: Film audio producer. Tim works principally with sound rather than music; primarily creating, editing and sequencing sounds for commercial cinema. His use of speciality audio library management software and techniques makes his work particularly interesting for comparison with our music producers. He uses Pro Tools, the same system used in many of our other cases. (Cognitive dimensions interview)

Wayne Laird: Classical music post production editor. Wayne works with multiple takes of orchestral performance recorded in multi-channel audio, splicing them into a single flawless performance for commercial release. His use of Sequoia, a novel audio editing software system, makes for interesting comparison to other systems. (Cognitive dimensions interview)

In addition to the seventeen participants in the study, a pilot participant was also used in refining the activity theory and cognitive dimensions interviews. The pilot participant is a graduate industrial design student developing a novel DJ hardware interface for live performance of original compositions. He had himself interviewed a number of professional DJ-producers and provided an interesting perspective.

Chapter 5

Activity Analysis

5.1 Introduction

In this chapter we describe computer mediated music production from an activity theoretic perspective, based on our interviews and observations in the field. This chapter provides an overview and framework for understanding this activity, which will become the foundation for the following four chapters. We begin by defining the activity in question and explaining the role that performance, improvisation, and preparation play in both studio and stage contexts. We then explore the components of the activity, including the subject, tool, and object of activity; the mediating role of tools in music production; producers' actions; the varying and flexible roles of our participants; and the importance of internalisation and externalisation.

We then introduce four aspects of abstraction that we found to be of central importance throughout the various actions of our participants. The following four chapters will then provide detailed examples of each, and issues we have identified through our field research. We conclude this chapter by presenting three reoccurring themes (the consequence of rendering, option dilemma, and conceptual burden) that were found throughout these four aspects of music production abstraction.

5.2 Computer mediated music production

Much of the music we listen to today is recorded and manipulated with computer systems. In many cases, these computer systems are playing an increasingly large role in mediating the entire creative process rather than simply acting as a recording and post-production technology. We are primarily interested in those activity systems that are towards the computer mediated end of this spectrum, where the entire development process from initial composition and concept formation through to the arrangement, mixing, and performance are heavily informed by computer music systems. To this end, we studied the activity of computer music making in the popular idiom. To contrast, in the traditions of experimental and avant-garde computer music activity, subjects constantly seek to subvert and transcend established tools and techniques, which makes activity analysis and tool design a difficult and case by case proposition. Instead, our investigation of ‘popular’ forms of computer based music making — those with origins in the traditions of blues, jazz, funk, pop, and rock; now found in the many common varieties of electronica such as dub, hip-hop, house, industrial, drum and bass, and much of modern pop music — is more readily analysable and amenable to general observations informing activity development and tool support. This activity will here be referred to as the activity of ‘computer mediated music production’.

As we shall see, music production has particularly interesting characteristics that differentiate it from many other forms of activity:

- Due to its creative open ended nature it is typical that end results are not predetermined and aspects of the activity are often ad-hoc.
- Role changes are very common, with people often taking on different responsibilities as required or desired.
- Music production is characterised by extremely heavy use of abstract representation in mediating tools.

- The object of the activity, representations of the composition, become a tool for later live performance.

5.3 Contradictions: Live versus studio, and composition versus performance

The activity system of computer mediated music production envelops both studio and live performance contexts. The importance of improvisation and composition in both studio *and* live performance make it impossible to analyse them independently. A tension here between composition and live performance is an important contradiction in the activity that must be addressed in tool design.

For comparison with our activity system, consider a band of composer-instrumentalists. In such groups there are two ways in which composition can take place: through the individuals contributing material they have personally composed, married with the group interactively exchanging composition ideas and improvising together. For such groups there are a variety of points in time at which they might perform. Some improvise the composition entirely in front of an audience, while others have pre-prepared and carefully rehearsed every detail of performance. Most bands fall at some point in between, with many rock bands improvising just solos and embellishments, while jazz groups often use the ‘blues in E, follow me for the changes’ approach, closer to the improvised end of the scale. What this example makes clear is that the relationship between composition and live performance is a flexible one, in which the two can be intertwined in a variety of ways.

This contrasts radically with the composition / performance divide present in the classical tradition, where material is composed in one environment with a notation system, and then passed on to performers who learn and then perform the piece (although conductors and performers are also afforded a relatively small degree of improvisation and interpretation):

“[Classical composition was], from the outset, characterized by a relationship to time that was different from performance; with notation, not only was the musical work preserved in a concrete form, but musical time itself was represented in a spatialized pattern. The ‘urgency,’ anticipation, and shared sense of time characteristic of performance was replaced (for the composer at least) by a detached set of quasi-mathematical calculations and operations executed with little reference to ‘real-time’ modes of action.” — Théberge [133, p178]

While bands exhibit a more flexible dynamic between composition and performance, many modern sequencers, and the consequent activity systems of subjects, have inherited this classical division between live performance and composition. This is largely a result of the post-production based history of these tools, resulting in the multitrack-mixing model. For our participants there was a contradiction within the design of systems that force this dichotomy as they lacked the ability to rapidly move between performance style composing (or ‘jamming’ as it is often termed) and more calculated editing style composing:

“Cubase isn’t really made to improvise on. The big thing is having to start and stop your audio to make changes. In a live situation, if I’ve got this loop which I am playing over and over again, I want to vary that. It is quite hard to do on the fly.... Often I’m needing to record as we jam.... If I am jamming with a band, a good jam tends to come out of nowhere, you don’t know when that is going to happen. If the band is jamming something, I am often left scrambling around on the computer...”. — Emile De La Rey

Emile went on to describe how his experience composing with others could be hindered by his software tools, and how it contrasted with composing in a purely instrumental band situation:

“[In a band] I could just jam [but with a computer there is] more boring technical work than if you’re just standing around jamming. Sometimes you just lose the creativity. It would frustrate [my collaborator, whereas] I’ve trained myself just to postpone my creativity.” — Emile De La Rey

This betrays a vital point underlying the opinions and perspectives of research participants steeped in their tools and the limitations inherent in them. Because these subjects have by definition trained themselves to work around or just deal with the limitations of their tools, they have in many ways accepted them. In this case, the frustrations felt by a collaborator who is more used to the immediacy and flexibility of a composing band environment, exposes the real limitations of the live versus studio dichotomy. In the heat of the moment, there may be no time to prepare for live interactive performance. This is true even in a studio environment, because composition will often require some variety of performance. Andy Turley also found this contradiction, created by the software, frustrated his desire to blur the distinction between composition and performance:

“Ideally, what I’d like to do is [compose] live with [my vocalist]”
— Andy Turley

Paddy Free spoke about how the “studio is more precise and less spontaneous than live set-up”. This lack of spontaneity drove him to prepare his material and ‘perform’ live interactively in the comfort of the studio with no audience in order to discover new compositional possibilities, and to integrate them into an unfolding and precisely defined arrangement:

“[With the live set-up] we’ve created the conditions for spontaneity, where you are spontaneous in different ways.” — Paddy Free

Jeramiah Ross also found it vital that he have the ability to interactively compose in a live environment, going as far as to prepare to perform improvised arrangements in front of an audience as part of the composition process:

“A lot of tracks that I write spend a lot of time in the live [performance] environment before they go anywhere near being [a solid arrangement].” — Jeramiah Ross

We revisit this aspect of live interactive performance in more depth in section 8.3.3.

The now-revealed importance of spontaneity in a studio composition environment provides an argument for removing the work involved in this preparation wherever possible, and ideally integrating the two entirely, to more closely match the activity system found with bands of composer-instrumentalists. While current systems for mainstream computer music professionals are weak in this area, experimental computer music research systems have long been moving to such integration:

“M sports a variety of mechanisms that let users work so interactively with the computer, that the distinction between composing and performing becomes completely blurred.” — Eduardo Reck Miranda [103, p.204]

“The distinction between composition and performance, or between notation and instrument is becoming increasingly blurred in contemporary music technologies.” — Blackwell and Collins [19]

“... [The] roles of composer and performer [are] roles that often become blurred in interactive works.... One of the new challenges facing composers of interactive works is to create malleable forms based on flexible musical structures that respond to human input.” — Todd Winkler [145, p.28]

5.4 Subject, tool, and object

In this section we describe the subject, tool, and object triad present in the computer mediated music production activity systems we observed. Figure

5.1 shows the activity triangle, and emphasises the mediating role of music software systems between the subject and the object of their work. The central human subjects in this activity are first and foremost “producers”, although they take on a variety of different roles in the course of their work, and also collaborate with specialists. The object of their activity is a musical composition, created through the medium of music hardware and software tools, most predominantly the Digital Audio Workstation software at the hub of this activity.

Figure 5.1: Music production tools play a vital mediating role between the subject and their musical object.

From object to outcome

Our participants were primarily engaged with creating and manipulating a representation of their musical composition inside the computer. This digital artefact acted as the primary object of their activity systems. The outcome of this activity was any number of audible results, including artefacts such as

session files for audio mastering, Compact Disks, or MP3 files for reproduction by listeners. Additionally, for many of our participants, a vital type of output was live performance of the composition. For these live performances the outcome was created in front of an audience through real-time manipulation of the object. In this case we saw the object itself become a tool, a phenomenon that is common to many types of activity system as described by Bødker [25].

One typical example of how a producer created multiple outcomes from an object was how Paddy Free used a digital representation of a composition to create an album version, a remix version, and multiple varying live performances of the same piece. Among others, Paddy Free, Andy Morton and Andrew Turley created remixes of other producers' tracks, and Simon Rycroft and Paddy Free created different versions of pieces during the production process for the individual musicians involved:

“[We] release different versions of the track to different instrumentalists for them to practice to.” — Simon Rycroft

Mediating tools

While all human activities are mediated by tools, music production is distinctively so. During the music production process the developing composition (the object) is held in a representational form inside various technologies, and participants' manipulation of that representation is entirely dependant on the facilities of these mediating tools. In order to work around the limitations of some tools, and exploit the strengths of others, participants expend great time and effort in moving representations between them. For example, Jeramiah Ross would move the digital representation of his compositions back and forth between Ableton Live and Acid in order to manipulate it in the way he required. In the transition between these tools, important aspects of the representation were often lost.

Music production has a profound focus on technology. Industry magazine articles and web resources are produced analysing in great depth the features

and possibilities of the new tools that are constantly being released. At the same time, some professionals feel that slavishly following technological developments can be a distraction from the important activities of music production:

“The technology changes, but it is just the same old technology repackaged.” — Andy Morton

“It’s funny because I’m not really that interested [in new music technology]... I haven’t really had much interest in finding out about new software... My focus is more just on getting the job done.” — Simon Rycroft

A wide variety of tools were used by our participants. Many of our participants bring to their work a large background of musical knowledge, training, and experience; all of which are tools in their own right. Instrumentalists used both their skills in performing and composing, as well as their psychical tools: acoustic or electronic instruments, in addition to their voices.

Audio was created in many ways, and not limited to the capture of the audio from acoustic instruments. Participants used samplers to play back chunks of decontextualised sound, and used synthesisers to electronically generate sound from scratch. All sounds, no matter what the source, were often radically transformed through the use of effects processing.

Participants and those they worked with controlled these sounds with physical gestures interactively through the use of instruments. Instruments used during observation sessions include drums, drum triggers, percussion, keyboards controlling synthesisers and samplers, control desks, electric and acoustic guitar, bass, flute and vocals. Participants also created instruments from tools with other intended purposes. For example, vinyl record players were manipulated by DJs such as Andrew Turley and Andy Morton to create and control sounds. Mixers that were originally designed for controlling volume and equaliser settings were used to radically and interactively con-

trol and shape sound, as was the case with Paddy Free, Simon Rycroft and Jeramiah Ross.

Participants captured audio from the physical environment with microphones, pre-amps, mixers, and analogue-to-digital converters. These tools were used in studio, live, and field-recording contexts. Once captured or generated, audio was transferred between tools with digital or analogue links.

Gestural and audio information was organised and augmented with a wide range of sequencer and audio editor tools. Participants used these tools to edit and create gestural semantic information without performing concrete gestures. Participants with computer programming skills took advantage of advanced programming tools to perform complex and abstract operations on this information.

Many tools used by participants combined several of the above functions. For example, the MPC drum machine used by Andy Morton and Christiaan Ercolano included a physical gestural interface for triggering sounds, a simple sequencer for recording and editing sounds, and a sampler for recording and reproducing sound. Computers, which are tools in of themselves, support software that provides these tools. Several participants, such as Alex McLean, Nick Collins, Paddy Free, and Jeramiah Ross among others used laptops for portability and use in live situations.

Participants augmented their tools by tailoring them to their own uses. Some examples we observed included creating new instruments, labelling and rearranging tools in their studio environment, making notes on their use of tools, and programmers creating their own computer music tools.

Between them, in our interviews and observations, the participants described their use of all of these types of tools. Each subject used a different set of tools and combined them in many ways, often changing their use on a project by project basis. Each of these tools has found its place in both live and studio contexts, often subverting its original intended purpose.

5.5 The subject and their roles

Subjects involved in music production must fill many roles. Our interview participants varied greatly in the specific roles they played, but all of their activity systems relied on other collaborating subjects. Our professional music producers often introduce other collaborators in order to help feed their creative process:

“Even when I’m making my own music I like to catch different peoples flavours.” — Andy Morton

In some cases participants would create music without active input from others, which demanded that the subject take on all of the required roles for the project. More interesting interactions, both for the artist and from the perspective of our analysis, occurred when more than one subject was involved as was often the case with the participants.

The most salient and overarching role that emerged from the interviews was that of the *producer*. The producer acts as a guide, both for the music production process and of the musical content itself:

“My role if I am producing a group, I liken it to a musical tour guide in that it is up to me to look for the best things that I relate to in the music or in the band, and present those in a way that I think is going to be most engaging.” — Paddy Free

The producer’s role is analogous to a manager in other settings. They take responsibility for the project, which can mean finding the other participants, dealing with people conflicts, making medium to high-level musical decisions, and organising the day to day actions of the activity.

Another vital role discussed by participants was that of the *composer*. Producers will act as composers to a varying extent, some are the sole composer, while others might limit themselves to just making comments on the compositional work of others. Composers in our interviewees’ activity systems all took on other roles, at least as either producers or instrumentalists.

Instrumentalists can be active in both the studio and performance aspects of music production, and for our purposes here the term ‘instrumentalist’ includes vocalists and MCs, where their voice can be seen as their instrument; as well as DJs who perform with record players. The decoupling of musical gesture from sound production (as shown in figure 2.1) has removed the necessity of the instrumentalist role in the creation of music. As it is possible to create music through notational systems rather than through direct gestural control, we might expect a reduced emphasis on the importance of instrumentalists in the music production process. However, for the most part, instrumentals played a vital role in our participants’ music production activity as described in section 5.4. Instrumentalism was a primary way in which our producers and composers became involved in live performance, even if not in the studio.

The role of *engineer* is particularly important for our investigation. There were two important types of engineers that emerged through interviews and observation. The *audio engineer’s* responsibilities include set-up and control of audio recording equipment such as microphones, acoustic design, mixing desks and effects units. The *operator engineer’s* responsibilities include the operation of the recording and editing equipment. In larger activity systems the engineering roles will often be split between two or more specialists. Audio capturing engineers were also important in a live context to provide and mix the sound for the audience. The majority of our participants took on the engineering role as a subset of their production activities.

There are a number of peripheral roles of less significance for our discussions. VJs (Video Jockeys) produce visuals to accompany live performances, and may want to have aspects of these triggered by musical events. The recording and live performance industries provide a huge number of additional roles in order to support the music production process in the context of economic, legal, and marketing support; which were out of the scope of our investigations. Finally, there is the role of audience members, both for recorded and live products.

There are many variations in the way that subjects act out these roles in actual activity systems. Our participants each demonstrated the ways in which roles often shift in their own activities, and in comparing their accounts we also see the variety of configurations across different activity systems. The following sections describe the roles in six of the many activity systems studied in our field research: those which best illustrated the many interchanging roles producers can take on, and how they divided their labour with others. Paddy Free's examples (working with both the groups Pitch Black and Salmonella Dub) illustrate how an individual can feature in multiple distinct activity systems.

Submariner

Our activity interview allowed us to discover that Andy Morton (AKA Submariner) fits reasonably closely to the role of producer. However, the additional roles that he takes on vary between each project. Often he will help music 'artists' who are working on a project to take the work to completion, as he felt that wrapping up a project could be a difficult process for them. Andy will often take up engineering roles, both in operating and audio engineering as required, as well as adding his composition and instrumental skills. Artists who Andy works with will provide both instrumental (and often compositional) material. The roles that Andy takes on start as producer, and can expand to encompass all the other major roles as required on a project by project basis, in reaction to the skill sets of the artists he is working with. Andy described how at times, his artists would want to act as operator engineer in order to manipulate the developing composition themselves. However, they did not always have the skills to make this possible.

Pitch Black

Pitch Black is a collaboration of two members, Paddy Free and his colleague Michael Hodgson. Paddy described how the typical roles they take have shifted over time, but these roles also fluctuate during their activities. Their

original roles were defined along their technological skill-sets — differentiating between the control of music and video. As the video specialist’s skills in music technology increased, their division of labour changed to the current state where Paddy Free specialises on the musical/harmonic components and the other member on sonic/textural aspects. At the same time, both members contribute to some extent to both of these.

Pitch Black works largely as an autonomous unit of two, with the members sharing the roles of producer, composer, instrumentalist and engineer. When they perform live they both work as instrumentalists. One member controls “pattern structures” while the other has control of “all the individual sounds within those patterns” and limited aspects of structure as well.

Salmonella Dub

Paddy Free from Pitch Black also acts as a producer for the dub band Salmonella Dub. Recently Paddy has begun sharing production with one of the band members who also acts as an audio and operator engineer. Production is also shared to a certain extent with all of the band members, each of whom take on a certain number of tracks to “shepherd” through the album production process.

Composition is split between Paddy and the band members. Paddy described how he composes the higher-level arrangement of the track by incorporating the best elements of the band members’ compositional ideas which they record with him in the studio. The band members are composer-instrumentalists, coming up with their own musical material in reaction to the developing arrangement that Paddy presents to them. Paddy also provides some additional instrumental performance on keyboard where necessary.

In a live context the instrumentalists take their compositions onto the stage. Additional instrumentalists are brought into the group to augment the live performance.

Rhombus

Rhombus originated with two core members. Simon Rycroft takes on more of the production role, and also composes, engineers, and is an instrumentalist. His collaborator, Thomas Voyce, focuses on composition and instrumentalism, and also acts as an operator engineer. In our activity interview, Simon explained how once a composition has taken shape, the pair bring in instrumentalists who record parts to replace stand in material that had been created by Simon or his collaborator. For example, a synthesised horn part might be replaced by the performance of a saxophone instrumentalist. In some cases the instrumentalists also contribute to the composition, either in improvising in the studio, or writing material in their own time to add to a piece.

Both of the core members perform in the live band, with the addition of other instrumentalists. Rhombus has recently begun moving to more of a traditional band format, with a larger emphasis on group composition with the whole band.

Sstimuluss

Sstimuluss is produced and engineered by Emile De La Rey. Many tracks are composed by his collaborating instrumentalist who creates the basic song structure and vocals. Emile then composes a backing track on the computer to support the vocals. Emile also produces, composes, performs instruments for, and engineers purely electronic tracks, as well as providing vocals on occasion. Other instrumentalists have been working with Sstimuluss, and also perform in the live version of the group.

ChampionSound

In ChampionSound, the instrumentalist (an MC) composes a set of lyrics and approaches Andrew Turley, the producer (a DJ), with an idea for an appropriate backing track. In the activity interview, Andrew described how

as the producer he would then develop the backing track by editing samples of existing music and adding basslines, drum grooves, and other elements by editing notation. In doing so the producer is also acting as an engineer and composer. Andrew Turley is currently looking for ways to move to being an instrumentalist as a way to more easily control electronic drum grooves, and sample triggering.

Across these six activity systems we can have described the flexible ways in which subjects take on the roles of producer, composer, instrumentalist and engineer. As Paddy Free said, “[the roles I take] really changes depending on where we are at”. Théberge [133, p.221] described this phenomenon as the ‘hyphenated musician’: the singer-songwriter-producer-engineer-musician-sound-designer. As the number of members collaborating increases, we can see that their specialisation can increase as they allow others to take on other roles. Even when members have all of the skills necessary to perform a role they may not do so in order to elicit the personal contribution of others with those skills.

5.6 Activity hierarchy analysis

Our use of the activity interview exposed the actions at the core of our participants’ music production activity. At a high level we can see how these fit into Leont’ev’s hierarchical model of activity [93] in figure 5.2. In this section we describe the generalised pattern of actions that this activity takes, as revealed by our interviews and observations. We illustrate this pattern with examples from our participants’ activity systems.

One of the more interesting characteristics of the activity is that there is no dependable or determined ordering of actions. This activity is a form of ‘situated action’ [130] in which subjects constantly adapt their behaviour in response to the state of their unfolding work. As music making is an exploratory and creative process, participants moved back and forth between various actions in a variety of ways to serve their unfolding goals:

Figure 5.2: The activity, action, operation hierarchy of computer mediated music production.

“With Pitch Black, everything is in a state of flux right up to the end. . . . If Mike makes a big build, that will change the feel of a section, so the whole piece will have to be rearranged. Everything is really up in the air, which is why it is kind of hard. . . . When we [are] writing, at any point you don’t really have a final outcome. You arrive at the final outcome without really having it as a goal. You never know where you are going, you just keep going until you end up with a track where the conscious mind doesn’t perk up.” — Paddy Free

Each participant had their own methods that resulted in variations between the ordering and exact actions conducted. Individual participants even varied the nature and ordering of these actions from project to project:

“It differs for every track.” — Simon Rycroft

This is a well known aspect of computer mediated composition, as articulated by Pennycook among others:

“... a composition emerges from seemingly arbitrary processes that the composer constructs to suit his or her immediate creative

needs. The interface between the composer’s imagination and the finished product is a tool that allows the composer to experiment with musical materials while being restricted to the requirements of the medium.” — Pennycook [116]

Another complication of computer mediated music production that we encountered, also touched on by Pennycook, is that apart from ephemeral aesthetic concerns the specific goals driving each of the participants’ actions were so transient that their articulation and systematic analysis were impossible:

“Identifying composition tasks in terms of goals seems fruitless.”
— Pennycook [116]

There were however, a number of common types of actions that our participants engaged in during the music production activity that we *can* identify. Each of these actions occurred at various times during projects depending on the changing goals of the producer. Our activity interview helped reveal details of those actions that were not seen in observation sessions, due to the practical time constraints of studying such long term activity. However, many of them were seen in our observation sessions, which lasted up to four hours, and occurred at various points during participants activity.

5.6.1 Actions to generate material

One centrally important set of actions supported the goal of generating and capturing musical material. Participants used a variety of virtual (synthesised) instruments and real instruments with microphones and inputs into their computer systems. In one session, Eric Vani and his collaborator used a variety of percussion instruments, a drum kit (with electronic pads and acoustic drums), sampled drum sounds, a variety of synthesised instruments controlled with a MIDI keyboard, a wooden flute, and a bass guitar. These various sources were then captured as the participants improvised with them

to create new MIDI and audio material. This material was then added to the developing musical representation in successive layers. Some participants, such as Andy Morton and Andy Turley, took samples from existing commercial songs and used it as material for new compositions. Christiaan spoke about the time consuming process of capturing and preparing material with the MPC:

“You are fucking round. That is a huge chunk. That is the first day. You don’t do any song writing the day you are inputting samples and trimming them, and getting ready for song writing.”

— Christiaan Ercolano

While participants often created material through their own, or their instrumentalists’ improvisation, Simon Rycroft’s collaborator would often write musical scores for instrumentalists to record later. To a lesser extent, participants sometimes also entered new musical material through editing the visual notation on screen. New material was also created by running existing material through effects processing. In an observation session Andy Morton composed material by looping four bars and slowly adding musical events until he had the foundation for a complete backing groove. He also demonstrated another groove he had created that was awaiting compositional input from a vocalist. This form of loop-based composition was common, and mirrored the way in which instrumental composers can create pieces by repeating and refining musical passages.

Compositions often began from a key musical idea that started the creative process. Andrew Turley would initially try to “...find a really good hook of a loop.” Simon Rycroft described beginning with a “beat that works” or a melody to start the process. Andy Morton and Paddy Free also usually began in this way:

“You always start off with something that gives you a clue as to what the finish might be. Like a piece of wood, I could make a boat out of this ...but I can’t make [a variety of other things].

You kinda know you are going in that direction, but it is not until you get into the final detail that you can really see where it is all heading.” — Andy Morton

“[I will] start with anything, throwing it against the wall. Find a key idea, [and determine] how will that work in the timeline. Find something you like, and something that relates to it.” — Paddy Free

Some of the participants would occasionally work as a more traditional recording engineer, where they were primarily tasked with capturing a pre-existing composition as a high-fidelity polished recording. While this is an interesting class of activity systems, for the purposes of this research we are more focused on the types of computer music production where the computer system provides a central role in the composition and performance process.

5.6.2 Actions to refine material

Participants were observed performing many actions with the goal of refining material. For example, Paddy Free was observed for several hours of editing and fine tuning material for a piece he was working on. The participants’ actions consisted of correcting timing mistakes, reordering and assembling phrases from smaller elements, applying effect processing, and fine tuning effect parameters. Refinement actions were often performed by putting the software in a loop mode so that as material was repeated, many refinement actions could be applied with constant audio feedback. Rather than looping, Eric Vani performed refinement actions in real-time as a piece played back, jumping back to previous points on the timeline to hear it again as necessary.

Another class of refinement actions was seen in mixing; where volume, equaliser, and effects processing settings were modified to ensure a good balance of sound. For several participants, mixing actions were integral throughout the entire production process. Simon Rycroft described how the desire

to mix early would result in pulling the piece into a mix-capable software system earlier than he might otherwise:

“I jump the gun It sounds great on the MPC [but] I could arrange it here or I could put it on Pro Tools and then add a whole lot of effects. You can mix it [in Pro Tools]. . . . Often what we will do is take the initial base and transfer it across to Pro Tools [from the MPC], and mix eight bars so it sounds fat.” — Simon Rycroft

5.6.3 Actions to arrange material

Another class of actions performed by participants was related to the goal of arranging material. This included actions such as repeating material, reordering medium and large periods of material, and shifting material on the timeline to make space for new material. Paddy Free described how he began the arrangement process with material that he liked, which he then combined with other material that related to it in someway. In observations, Andy Morton used Atari Notator’s arrangement view to create and then modify the form of a new piece, building it from sequence objects he had created previously. As he was synchronising parts of the arrangement in both Notator and Pro Tools, changes in one arrangement would need to be duplicated in the other system. Clinton Bradley performed arrangement actions by grouping sections of material into “track folders” and reordering and repeating them as desired.

5.6.4 Actions for live performance

One vital goal for participants was preparing for live performance. Some participants had simpler needs, and the actions they were required to fulfil for this preparation reflected that. This was most clearly seen with those who performed live against a set backing track. For example, Clinton Bradley helped the band Nine Inch Nails prepare backing tracks for live performance:

“What we did is we all sat together and figured out what the band couldn’t play live, and everything else had to be [put into sync tracks].” — Clinton Bradley

The live instrumentalists would then perform with these pre-determined sync tracks playing behind them. Andy Turley also used this method, combining static audio tracks to provide backing for a live MC vocalist. In order to obtain a degree of live spontaneity in the computer managed material, some participants exploited real-time effect processing of pre-arranged material. Emile De La Rey was observed performing live with a performer who provided vocals and guitar. The computer played back an pre-arranged piece, but also processed real-time effects for vocals and guitar. Emile modified effects processing parameters in real-time during performance for both the pre-arranged material, and live instrumentalist audio input. In all of these cases, the actions necessary to prepare for live performance mainly involved creating a cut-down version of the studio version of a piece, using normal refinement and arrangement techniques.

Other participants desired more interactive live performance possibilities, but were required to perform a more convoluted set of actions to obtain them. The main goal for these producers was developing triggering control over arrangements, rather than being limited to a pre-determined arrangement. Organising material into a form where this was possible was essentially the reverse of the arrangement process: breaking down structure rather than developing it, and creating referencable abstractions that could be recalled in a live setting. As an example, when preparing for live performance, Simon Rycroft spent a great deal of time preparing material from Pro Tools session files into samples to be reimported into the MPC sampling sequencer. Paddy Free also spoke about the large number of actions required to prepare for a performance using keyboard triggered samplers. Various sized blocks of material would need to be broken out and systematically loaded into samplers, and grouped into control sequences in his sequencing keyboard:

“That [was] complex when we were using the two samplers as the

main sound generating things, and MIDI sequencing them.... [We had] to go through a process of taking those MIDI files, converting them to audio, doing the final arrangement for the album, and then having to take those audio files, resample them back into the sampler, MIDI sequence them fresh, trying to keep exactly the same relative levels, and exactly the same groove. That was ...time consuming.” — Paddy Free

However, there had been recent development in this activity for Paddy Free and other participants using Ableton Live. This software had reduced the number of required actions by providing a single environment in which an arrangement could be decomposed into referenceable and triggerable elements called ‘clips’ and ‘scenes’. Troy Kelly and Jeramiah Ross also used Ableton Live to move between live and studio contexts.

5.6.5 **Actions for housekeeping**

A fifth type of action commonly performed by our participants would have been overlooked if we had not based our analysis in detailed observation and interviews with professional producers. We found an important class of actions supporting housekeeping and maintenance of the developing compositional object. Our participants had developed a large quantity of specialist expertise through their extensive experience in how to manage their project files. Actions such as renaming, archiving and deleting, reducing and refactoring material were all vital to the successful development of complex digital representations of the composition. Specific examples from our participants’ activity systems, and the various issues these raised are described in chapter 9.

5.7 Internalisation and externalisation

A central concept in activity theory is the division between internalisation and externalisation of human activity. Internalised aspects of activity take place in the mind of the subject as purely mental actions and operations, while externalised activity employs the use of the body and artefacts.

Externalisation

One of the most striking features of the computer music production activity is the extent to which activity and mental processes are externalised:

“The piano, manuscript paper or counterpoint are embodiments of, and stimuli for, profound thinking — and this applies equally to the computer, the synthesizer or the tape recorder.” — Keane [61, p.116]

As the production of a complex digital artefact is the central locus of activity we can directly observe the constant process of externalisation. With an externalised digital artefact such as this, the subject’s thinking unfolds through the abstract direct manipulation of represented musical concepts. In our observations, the iterative process of creative action took place through the medium of the externalised object. One result of our participants working in such an externalised activity system was that interview questions about if, when, and why subjects externalise activity did not resonate with participants. In retrospect, it became clear that when an activity is almost entirely externalised, asking about externalisation makes little sense to participants.

Internalisation

One interesting component of internalisation that participants described was what Paddy Free termed ‘colonisation’ of the mind. This colonisation process is the internalisation of their software’s user-interface system model into the mind of the subject, becoming their dominant mental model. The result

of this process was firstly that participants would conceive of and perform mental actions in terms of the entities they had been colonised by. Secondly, it seemed to be nearly impossible for participants to conceptualise new system models that would differ from their internalised mental models. When asked about what personal improvements they would like to make to their sequencer of choice, participants showed how hard it was to break out of the effects of this colonisation process:

“I’ve worked my way around [the MPC sequencer] so much I would never think of doing such a thing. . . . I’m just so used to it I can’t think.” — Christiaan Ercolano

“It is hard for me just because I’ve been using it for so long. It probably could [be improved] but I can’t think of anything.” — Andy Morton

“That is a hard question because I have adapted to these paradigms. . . . I’m probably so colonised by the paradigms of the software I have learnt with. If I was to jump into MAX lots of it would seem strange. . . . I’m sure it could [be improved] but it is a question of I’m so colonised by this other programmer’s ideas that it does everything, it works for me.” — Paddy Free

Paddy Free went on to say how learning a new piece of software would require an ‘aha’ moment where the system model would mesh with his mental model:

“Concepts of how to do things are a bit ephemeral until you have the ‘aha’ moment, and that is palpable with every bit of software I’ve owned You suddenly slot into the designers’ head space and you realise if I think like *that* everything is completely intuitive.” — Paddy Free

Simon Rycroft found that this divide between the existing colonised mental model, and new system models was a significant barrier in adopting new software:

“It’s funny actually, I’m so used to using Pro Tools the way it is. . . . I haven’t really had much interest in finding out about new software. It’s because I’m so used to Pro Tools, and I know it so well, that it is difficult to see myself going to learn a whole new system. I really should, just to see what’s out there.” — Simon Rycroft

However, over the course of the research period Simon had begun to use Ableton Live on a regular basis, so these barriers could be overcome when required.

This colonisation process could also extend into influencing how participants composed, and the types of results they created:

“Yeah it definitely does influence me. Because I tend to work in segments, every thing is cut up into two or four or eight bar segments. So I am copying things over and then never going into the details which lends itself to the whole layering thing.” — Emile De La Rey

“[It] is not the most beautiful way to write a song because you always throw things in as loops, where you’re not doing a linear path of what a song should be.” — Christiaan Ercolano

5.8 Abstraction and activity

By combining our activity theory based interview responses with those from the cognitive dimensions interview and our observations of professional producers, we were able to develop a composite picture of the interplay between participants’ activity and various abstraction mechanisms in their tools. As

depicted in figure 5.3, the aspects of abstraction for music production that we identified in our fieldwork cut across producers’ various actions identified in this chapter. In this section we introduce these aspects of music production abstraction, and detail three themes that reoccurred throughout.

Figure 5.3: We analyse Producers’ activity both in terms of their ‘actions’, and the orthogonal ‘aspects of abstraction’ that play a vital role throughout these actions. Reoccurring themes in our findings included rendering, option dilemma, and the conceptual burden inherent to the manipulation of these complex digital artefacts.

5.8.1 The four aspects of abstraction

During the course of both data gathering and qualitative coding, interview quotes and observation notes coalesced into four reoccurring aspects of abstraction that were particularly significant to our participants’ activity. The following four chapters describe each of these aspects of music production abstraction in turn, all of which are grounded in examples from our producers’

activity systems. In this section we briefly introduce how these four aspects of abstraction for music production were grounded in the observed activity, and then present a number of abstraction issues that crosscut through all four.

Processing abstraction

Firstly, audio processing was a significant component of the activity system for all of our participants. From basic equalisation, through to the synthesis and radical transformation of sound, we observed participants processing musical material in complex and multi-layered ways. The actions that participants took to group and organise both audio processing units and the resulting audio were heavily influenced by the abstraction mechanisms available to them. Conversely, the actions we observed in the field, and the personal justifications for those actions, provided a rationale for the ways in which our participants would use or ignore available abstraction mechanisms for managing audio processing and its results. Detailed discussion of processing abstraction in relation to producers' activity is the basis for chapter 6.

Voice abstraction

Secondly, an essential characteristic of our participants' music production activity was composition with multiple distinct voices. All of our participants utilised the abstraction mechanism of 'tracks' to manage and manipulate various rhythm, lead, and accompaniment voices. With our participants we discussed and observed in action many interesting issues regarding the grouping and division of these voices that was required in their work for a myriad of reasons. The specific nature of these issues are laid out in chapter 7.

Temporal abstraction

The third reoccurring aspect of music production abstraction we discovered were related to the temporal organisation of material. As composition is

fundamentally based in the structured organisation of audio in time, it was no surprise that the ways in which our producers manipulated and organised material temporally was the basis for many interesting issues. All of our participants wrestled to various extents with mechanisms for the hierarchical structuring of time, and this was central to their descriptions of linear and non-linear approaches to arrangement. We examine these findings in great detail through chapter 8.

Reuse and versioning abstraction

The fourth and final major theme that emerged from our data was the reoccurring issues we found our participants encountering with managing material used in multiple places, and the implications of creating different versions of reused material. Repetition, and theme and variation are central to composition and therefore reuse (through copy and paste, or referencing) was common to all of the participants' activity systems. Likewise, the management of distinct versions for backup and recall, at various granularities, was a reoccurring issue for our participants. Additionally, the reuse of material from a library of previously created musical resources was also important. The particulars of how these common aspects of production activity were affected by various abstraction mechanisms in existing software is the topic of chapter 9.

While these four aspects of abstraction and production activity can each be discussed individually, there were many interactions and implications between them, and this is reflected in each of these four chapters. Additionally, there are a number of themes that emerged that were common to all four, and therefore can provide a foundation for their discussion. We turn to these now.

5.8.2 Rendering considered harmful

DAW systems take multiple simultaneous audio feeds, audio synthesis output, effect processing, and parameter automation, and combine the result in real-time to create a finished audio result. As we observed in practice, a cornerstone of the production activity relies on the ability to capture intermediate results of this process as raw audio, in a process called ‘rendering’ or ‘bouncing’. This audio result can then be used in place of source material. Observations and interviews conducted for this research exposed a wide range of motivations that drive producers to render material. These were overwhelmingly based in producers’ need to create and manipulate higher-level abstractions in their digital compositions, and them being forced to approximate this by replacing various types of material with concrete audio.

Unfortunately for producers, the commonly used process of rendering has a serious down-side. Rendering is a fundamentally destructive operation that results in only primitive audio without providing access to the underlying source material which created it. The act of rendering forces the producer to **prematurely commit** (cf. cognitive dimensions) to compositional decisions and make it difficult or impossible to make changes in the underlying audio production at a later point. Our participants expressed their attitudes towards rendering and the resulting loss of **provisionality**:

“If I want it to remain malleable and changeable, I have to leave it in data land.” — Paddy Free

“I don’t think I like [rendering too early]. I like keeping things open.” — Emile De La Rey

“I typically don’t bounce while I’m going . . . I like to run as much of it in real-time as possible because if I render to audio and then change the [underlying] MIDI I have to re-render it to audio again. I’d rather keep it as live as possible.” — Clinton Bradley

“[Rendering] means you don’t have as much control in the final mix, and you have to keep going back to older sessions.” — Simon Rycroft

Ironically, many professional DAW systems make much of their ‘non-destructive editing’ capabilities:

“Take advantage of an unmatched range of unique, nondestructive editing tools for creating grooves, experimenting with arrangements, fine-tuning performances, and more.” — Pro Tools advertising copy [10]

In practice however, for a multitude of reasons detailed in due course, our participants constantly performed destructive rendering operations with material containing so called ‘non-destructive’ edits. As the resulting raw audio permanently locked in those previously non-destructive edits, the advantage of this feature was often worthless.

The following four chapters describe many cases where producers were found to be rendering material in this way. Each of these arose when systems failed to provide means of abstraction required by our participants, and they were forced to settle for the inevitable **premature commitment** this entails. While the addition of features such as audio ‘freeze’ have made some initial steps in this direction, they have proved to be a poor fit with our participants’ abstraction needs as discussed later.

Alternatively, a system that did provide the appropriate means of abstraction without the limits of freezes — that resolved this contradiction by truly meeting the needs of producers and their cognitive abilities — could make significant improvements to the creative process:

“That would be ideal, it really would: Everything being undoable, but also have the equivalent of being able to store stuff away so that [they are in] freezes or meta-freezes. . . . That’s what good hypothetical amazing software would do: Force you to make no

decisions, but allow you the flexibility to make decisions. And once you've made that decision you get all the advantages of not having to deal with it if you don't want to." — Paddy Free

5.8.3 Option dilemma

One particularly interesting issue that came up time and time again in interviews was described by Paddy Free as "Option Dilemma". Electronic music technology in general, and computer audio software in particular, has created an explosion in the possibilities for creating and sculpting sound. Now more than ever before, music producers have control over a vast number of intricate particulars of sound in addition to the traditional elements of composition. These additions range from the exact mix of generating harmonics, attack envelopes and equalisation, through to event processing and compression. We can contrast this with composing for a well defined instrument such as the piano, where composition decisions are largely limited to the standard performance variables of notes played, how hard they are played, and their timing. Option dilemma is the compositional paralysis caused by overwhelmingly open design space.

Participants voiced strong feelings that computer music systems encouraged the endless experimentation and fine tuning of the minutiae of sound design, in conflict with pushing forward and working on higher level compositional decisions and creating finished works:

"So much time is spent refining material which you may or may not use. Mostly the mixing, moving and micro nudging. . . . But perhaps only half of [this] material made it onto the album." — Simon Rycroft

When participants were asked about their reasons for rendering material to audio, they often cited the need to reduce option dilemma as a key motivating factor. Paddy Free articulated option dilemma, and how issues like this put the limitations on the users' side of the fence:

“It is reducing your options, because you are often fighting option dilemma which speaks to that — it’s the internal process now, not the capabilities of the gear, that are the main limitations.”
— Paddy Free

Further, for Paddy, the necessity of rendering forced him to make hard decisions which became an important vehicle in enabling him to move forward in the production process:

“The way of dealing with it [through rendering] is ultimately good, because being forced to make decisions is good. It forces you to progress. It forces you to evaluate on your scale of is this a good decision or not.” — Paddy Free

Jeramiah Ross also expressed the benefits of using rendering as a way of coping with option dilemma and being forced to move forward:

“The problem with having too much editing [capability] is you can spend hours tweaking one bloody high-hat sound. If it is locked down, and it is there and you are using it, you are committing things a bit more, in a creative sense. You are not pissing around with sounds for hours trying to make everything sound perfect and then messing it up. That felt right in [Ableton] Live, so I get it out of Live, and it feels right in Acid and it is done.” — Jeramiah Ross

This could even extend to the artists that producers work with:

“Everyone thinks [that digital recording] is so much easier and quicker, but for the engineer it is a longer process. There is no pressure on them to make a decision. I’m still tweaking an album we mixed [six months ago].” — Troy Kelly

This leads to our first ‘evaluation question’, designed to summarise the key findings of our research that are relevant to practitioners. The full set of evaluation questions is included in appendix C.

Evaluation question

- Are there facilities to help producers deal with option dilemma, allowing them to mark off decisions and move forward?

5.8.4 Conceptual burden

Almost all of the processing abstraction issues described above stem from sheer complexity of the system producers are working with. When working on a reasonably large project, producers can be overwhelmed by complex webs of effect processing units which also become **hidden dependencies** for the material being manipulated. Combined with the additional dependence on the correct effect automation, we can understand why rendering to pure audio becomes such an attractive proposition.

Paddy Free describes how he deals with complexity in this way:

“I bounce stuff down to audio. Committing stuff is a really good way to just wipe off complexity.” — Paddy Free

Troy Kelly described how other producers hire him to deal with this complexity for them between sessions:

“The trick is project management. I have a friend who has a studio and he hires me just to manage his projects. I go out there and he just goes ape-shit on a session. He is like ‘don’t talk to me man, just fix it up’, and I’ll spend half a day on a project. Bouncing it all down to nice tracks for when he comes back to it.” — Troy Kelly

“I still find synth programming on computers, even though it is exponentially greater than any synth you could ever have physically, it is still very finicky. Still so many layers and elements to create the sound.” — Andy Turley

5.9 Summary

In this chapter we have provided a description and analysis of computer mediated music production activity. Through the framework of activity theory, our discussion was grounded in examples from our activity interviews, semi-structured interviews, and observations with professional producers. We described the novel characteristics of this activity system, the nature of the subject and their interchanging roles, the use of a myriad of mediating tools, and the object and outcome of activity: the creation of complex digital artefacts for music production. Activity theory provided us with a flexible framework to describe and categorise the various actions of participants that we discovered in our fieldwork, and we explored the contradiction between composition and performance in live and studio contexts. We also saw that while this activity is pervasively externalised in the manipulation of a digital representation of a composition, producers are simultaneously ‘colonised’ through the internalisation of the conceptual model embodied in their tools. Finally, we introduced the four ways in which we found abstraction to be playing a vital role in our participants activity, which we explore in great detail in the following four chapters. Three issues that cut across these four aspects of abstraction were the problems of rendering, ‘option dilemma’, and the conceptual burdens of complex music representations.

Chapter 6

Process Abstraction

6.1 Introduction

Audio processing is one of the most fundamental aspects of computer mediated music production. Computer music systems that generate or manipulate audio use Digital Signal Processing (DSP), in which various digital signals are created, transformed, and combined. DAW systems provide various means to specify and manipulate this processing, as well as the results of processing, through a collection of abstractions presented by the user-interface which we call ‘processing abstractions’. The exact nature of these abstractions can have significant impact on users’ activity, as described throughout this chapter.

Modern computer audio tools have been strongly influenced by historical factors. Prior to the development of *digital* audio processing, producers created and manipulated electronic sound with analogue hardware systems. The dominant paradigm for the design of this hardware was a modular system of signal generating and manipulating analogue electrical components connected with cables. With the advent of software DAW systems, this model was reproduced through DSP in software synthesisers and effects units. These systems share the computer’s CPU resources to perform this DSP, which would have been traditionally performed by dedicated hardware.

6.2 Processing power

One of the principle difficulties with the standard processing model employed in DAWs is the limit of processing power. Only a fixed amount of audio processing can be computed in real-time by a single CPU. Unfortunately however, our participants were constantly hitting this limit imposed by their systems, exposing a fundamental contradiction in producers' activity. This problem is showing no sign of being addressed in the foreseeable future, primarily because as more powerful systems becomes available the developers of digital instruments and effects implement increasingly sophisticated and resource hungry components to achieve interesting results and more "realistic" or complex instrument synthesis. This continuing contradiction was described by Evan Brooks, founder of Digidesign and the Pro Tools DAW:

"As it turns out, what has happened is that people's needs and desires have grown to keep pace with the capabilities of computers. What people were doing with Pro Tools ten years ago you could probably do entirely in software now, but the fact is that people are doing a whole lot more than they were ten years ago, and they have a whole lot more things that they want to do, so they're still pushing the limits. . . . I don't think they're anything that people are particularly missing right now, 'cause just about everybody's needs have been met, but people keep pushing that envelope and coming out with new ideas. As the processing power grows, so does people's ingenuity." — Brooks [115]

The resulting CPU limitations were consistently an issue with our participants who found their systems unable to keep up with their processing demands. As a result they were often forced to render the results of processing, removing their requirement for CPU resources:

"There is a point where you have to stop keeping things open, whether you either commit something to audio or you bounce

tracks together for system resources ... [or else] the computer will stagger. You have to then bounce things down..." — Paddy Free

"The main thing that stops me from making changes is usually just processing power ... at a certain point you need to bounce it down. Especially if you want to make changes that include creating new instruments and effects." — Emile De La Rey

"When I use [auxiliary tracks] it is usually for very short periods of time and I'll get rid of them. I might use it to print a reverb onto a track and then get rid of them because they are using up CPU power." — Tim Prebble

"... plus my computer can't handle it. I get to a point where I am running fourteen VST plugins and trying to edit audio, and your computer is lagging like crazy, because it is trying to process everything that is running live." — Jeramiah Ross

Even with the option of freeze features, CPU limitations could still be an issue. Jim Frazier described how the real-time mixing of a large number of frozen tracks could still have a problematic impact on system performance:

"My reason for [rendering to] stereo stems, as opposed to freezing, and more importantly, as opposed to using folders, is mainly one of CPU resources. The computer has to work a lot less, when it's playing back a few stereo tracks, as opposed to all the individual audio and audio instrument tracks, packed into a folder, or even frozen. Again, this was born from an attempt to have the computer working as little as possible while recording vocals, so that the audio buffer could remain at a low enough value, so that latency from monitoring back through Logic would not be a factor or issue to the singer." — Jim Frazier

Evaluation question

- Can producers avoid manually working around CPU limitations?

6.3 Viscosity caused by processing

Another important reason that participants used rendering was to reduce the **viscosity** of the material they were working on. Effect processing automation — both on the track in question, and any auxiliary tracks to which it is being sent — creates temporal and track dependencies for any audio material placed at a given point on the time-line on a certain track. These dependencies can make moving material a complex and time consuming action, where each of the dependencies need to be properly reconciled with the new location of the material. This was a common problem, as this type of rearrangement of material was one of the most important actions for our producers.

Two location dependencies were apparent in our studies, both of which related to using a piece of material in a new context. These were when moving material on the time-line, and when moving material into a context where there were differing (or no) effects units.

6.3.1 Moving on the time-line

Material at a specific point on the time-line is dependent on two factors: i) the *configuration* of effect parameter automation at that particular time, as well as ii) the *state* of the effect processing components at that time.

i) Firstly, the audible result of effect processed material can be drastically altered by the particular settings of the effects components involved. This can vary from minute fine-tuning of the audio, through to drastic manipulation creating a result that is unrecognisable from source material. When participants moved material from one point on the time-line to another they needed to be very careful that any relevant effect parameter automation was also moved. This automation includes both that for the track on which the

material resides, and also automation on any effected auxiliary tracks to which the material is also being sent. For example, a vocal line might be on a track providing a varying chorus effect to fatten it up. This track might also be routed to a shared auxiliary track that is providing a slowly widening reverb effect. If this material is moved earlier on the time-line without the effect automation also moving, it may end up with a very different sounding chorus effect and a much thinner reverb effect.

ii) Secondly, the resulting output is also highly dependent on the state of the effect processing units at the time they begin to process a particular piece of material. We can understand this through a real world analogy. The sound we hear from a performer singing in a very reverberant concert hall is dependent on the echoes we can hear in the hall from previous notes that have been sung that are still ringing in the air. In an environment such as this a singer can begin to harmonise with some of their own previously sung notes. If a particularly sweet reverberant harmony caught the ear of a producer, they might want to use this as an element in a piece. In this case let us say the echo of the word “Doe” sounding simultaneously with the currently sung word “Me” in the interval of a major third. In the digital signal processing world the reverberant properties of the hall could be simulated in a reverb unit, and it could be used to effect a previously un-effected vocal part captured on a track. If material containing the sung word “Me” was moved further down the time-line, while the “Doe” material remains in the same location, the “Me” material will no longer sound the same, i.e. it no longer has the reverberating remnants of the word “Doe” sounding with it.

When either of these cases occur in subjects’ activity systems the end result is that blocks of material cannot be trivially moved to new points in the time-line. Because the state of the effect processing network is complex and not uniform as time passes, material is difficult to move without potentially undesirable changes to the resulting sound.

As a result of both of these factors, we observed participants dealing with this by rendering material down to pure audio.

Paddy Free emphasised the serious burden that effect automation **viscosity** can create, resulting in many of what cognitive dimensions refer to as hard mental operations:

“In a time-line based thing, if you do have automation that is an integral part of the sound, be it turning reverbs on and things like that, it does become a mind-fuck to move it around on the time-line because you might have other automation doing things at that point. Whereas if you have a reverb burst printed to audio, that is a reverb burst no matter where it goes.” — Paddy Free.

He later discussed how being forced to render audio when transferring between audio editing applications eases moving material on the time-line:

“If you have a whole lot of elaborate automation and routing and stuff to create something in [Ableton] Live, to lock that off and open it as six audio files in Logic is a real clearing of the head space. You don’t have to think about these dependencies anymore — if I move that on the time-line, you don’t have to think about ‘do I have to move automation?’ ” — Paddy Free.

Jeramiah Ross described how much easier it was to move rendered chunks of material around the time-line, rather than material that was still dependent on processing:

“Oh shit yeah, [it is so much easier] cause you can just grab the whole lot and select the bits that you want. Because you are not dealing with MIDI data or plugins or anything that is processing, because it is all committed audio.... I’m just dealing with the sounds that felt right when I made the music.” — Jeramiah Ross

Tim Prebble talked about the modal behaviour of Pro Tools, where sometimes automation moves with material, and sometimes it does not, also revealing the **hidden dependency** issues:

“In your normal view all that automation is hidden. If I go and delete this chunk here and pull all the rest across, what happens to the automation? Sometimes it does and sometimes it doesn’t [move with the material] depending on the modes you are in. Sometimes that automation is contributing as much to how well that sound works as the sound itself is.” — Tim Prebble.

The fact that this automation was often visually hidden in Pro Tools seemed to be a significant cause of the confusion that producers could have when moving automated material:

“[Automation] is extremely unreliable. I’ve drawn in automation before and moved [audio] and it doesn’t move the automation with the audio. . . . Whereas Apple Logic is brilliant because the automation is drawn on top of the audio so if you move the audio, the automation goes with it. You can actually see it when you look at the audio.” — Troy Kelly

In order to create a more direct association between automation and material, rather than to the track, participants would sometimes edit material directly:

“If I want the drums to fade out I’ll just fade them out. I make all my changes on the audio directly. . . . It feels like it is more reliable, more permanent, in terms of the project.” — Troy Kelly

On the other hand, sometimes it was more appropriate for automation to remain at a certain point in the timeline, rather than just being associated with material:

“Today I was doing a voice-over for a particular track and I had automation so that the track turned itself down, and I wanted to try different sound beds. What I wanted to do was to try different [material] with the same automation. [But] if you replace it the automation goes away.” — Simon Rycroft

6.3.2 Moving to non-processed context

A more extreme example of moving material to a new context took place when participants wanted to move something from one track to another. Material that is highly dependent on its effect processing cannot be moved to a different track that does not have the same configuration of effects processing units without losing its sonic character. However, as we will discuss in chapter 7 there are important reasons to organise material on tracks other than just for the effects that are processing those tracks. When participants needed to move material onto a different track for organisational reasons, they were observed rendering down to audio.

Paddy Free explicitly described how much easier it was to manipulate rendered audio for moving between tracks:

“[Rendering to audio] is the easiest way because we ultimately are dealing in audio. To have it as an audio file, with no plugins on it, it can just be moved up and down on any track anywhere on the time-line. That is the unit of currency.” — Paddy Free

In an observed composition session, Paddy rendered small chunks of a vocoder effected guitar part. These were then pulled back into the main project onto guitar tracks with no vocoder effect unit. As the effect results were baked-in to the rendered material they could easily share a track with other material without the same effects.

Evaluation questions

- Can producers easily change materials’ voice and location in time without losing processing context and how it will sound?
- Can automation parameters be associated with material or points in time as required?

6.4 One-off effects

A common area where participants' actions were in conflict with the provided abstraction mechanisms was dealing with one-off effect processing, where a relatively short chunk of material needed an effect applied. There were two situations where one-off effect processing would result in participants rendering effects processing to audio, one where an effect was applied on a single track, and the other was where shared effect processing temporarily diverged.

6.4.1 Effect on a single track

When an effect needed to be applied to only a portion of a single track, it could be problematic that participants were required to associate an effect unit with the entire track, as only a limited duration of processing was actually required. An effect associated with the track would exist for the entire duration of the track, conceptually complicating things and potentially consuming valuable CPU resources.

Paddy free explained why he rendered one-off effects to audio, rather than leaving an effect unit attached to the track:

“I might either do the exceptional send and then print it to audio or I might insert one into that channel for that reason for that effect Even if I just wanted this exceptional event to happen, I would probably commit it back to audio so I could just think of it as an event, and not think about it as ‘oh that has to stay there to open to that in time’.” — Paddy Free

Even when the effect was a simple volume change to a small section of material, Paddy was observed doing in-line edits and committing to audio rather than using the track's built in volume automation. This also allowed him to move the material to a non-processed context as discussed in section 6.3.2.

Inline edits were impossible where the processing resulted in a ‘tail’. For example, a large reverb effect can result in an echo sounding much longer than the source material:

“Sometimes I use audio-suite style plugins where you are doing [edits] on the material itself. [But] for delays and reverbs and effects that have a tail I would use auxiliaries . . . [because otherwise the tail would overwrite what follows].” — Simon Rycroft

Clinton Bradley usually dealt with one-off effects by “chopping the tracks [into] one that has effects and one that doesn’t. [Then] just chop those single words and stick them on [the effected track]”. He was able to manage the vast number of tracks this required by folding them into Apple Logic’s folder track abstraction mechanism:

“I might have five or six tracks that make up the single vocal [inside a folder track]. At that point it is like dealing with one track.” — Clinton Bradley

For more complex processing scenarios he would fall back to creating automation curves to manage the effects parameters. For other participants, without an abstraction mechanism such as folder tracks, things could become complicated:

“I get lazy because I move things from track to track instead of keeping a dedicated track, but I find I end up with a badly organised session. I do it all the time. It is a trade off. The other way is to automate all your effects but it is often easier to just use discrete tracks. But the disadvantage is building up your track count.” — Simon Rycroft

Simon went on to say that this approach would become particularly problematic when he returned to a project many months later, as the relationships between the many tracks would be unclear.

6.4.2 Temporary diversion of shared effects

Participants also rendered one-off audio processing where an effect unit was being shared by several tracks but needed to temporarily diverge. As an illustrative example, consider several tracks running through a shared reverb, with the reverb's parameter automation effecting each of the tracks identically. If one of these tracks needed a differing reverb “pan” setting for five seconds, the track will need its own separate reverb unit.

To avoid having to create a second but otherwise identical reverb unit, the participants would often render the one-off effect so it could stand alone, or alternatively render the shared reverb output and use the reverb unit on the deviation:

“If I want to print that reverb or edit that reverb because it is great for nine sounds and crap for one, I’m either going to have to put in another reverb or I’m going to have to commit to those nine, print them as audio and change it.” — Paddy Free

Evaluation questions

- Can short term or diverging effect processing be created without adding complexity or CPU overhead?
- Do the object editing facilities allow for tail effects?

6.5 Editing effected results

Another reason that participants rendered effected material was to enable easy editing actions. This sort of editing includes breaking material into smaller blocks, merging and deleting blocks as well as moving and reordering blocks of material (as discussed in section 6.3.1). In the activity systems of electronic music in particular, the results of effects processing have a very important role. As participants reported, and we observed in action, results

of processed audio is often used as first-class compositional material, just as traditional recording artists might use a recorded guitar part.

Jeramiah Ross described why the kind of editing that rendering makes possible is important:

“I bounce down [the effects] as separate tracks. That is cool because you can choose when to end a delay unit. You can actually chop up the effects, so the effects themselves becomes a sound that you can manipulate. So you can reverb the reverb, or reverse the reverb but still have the main sound going.” — Jeramiah Ross

Simon from Rhombus echoed this sentiment:

“Especially with a lot of electronic music the effects, particularly filters and delays, would be a part of the composition.” — Simon Rycroft

In an observation session, Eric Vani rendered heavily processed audio tracks and proceeded to chop them up and move blocks into different sections of a piece under construction. His rationale for this was that it was necessary to allow him to “push in one direction” even though, as he acknowledged, they might lose some possibilities.

Paddy Free’s collaborator in the electronic dub group Pitch Black works primarily in this way — building compositional material from the result of heavily processed audio:

“So much of our stuff is actually made out of the mixing, out of the dub. . . . What was usually the last thing in the process — the final mixing — now became another object to compose with on the time line, and that was a pretty special thing.” — Paddy Free

For the same reasons, Emile described freezing features as “deficient” because he too needed this ability to “edit the bounce”.

These examples underscore a fundamental lack of effective abstraction mechanisms that allow producers to treat the results of audio processing as first class material. Rendering processed material provides a vital crutch to approximate such abstractions, while failing to retain and manage vital links to the underlying material it was created from.

“If I have complex automation I need to be able to freeze [them] but still be able to move them around.” — Emile De La Rey

Also, until material has been rendered to audio it is generally not possible to see the waveform visualisation of the audio that is actually being heard. A recorded guitar part on a track with a real-time digital delay effect will still be graphically depicted as a waveform matching the original recording. While the digital delay effect will have introduced important additions to the sound which are perceived as new perceptual objects to the listener, these additions will not appear in the waveform view, breaking the **consistency** of the visual presentation. If the producer wants to identify, select, and edit these perceptual objects visually, they will need to first render the material to audio.

Evaluation question

- Can results of effected processing be edited and viewed as first class audio without regard for underlying effects processing, while allowing underlying processing to be viewed and modified even after the results have been edited?

6.6 Rendering for live performance

Another situation in which the provided processing abstraction mechanisms can be lacking is when they fail to account for the live performance aspect of

many producers' activity. In a performance context, tracks presented with all effects units, and all of their automation graphs and parameters, can prove to be overwhelming. In order to perform live, participants sometimes simplified the representation of their compositions by rendering many live processed effects down to audio. This process made up one part of what Paddy Free referred to as "setting the conditions for spontaneity for live performance". Simon Rycroft and Clinton Bradley also followed this method:

"[For live performance] you can bus [effects], but it fills up too much space visually with all the busses and everything". — Simon Rycroft

"[For live performance it was] all taken from the master tapes, sequences, mix downs, cassettes." — Clinton Bradley

Evaluation question

- Can processing complexity be reduced for use in a live setting?

6.7 Hiding edits

One recurring problem for the producers in this study was keeping a listener's perspective. The chief obstacle in this regard was the information provided through the visual display of their composition. Because producers can see visual representations of edits during playback, it was often very difficult for them to get any real sense of how the material would be perceived by users who did not know that the edits were there. As edits may vary in their success at providing a seamless audio stream, it is important for producers to be able to hide them to allow them to more accurately judge their effectiveness from a listener's point of view.

Tim Prebble described how the visual representation of edits changed his perception of the sonic result:

“If I can see an edit coming up it informs me, so you are not listening to a sound-track, you are listening to a detail of it.” — Tim Prebble

Additionally, being able to look ahead on the time-line to upcoming musical events as well as edits can change the producer’s perception of compositional development. The majority of the participants worked at least partially with the monitor turned off. The reason for working in this way was most clearly articulated by Paddy Free:

“Often, so often, I’ll make a point of turning off the screen to listen back to a mix because it will throw you. You don’t listen to it in the same way. . . [for example] this is a bit boring but I can see a big change is about to happen, and that’s bad. No one else will ever have that ability. You will be more tolerant of things that don’t work if you see that a change is looming.” — Paddy Free

While the graphical representations can be a problem in providing *too much* information, they can also cause problems by showing data that is actually misleading. For example, the psychological perception of loudness is only informed to a limited extent by its measurable energy level (represented as decibels) [96, p.67]. Television commercials exploit this psychological artefact by being perceptually louder while meeting the broadcasters’ decibel limits.

Professional producer Troy Kelly was also well aware of the pitfalls of depending solely on graphical representations:

“You really have to detach yourself from the visuals. Just because the meter is loud doesn’t mean it feels louder.” — Troy Kelly

Evaluation questions

- Is it possible to hide edits?

- Can looking ahead on the timeline be disabled?

6.8 Real-time processing

Because effects processing is so central to developing compositional material, several participants often created effect feedback systems inside their DAW. These feedback systems are built by creating cycles in an effect processing network, where the output of an effects unit is fed back into itself or one of its parents. The quantity of feedback into the system is controlled by various effect unit parameters. These systems were performed in real-time, captured as parameter automation, and the audio result was typically rendered for editing.

Participants also created complex effects processing networks with or without feedback, which were central to their compositional process. For example, Emile developed a piece by taking a microphone input, splitting it into three separate signals divided by differing frequency ranges. These were each delayed individually to space them out temporally creating a rhythm seed to send to a re-synthesiser that created a new sound from the processed material. These sorts of networks resulted in complex channel structure and interdependencies which could become difficult to manage.

An interesting consequence of the real-time processing model employed by multi-track systems is non-deterministic playback. For example, this manifested during an observation session where effects processing of a section of material varied depending on the point at which the playback head began. Analogous to the problems of moving material on a time-line (section 6.3.1), changing the starting point on the time-line means that material is processed without the same effect unit state that would be present had the piece been played from the very beginning. Troy Kelly regularly encountered this issue in his production work, and used freeze features to remove seemingly non-deterministic playback:

“Once you freeze it you get used to the sound of it frozen because

it has been applied, for instance how fast a compressor is, the attack time at the beginning of a phrase will be different every time you play it back. For instance, this one song has a stop, and the bass and guitar stop ... and every time it would be different depending on how the delays work. It is just locking it down [to a frozen or rendered state that corrects this problem].” — Troy Kelly

A related issue was observed when a Paddy Free copied and pasted a guitar motif to repeat one bar later. When the motif was repeated the effect unit was not in the same state as the first copy and a different result was heard. Because the participant’s intention was to have the same result sound twice in a row, he was forced to render the correctly processed version, and to create a new track with no effects, which he could copy and paste it to.

Evaluation questions

- Is it possible to create feedback loops and manipulate them interactively?
- Is it possible to have predictable results of effect processing?

6.9 Summary

There were a number of ways in which existing mechanisms for abstracting audio *processing* were deficient or non-existent. This was most prominently visible where producers needed to *render* material (losing important information as a result) rather than using ‘freeze’ functions or folder track mechanisms. The causes of this were a combination of performance limitations not being resolved; needing visualisation and unhindered editing of the end results of processing; reducing unneeded complexity for live performance; and requiring unchanging results regardless of where material was moved or the initial state of effects units when playback was triggered. Additionally, the

current abstraction mechanisms from the multitrack-mixing model of insert effects and send channels were not adequately suited to producers use of both short term as well as shared, but temporarily deviating, effects. Participants also expressed dissatisfaction with the fact that it was not possible to easily abstract away edits in a non-destructive fashion, and without losing both waveform visualisation and further editing capabilities. Our findings uncovered how this could be problematic, as participants reported that viewing edits on a track during playback could bias their perception of the quality of those edits, creating a further rational for the abstraction of edits.

Chapter 7

Voice Abstraction

7.1 Introduction

Parts are central component in the composition and performance of music. Parts are the individual lines of material played synchronously by a number of independent ‘voices’. Voices are entities that have their own persistent existence for the duration of a piece, and are not just associated with vocalists. While DAW systems typically start by representing voices with ‘tracks’, there are differences in how they represent and allow manipulation of these abstractions. How well they fit the needs of producers’ activity can have dramatic consequences for the way in which producers work. We call ‘voice abstraction’ the means by which voices and parts are modelled and represented to the user, and its interaction with the music production activity is the subject of this chapter.

In pre-electronic music, performance was typically organised around a relatively static number of instrumentalists (voices), each of whom have a part to play during the piece. Each of these parts have their own identity over time. For example, a short phrase to be played by a flute voice must be assigned to one of possibly several flute parts, and thus to an individual (or a group of) performers. The maximum number of distinct simultaneous flute phrases performed at once dictates the number of flute parts, which clearly

maps onto performers. Each part traditionally has a name, such as ‘First Violin’, ‘Alto’, or ‘Second Bassoon’, to name but a few.

Typical bands also have a relatively static number of members. Band members often play for a significant portion of each a piece, and part of the reason for this is social — each performer needs to have something to do for a good proportion of the time:

“There are five [band members], and it has to be coherent for everyone to have a good go at [performing].” — Paddy Free

While this abstraction of voices could be taken for granted due to its necessity in traditional instrumentalist music, electronic music removes the *requirement* of the instrumentalist, and therefore also a core need for the current abstractions for voices and parts as well. Digital representations of electronic music can describe each musical event with its individual synthesis parameters without regard to any other. Such a representation would contain no abstraction of voices, even if the composer or listeners could hear what they perceived as continuous lines of melody and instrumentation. This perception (rather than compositional organisation) of voice abstractions is ‘stream perception’.

7.1.1 Stream perception

The second use of voice abstraction stems directly from the human perceptual system. The faculty of composers, and indeed all listeners, to perceive coherent voices in music is part of the gestalt process of Auditory Scene Analysis described in detail by Bregman [27] and introduced in section 3.2.1. Bregman used the term “horizontal integration” to describe the particular process at work where a temporal sequence of sounds seem to cohere into what musicians would call a part. Horizontal integration is activated by sequential audio events that minimise differences of pitch, temporal delay, and over-tonal / textural properties.

The streams we perceive are a result of the workings of our audio perceptual system, which has evolved to identify and classify sounds and their sources in the natural world. However, as Bregman explained, our audio perceptual systems can be fooled. A perceived stream may in fact come from multiple physical sources. Alternatively, a single source performing at high speed can create the illusion of multiple streams by radically varying timbre or pitch as described earlier. Musical compositions exploit this phenomenon when a single skilled instrumentalist rapidly intermingles notes from melody and bass parts to create the illusion of accompaniment. The same is true with electronic music production where horizontal integration may or may not map to the process abstractions that created the sound in the first place.

In his role as a film audio producer, Tim Prebble described how after building a composite sound out of its component parts, he would want to deal with an abstraction matching the holistic perception of the sound:

“[The sound] has become an amorphous kind of whole rather than elements. As part of that process of premixing, I’ll take the 100 elements for a car crash sound, they’ll be balanced up and mixed into a couple of blocks of sound, and from that point of view the elements don’t matter anymore, it is just a unit in itself.” — Tim Prebble

This could be used to create what is perceived to be a single instrument from multiple sound sources:

“I’ve done keyboards on an album and I used hammond, clav and a harpsichord to create one sound. You bounce it to one group and it sounds like one instrument. I ran them to a subgroup and bounced them out as a stereo track.” — Troy Kelly

Paddy Free’s example from section 6.3.2 also demonstrated how he reorganised material onto a single track, creating a unified voice to match how he

perceived it, rather than broken into tracks resulting from the effect processing that created them. Here then, we have exposed a tension between two contradictory organisational schemes inherent in producers' activity. As a result, the organisation of material into one scheme can damage the **closeness of mapping** to the other.

Evaluation question

- Does the system provide voice abstractions to capture both the concepts of instrument parts and perceived parts?

7.2 Voice duration

In multitrack systems, musical material is placed onto any number of tracks, each of which lasts the full duration of the piece. Tracks can be used to capture the abstraction of parts or used to map voices perceived as horizontally integrated streams. In the multitrack model, each track lasts from the beginning until the end of the piece, even if it contains only a few measures of audio. One resulting issue for study participants was an explosion in the number of tracks required, resulting in a highly **diffuse** information display. For example, the addition of a vocal harmony for just one measure of a song often requires a whole new track to be added for the duration of the piece, most of which is empty:

Interviewer: But you don't find that you don't have to juggle heaps of tracks with one little bit happening on each of them?

"Yeah you do, but that is just life. I'm doing an EP at the moment and we are up to fifty vocal tracks at the moment [with just three vocalists]. It is like 'do you put this little part on here?' If there are some 'Ohhs' in a song and they haven't happened before do you just put them on another backing track?" — Troy Kelly

But Troy points out how this idea of putting transient musical material onto an existing backing track is not without its own problems:

“But then later [the client] will be like ‘can we have more reverb on those ‘Ohhs’?’ and it will turn the reverb up on **everything** on that track. So it is much easier to just create a whole new track and make sure it is all cool and lock it down [on its own track] for good.” — Troy Kelly

Evaluation question

- Does the system provide an abstraction for dealing with transient musical events?

7.3 Organising voices

Participants used the voice abstraction provided by tracks to organise material, making the developing digital artefact easier to search through. There were two strategies used. The first was that producers often placed tracks next to each other if they felt they were closely related. Groups of related tracks are formed by using their visual layout on the screen:

“In the writing phase you are dragging tracks left and right, and grouping them [by putting them next to each other], and that is helping you organise your ideas and things.” — Paddy Free

“I group them in automation groups with them sitting next to each other.” — Simon Rycroft

“I do group like things together, the distorted guitars are together the acoustic guitars are together. Maybe not initially while I’m frantically writing, but I will at some point organise.” — Clinton Bradley

The second organisational strategy that some (but not all) participants used was to form a habit of ordering tracks consistently. Depending to a certain extent on the style of music being produced, the same track ordering can be used in different pieces, making it easier to locate particular tracks.

“I have a specific order for my instruments so I am not constantly going up and down [searching for tracks] in every piece of software.” — Troy Kelly

“Usually I start with the drums, percussion, then then bass, keyboards, [miscellaneous tracks], then vocals will usually come last.” —Simon Rycroft

However, some producers preferred a more organic approach to track ordering:

“There is an element of haphazardness to it. I don’t always put the vocals on track one. I generally start with a blank slate.” — Clinton Bradley

Neither of these organisational strategies have an effect on audio output, and so act purely as a means of managing complexity.

Evaluation question

- Does the system allow producers to make use of visual spatial layout to organise voices?

7.4 Amalgamating voices

One simple problem with large track counts is that they consume large quantities of screen real estate. While producers can rapidly move over large multichannel mixing desks, they encounter difficulties when this same metaphor is used in the software world:

“In [Apple] Logic for example, when you’re on the final mix and you’ve got a mixer page that is fifty channels wide, that is much harder to get into than seeing it on a desk which is the paradigm that it is mimicking.” — Paddy Free

Multitrack software systems use of this model breaks down when it requires scrolling and zooming. Scrolling across a large number of tracks guarantees that many tracks will be off screen and difficult to access. At the same time, shrinking each track by zooming out or using focus+context views [36] makes it difficult or impossible to quickly see and compare wave form and automation curve data. Paddy Free described how this could be a problem in his work:

“If you are fifty tracks deep on [Apple] Logic’s arrange page, and if you really want to see all those things, you are going to have to make it really small and narrow and piddly little things so that you can no longer see the labels.” — Paddy Free

Tim Prebble was frustrated by the same issue, and suggested an abstraction mechanism he would like to see in Pro Tools:

“You only have so much screen size. If I had a session open that had a hundred and twenty tracks, if I group them I still can’t edit all those hundred and twenty tracks at once. I want to be able to see them [all at once]. I should be able to collapse those one hundred and twenty tracks into six tracks, where each track might have sixteen tracks underneath it, and if I select a chunk and copy it, it looks like I’m copying one track, but it is copying all the tracks underneath it.” — Tim Prebble

This corresponds to the problems Nardi found in a more traditional end-user programming context:

“Users had a strong preference for being able to view and access as much data as possible *without scrolling*.” — Nardi [105]
(Emphasis added)

Often participants went beyond grouping voices spatially on the screen, and actually amalgamated groups of voices in this way into a single voice. This allowed the producers to view and manipulate a multitude of distinct voices as a single entity. This process allowed participants to address one of the principle limitations in the multitrack-mixing model:

“The sequencer can draw the user’s attention to selected aspects of the music-making process that are microscopic in their level of detail, but it does so in a fragmented and disjointed way. In a curious fashion, then, the temporal and spatial separation between performers in the multitrack studio is reproduced in the separation of information on individual tracks on the sequencer.”
— Théberge [133]

There were two ways in which participants merged several voices into one. In the first case, multiple takes could be “comped” down into one, taking the best from each part or just one of the takes as is. As each take constitutes a different version of material, reuse and versioning abstraction plays a role here, and so we discuss this further in section 9.4.2. In the second case, which we examine in detail now, participants amalgamated multiple voices together using summing mixing, where the end result is a mix-down of several voices playing together while being represented as a single voice to the producer.

This amalgamation predominantly took the form of rendering several tracks into a single track, but also included grouping multiple tracks into a single unit using folder track tools. One rationale for applying this form of abstraction is to apply effects processing to a group of voices, as was described in chapter 6. Producers’ work produced many additional goals for amalgamating voice actions as we describe now.

For producers who perform live electronic compositions directly from a multitrack system, it can be useful to minimise the number of tracks they are presented with. Reducing track counts both simplifies what the performer is dealing with, making it easier to locate material, and also gives them meaningful handles on higher level constructs for manipulation. For example, if all of the drum parts are amalgamated, they can then be muted and modified as a whole. Our pilot participant explained that in a live context there was no time to work with a large number of voices, and that it was necessary to minimise the number of voices that a DJ performer was required to work with. Paddy Free and Simon Rycroft had similar perspectives:

“Making a scene is the ability to trigger a whole bunch of multi-tracks as one. So that is an abstraction, that is an overlayer.” — Paddy Free

“I do [render down tracks] for live [performance] because it makes it easier to trigger, because you don’t need to trigger eight channels. You can just trigger the drums as one track.” — Simon Rycroft

Clinton Bradley did the same voice amalgamation for his work with Nine Inch Nails and other groups who performed live:

“...it was grouped into seven separate tracks and I’d subcategorise everything for a given song. I’ve also worked with bands where the different parts are saved in folders so can be sequenced live.” — Clinton Bradley

As we have already seen in section 5.6.4, many aspects of live performance are also present in studio activity. Complexity is also a major issue in the studio, and we found that large numbers of tracks are key contributor to this. As with the live context, increasing numbers of tracks increases complexity, and so finding specific material becomes more difficult:

“The more complicated things are, the more tracks there are, the more ways things can get lost.” — Andy Morton

The problem of excessive numbers of tracks can be problematic right through to the final stages of production:

“If you went into a final mix with all these source tracks there is no way you could do the specific things to individual tracks that you need to. You would run out of time and energy.” — Tim Prebble

To address this, producers attempt to group and amalgamate tracks to achieve a form of information hiding and encapsulation:

“Managing, conforming, and updating all those tracks can be a nightmare because you are dealing with sessions that are hundreds of tracks wide, and being able to simplify it down to a representation where you just have four blocks [is ideal].” — Tim Prebble

In situations where participants were forced to render tracks together, this could have the benefit of reducing the complexity of dealing with large track counts:

“[An advantage is that] you are forced to make mix decisions earlier on, rather than working with a huge number of tracks.” — Simon Rycroft

‘Refactoring’ is a process from software engineering [66] in which developers revisit complex program code, editing it to clean up problems and create new abstractions. We discovered an analogous process in music production activity. Overwhelming track counts is such a problem for some producers that they bring in other people to refactor their projects, as we quoted previously:

“The trick is project management. I have a friend who has a studio and he hires me just to manage his projects. I go out there and he just goes ape-shit on a session. He is like ‘don’t talk to me man, just fix it up’, and I’ll spend half a day on a project bouncing it all down to nice tracks. He never wants more than sixteen tracks when he gets it back. Sometimes you’ll go through all the backing vocals and edit them all up and group like six tracks of backing vocals together because it is one person as one long track, and bounce that.” — Troy Kelly

Here Troy’s use of bouncing (rendering) mirrors what we saw with processing abstraction. Abstracting away the complexity of large numbers of tracks is so important that producers are willing to forgo the flexibility that keeping them as independent tracks would allow.

One way in which some software systems attempt to address this is through “folder track” mechanisms. Producers can place multiple tracks inside a folder track, and in many cases place folder tracks inside other folder tracks. Troy describes how beneficial this can be when using Cubase:

“One good thing about Cubase is the folder tracks because [all the tracks] are still there but they’re not there, and then when you pop over to the mixing desk they’ve disappeared so you’re not scrolling through all these tracks, but they are still playing.”
— Troy Kelly

Clinton Bradley described how nested folder tracks allowed him to manage hundreds of voices without being presented with them all simultaneously. Guitar tracks, vocal tracks, drum tracks and the like were all bundled into their own folder tracks, so there were just “five or six folders that I’m dealing with instead of hundreds. . . . That dirty harry track I think that was a hundred and eighty tracks”.

Participants also rendered multiple tracks together when they wanted to work on other things. For example, during an observation Paddy Free wanted

to record material to an effects-laden guitar part. This guitar part was then to be heavily edited to create new phrases. In order to hide the complexity of the many tracks already in the piece, he rendered them all down to a stereo pair and imported them into a fresh project. In this “cleaner” environment he was free to concentrate on just new material with minimal distraction. Once the new material had been carefully crafted it could be taken back into the original project containing the full number of tracks. Also, similarly to the perceptual consequences of the **visibility** of edits (as described in section 6.7), not seeing so many tracks allowed the producer to hear the piece without being unduly influenced by what was on the screen. Paddy Free described this as giving him “fresh ears”. Untainted by edits and track organisation, he was able to rely on his natural stream perception to gain a truer picture of the result. Rendering for this reason can have the added benefit of saving processing power, as we discussed in chapter 6.

Jim Frazier also removed extraneous tracks when focusing on a particular set of voices:

“Visually, I focus better when the majority of what I am seeing is the vocal tracks, and not twelve tracks of drums, five tracks of guitars, etc. The psychological focus is on the vocals, which personally helps me stay extremely focused on the task at hand.”

— Jim Frazier

For the same reasons, other participants used track hiding features to temporarily visually remove tracks from their projects:

“That is how I work: I’ll hide everything except the vocals and then I’ll mix that to zero, then I’ll turn it off, and I’ll go through every instrument and get them sounding how I want them. I hide all the other tracks so I don’t get overwhelmed.” — Troy Kelly

“[I hide tracks], because it makes the session less complicated to deal with visually. For example, if I am going back to an old

session and I just want to work on the vocals I'll make everything else inactive..." — Simon Rycroft

However, hiding tracks rather than grouping and amalgamating could be less than ideal:

"[Hiding channels is] slightly clumsy in the sense of seeing and finding things at a larger scale. You are forced to use groupings of tracks and hide and show groups. But it would be much better if you could actually collapse them down into blocks. If [a project] was built up of four groups of sixteen tracks I should be able to collapse it down so that all I see is four blocks of sound. It could be much more hierarchical." — Tim Prebble

The majority of participants described how they wanted to edit multiple tracks as if they were a single entity. For example, while a producer will sometimes need to edit a drum kit's various parts independently (high-hat, snare, kick etc.), it is also common for them to want to edit an amalgamated drum groove. An instance of this occurred during an observation session with Eric Vani. A piece contained several tracks, including five harmonising vocal parts with effects. The producer took these five tracks and rendered them into a single new track, and proceeded to edit it as if it was a single voice. Acknowledging the fact that rendering can eliminate some possibilities, he explained this as a necessary to push in one direction. Rendering multiple voices into a single track allowed him to create an abstraction so he could "treat them as one thing".

Other participants gave examples of when they would want to edit a number of voices as a single entity:

"Yeah, I'll always group the drum mics together, so it is cutting all of the drums, and sometimes the bass as well. So if I am changing takes I am changing the bass with the drums." — Troy Kelly

“[For example] a string quartet with four microphones [is a single group], but it takes up four tracks. In most sessions there is something that could be treated as one [entity], and often it ends up being bounced. With a single guitar, you have the [direct input] and the sound of the amp, and the [sound of the reverberation in the room], all for one guitar.” —Simon Rycroft

While mechanisms such as Apple Logic’s folder tracks allow many tracks to be chopped up and moved as one, they do not allow the amalgamated whole to be treated as a single waveform. Folder tracks require special treatment as they cannot be directly associated with their own effect processing, and do not allow operations such as normalisation, gain editing, reverse and invert, searching for peaks or silence, and time and pitch modification.

Just as voice abstraction can enable editing multiple voices as one, it can also be vital for manipulating larger arrangements. Some producers merged voices onto shared tracks temporarily to help with arrangement decision making. The simplicity of fewer tracks makes it easier to manipulate material, try different arrangements, and decide in which direction to take a piece.

“I try to simplify it. Figure out a way to visualise it simply so I can make the decision. I might sub-group it all down to just a two channel mix. Say if I had a five minute song it probably would be easier to take it down to two tracks and then do an edit, splicing and moving parts around. Then when I had made the decision that that was how I wanted to do it, I would go back to the original and do a nice job.” — Andy Morton

In contrast, if each voice is left on its own dedicated track, quick rearrangement operations become difficult. In this case, moving a chorus, verse or phrase requires selecting the correct duration of material on *all* tracks and carefully splicing them into the correct location somewhere else in the piece. The more tracks there are, the more **error prone** the operation becomes. This results in an increased risk of one or more tracks being missed or fragmented.

With the voice abstraction that rendering to a stereo pair provides, material becomes less viscous, and producers are able to rapidly prototype and test their ideas. This iterative and rapid testing workflow is a key component of the creative process [126].

Producers use of many techniques to amalgamate tracks can become problematic when appropriate visualisations are not provided. If a number of tracks are to be grouped, and the component tracks hidden, it will often be useful for producers to be able to see a visualisation of the amalgamated audio. In Wayne Laird’s work editing symphonic recordings, he would perform edits across multiple tracks while only being shown two of the tracks:

“The window that you see here is stereo because all these channels have been turned off visually, but only visually, because they are still here on the desk and they still sound. But it means graphically you are only working with two channels.” — Wayne Laird

In Wayne’s case, depicting a simple stereo pair was sufficient due to their similarity. In a symphonic recording scenario such as his, all of the tracks will be structurally similar as each track is simply sourced from various microphones located around the same performance venue. However, in multitrack scenarios, where each part is isolated from each other, tracks can be structurally distinct. In these situations where tracks diverge, such a solution proves problematic. Producers need to see visualisations of audio events in order to edit them efficiently. This is another reason that participants gained an advantage from rendering multiple tracks into a single track, as they were then presented with a waveform visualisation of the amalgamated parts.

While Apple Logic’s implementation of folder tracks allows multiple voices to be manipulated as one, it presents no waveform visualisation of the amalgamated result. Ironically, Logic’s own manual illustrates why viewing this detailed waveform information can be so important:

“If accuracy is what you need, you should edit the start and

end points of Regions in the Sample Edit window, not the Audio window. The same goes for any adjustments you make to the Anchor, which in many cases should really be placed on the amplitude peaks, rather than at the start of the attack phase of the sound.” — [6][p.284]

Clinton Bradley had managed to work around this limitation by constantly working in parallel worlds, with multiple simultaneous views. By jumping between a number of Apple Logic’s ‘screen sets’ (each consisting of a particular configuration of window views) he was able to view a display of high level folder tracks, the track contents of the currently selected track folder, and the audio-waveform editor for the currently selected audio region:

“I don’t ever need a composite because I am always looking at another window that has the contents of that folder. That is the beauty of screen-sets, you can have the entire arrange on one screen and the contents of each folder on the [other screen]. I will ideally use two 24-inch [screens] at least.... When I see waveforms is when I zoom into the individual tracks themselves. It is critical to have good waveform display.” — Clinton Bradley

However this workaround forces the producer to be exposed to the entire contents of the voice abstraction whenever they want to see a visualisation of what is going on, or perform detailed editing. This could cause problems when producers become confused by all of the sub-tracks in track folders:

“[Folder tracks] can bite you in the ass sometimes too because you think ‘where is that coming from?’ You’ve muted everything and they are still coming, and you are like ‘arrrg’. Because you can’t see it, you forget you put them in a folder track.” — Troy Kelly

Simon Rycroft liked the idea of built in support for voice abstraction, but emphasised how important it was for him to have a detailed visualisation of the amalgamated waveform:

“If you have say eight tracks of drums and you can shrink the group down to one track, it would be OK to edit it if you could see the resultant waveform of that group of tracks. I think it is important [to be able to see the waveform] if you want to be able to edit the tracks while they are combined. Because otherwise you couldn’t see where the transients are happening so you wouldn’t be able to make informed editing decisions.” — Simon Rycroft

Evaluation questions

- Does the system provide voice abstraction facilities to reduce voice counts and the resulting overwhelming complexity?
- Does the system allow producers to view, edit and arrange multiple voices as if they are a cohesive whole?

7.5 Separating grouped material

While we have now seen the importance in producers’ activities of abstraction though the amalgamation of voices, producers can also need to delve into the sub-voices for editing. Systems that only offer rendering as a means of approximating voice abstraction can be extremely limiting in this regard due to the lack of **provisionality**. The technique of aggregating through rendering cannot be easily reversed. When relying on rendering, producers need to be always mindful of the commitments they are making through the rendering operation, and weigh up the benefits of the abstraction with the consequence of reducing their options for using or editing sub-voices at a later in the production process.

Christiaan Ercolano described how this could be a problem on the MPC when creating tracks built from layers of voices. For his work, it was problematic if he could not separate out each individual layer:

“Again you are down to tracks, and you hope that your elements are on [individual] tracks so [that for] any given sequence of any part you are just muting or soloing.” — Christiaan Ercolano

In this case, the producer referred to this voice abstraction as a “stack”, and an important part of his compositional process was breaking this stack down to into its individual layers:

“What I tend to do is stack tracks up in a sequence, and then copy that out and work out what are the elements that work together. My arrangement is just a reduction of that huge stack ... I can just pile on top and experiment with different mixtures and then I know ‘oh that mixture is going to be this, and that mixture is going to be that’.” — Christiaan Ercolano

These different mixtures can themselves be captured in temporal abstractions as described in chapter 8.

Emile De La Rey was forced to work around his dependance on rendering to approximate voice abstraction by keeping un-amalgamated backups in separate project files. In order to change something in a rendered arrangement, he would make the change in the non-rendered project, render it again, and trick his software into accepting the newly rendered audio in place of the previous audio. This is an awkward but necessary process given the limitations of typical multitrack production software.

For the electronic group Pitch Black, accessing individual voices in grouped material is very important. They achieve this in a live environment by sending the different voices contained inside a pattern (see phrase abstraction in chapter 8) to different channels. Each of the channels are then manipulated independently on the mixing desk:

“We have patterns: some might be intros, full-on bits, and then a breakdown; but even in the most full-on pattern, Mike can still turn that into a breakdown by muting things on the desk.” — Paddy Free

In the studio environment, Paddy also aggressively applied rendering for voice abstraction, and like Emile, would have to move back to previously saved mixes to get access to non-amalgamated voices:

“You can deconstruct from within those [full] mixes completely. You can open out the full project down to its component wave forms again.” — Paddy Free

Producers’ dependence on project snapshots to store non-amalgamated voices requires meticulous project management. Due to the lack of direct software support for this form of abstraction, successful producers have had to learn the hard way to develop careful processes and judgement to work around this limitation.

7.6 Grouped voices and context

In systems that natively support some form of track amalgamation abstraction, participants had difficulty with actions that required them to see the components of the amalgamation in context. When looking at events inside one of these voice abstractions, participants needed to see how they were temporally related to events outside of the abstraction. This was best captured by Paddy Free who described how this was an issue in Apple Logic:

“That is when you get into putting things in folders and locking things down, which is good, but then if you want to get in and change something within the folder that is a can of worms to get your head around. Because once you jump inside a folder you lose the greater time line. You are only seeing the contents. . . . I can’t see it in relation to what came before or after.” — Paddy Free

This is a great example of how attempting to provide a means of abstraction can fail if it does not take into account how it will affect the activity in which it will be used.

Paddy described how his requirements could be better served if it was possible to unfold the abstraction in-situ, to allow inline editing of the abstraction with it displayed in its context of use, and thereby improving **visi-**
bility:

“If I want to get in and move something around I don’t have the visual paradigm of lining [events] up unless I explode the folder.”

— Paddy Free

In this case, exploding and refolding of track folders would require the constant manual recreation of track folders after the action was complete. Simon Rycroft echoed that this would be problematic for him:

“That would be an issue because I need to line things up. [Not being able to see them lined up together] is terrible because it means you can’t treat it in relation to everything else. How would you visually match up an element which is outside of the group? [It] would be a pain [to have to move things in and out]” — Simon Rycroft

For Simon, there was a more general difficulty here with needing to see any one track in relation to another for lining events up:

“I end up moving [tracks] together which disorganise my session which is a problem. It is actually a real pain in the ass, and my screen is too small: two fifteen inch screens. I often find [you have a part] and you want to adjust it to the kick drum, and they might not be visible together in the edit window. So you end up having to move them together and your session gets all shuffled together. What could be better is if you could select the tracks you could deal with and everything else just disappears like [Exposé] on the Mac. As opposed to selecting things to hide, if you could select things to work on, I think that would be a better way of thinking.” — Simon Rycroft

Evaluation question

- Does the system provide access to individual voices inside voice abstractions where multiple voices are amalgamated, and allow them to be compared with other voices from outside of the abstraction?

7.7 Summary

There were many occasions where producers' activities were at odds with the provided means of creating abstractions of *voice* parts. The track abstraction provided by the multitrack-mixing metaphor made it difficult to organise transient events, forcing participants to find a full length track on which to locate them despite their short duration. Large numbers of tracks resulted in a need for very frequent scrolling and zooming that some participants found frustrating. Many systems do not provide adequate means to abstract several voices into one without losing information through rendering, while systems that did could cause various tensions in producers' work. Some participants needed to quickly see sub-voices in context with other voices, and found that material in track folders could not be easily viewed in relation to external material. These findings reveal the various demands which producers' activities place on provided voice abstraction mechanisms, how use of these mechanisms affect their work, and where they are at odds.

Chapter 8

Temporal Abstraction

8.1 Introduction

Music is inherently temporal in nature, as it is expressed and experienced over time. There are several ways in which both composers and listeners experience musical abstractions over temporal periods. For composers, the representation of audio through time on the horizontal axis can be seen as a form of abstraction. Both composers and listeners experience two other types of temporal abstraction through grouping structure, and metrical structure. These two basic components of musical intuition are described in the Generative Theory of Tonal Music (GTTM) [95] introduced in section 3.2.1.

8.2 Hierarchical temporal grouping

One of the most important characteristics of temporal grouping structure is its hierarchical nature. As described by the GTTM, musically literate listeners perceive motif blocks as contained within phrases, and phrase blocks within sections. Sequencers such as Ableton Live provide features allowing the user to quickly create higher-level temporal blocks out of lower level blocks:

“The Consolidate command replaces the material in the Arrangement View selection with one new clip per track. This is very useful for creating structure.” — Ableton Live Manual [3, p.81]

Figure 8.1: In Ableton Live [3, p.81] and other DAWs, blocks of audio can be temporally aggregated to form a higher-level abstraction.

Figure 8.1 shows how three blocks of audio can be consolidated into a single larger block. Troy Kelly explained how he used a similar feature in Cubase:

“‘Events to part’ in cubase boxes [the events] and then you can bounce them to a selection. Once again it is just locked down. [It gives me] peace of mind and when I archive it, it is done.” — Troy Kelly

These blocking mechanisms allowed participants to arrange and manipulate their compositions at a higher level of temporal abstraction. Andy Morton described how Atari Notator allowed arrangement to be described with high-level temporal abstractions:

“I like it because [the arrangement] is loop and pattern based. You can layer the loops and the patterns, and it is just easy to copy things and layer it out like an equation, not like a linear

timeline based thing. It is more like one thing you can look at and see that's a song and there's the structure." — Andy Morton

Here Andy has also articulated another advantage of arranging at a higher level of abstraction: it can lead to terse (non-**diffuse**) and rapidly understandable arrangement descriptions. This is similar to how instrumental bands write down the sectional arrangement for reference during performance.

Troy Kelly used these higher level groupings to trigger sections in Reason's Redrum:

"They are like verses and choruses, all in Redrum. Everything is in Redrum. It is just there like 'A1' and I've just got it on key command — all the different patterns — so I can just go 'verse' and hit 1, 'chorus' and hit 2, and it is real easy." — Troy Kelly

Paddy Free used these higher level blocks for arranging in the non-linear system of Ableton Live:

"We have patterns. Some might be intros, full-on bits, and then a breakdown." — Paddy Free

Figures 8.2 and 8.3 show Paddy Free and Jeramiah Ross's MIDI keyboard modifications where controllers are annotated with the names of the higher-level temporal abstractions which are triggered when the corresponding control is pressed.

As with process and voice abstraction, an important reason for representing higher levels of temporal grouping abstraction is to enable information hiding for reducing complexity. Christiaan Ercolano explained how the MPC's lack of support for hierarchical temporal grouping forces a rendering approach, where lower level block groupings are lost:

"I tend to build it as a flat arrangement of [various repeated sequences], and you are trying to do it as efficiently as possible.

You are always trying to use an economy of sequences. You just don't want to just keep making more and more of these messy things that go here and here. *You can't see them as a group.*" (Emphasis added) — Christiaan Ercolano

However, in many cases, such as with Ableton Live, this temporal aggregation is a one way process, and inner structure is lost. Simon Rycroft explained how the destructive process of block consolidation in Pro Tools removed his ability to later review the substructure and edits underlying a block of material:

"It is called 'consolidate' and it is destructive. I often create a new playlist called consolidated. It would be handy [to be able to do a virtual consolidate] because you could go back and fix your edits later on. That would be good. The reason that you consolidate things is so you can drag them around and edit them as one instead of having to shift select them. Pro Tools will tell you you can't [select a number of blocks and edit them as one] because it has got a fade on it. You can't do certain edits [without consolidating]. If you could do a virtual consolidate it might let you do form changes or major edits on that piece of audio." — Simon Rycroft

Andy Morton also explained how the ability to both aggregate and de-aggregate was desirable, and gave an example from the old Atari Notator sequencer which he still uses:

"I like that within one bigger pattern you can have multiple patterns within that." — Andy Morton

Wayne Laid described another old sequencer that also explicitly enabled the representation of the hierarchical nature of temporal grouping:

“The sequencer called Dr T’s was a great one. It only gave you a data display, it didn’t give you notes or anything, but you could nest sequences. . . . Later programs never include all the neat things from the earlier ones.” — Wayne Laird

Despite tools to aggregate temporal blocks, many sequencing systems do not provide explicit support for representing the hierarchical nature of temporal grouping. Christiaan, well versed in producing tracks on the MPC hardware sequencer, described how he creates higher level blocks out of lower level block elements:

“Once you are in song mode you can make little groups of sequences and then bounce them out as a sequence. So you can go ‘cymbal hit, clean twice, dropped out one, clean twice, fill’, and that’s your first verse. . . . You bounce that sequence and you call it ‘Verse One’.” — Christiaan Ercolano

As the MPC does not support hierarchical sequence objects, this producer was forced to use the MIDI equivalent of rendering, thus losing the inner block structure of the verse in the bounce operation.

As well as providing a voice abstraction mechanism, Apple Logic’s Folder Tracks also provide for multi-leveled hierarchical temporal abstraction with some of the properties we have described. Clinton Bradley explained how these were central for the compositional activity:

“Folders are critical for song-writing and manipulation of a lot of data. . . . I might chop the five master folders into [sections] and I might put all five into another folder and chop that up.” — Clinton Bradley

Another example was revealed in observation sessions. Paddy Free created ABAC structures in Logic from audio material, spliced them into one clip, and then repeated this higher level block at various points in the composition. However, here, Logic did not provide adequate support for retaining

the multiple structural levels at which temporal grouping is organised. While Logic’s ‘folders’ feature can represent a hierarchy of temporal abstractions, its other limitations meant it was not suited to Paddy’s abstraction requirements.

While we have described the motivation for aggregating blocks into higher-level blocks, we also need to emphasise the importance of being able to access lower levels of this abstraction hierarchy. Illustrating the extremes of this, Paddy Free demonstrated how using Ableton Live he could cue major song structures from a single press of a button, while at the same time on another keyboard he was “completely live, just playing it as an instrument.” At an intermediate level he also had control over cueing intermediate level blocks:

“We’ve got it to the point where it is the equivalent of being in a band saying ‘take it to the bridge’. Plus I’ve allowed enough things so I can rock whatever. Or I can utilise the power of having freeze dried little snippets as well as the big chunks of arrangement. Once all the major teching is done it’s down to putting your hand up in the air and going slam and bring down everything, and that is instantaneous. . . . There is more spontaneity.”

— Paddy Free

At the upper end of hierarchical temporal abstraction, producers need handles on the abstraction at the song/piece level. In a studio environment this was adequately handled by saving each song to a separate project on disk. It was in the live context of participants’ activity where this did not work so well. As a performance set is usually made from a series of individual songs, producers need to be able to instantly switch from one song to the next. The MPC, like many sequencing systems, can only have one song loaded at once, and has an enforced downtime during loading where nothing can be played. Rhombus, who use the MPC live on stage, had to work around this by capturing audio to sustained outboard effects, and use the time this provided to load the next song. Of course it was far from ideal that this was required:

“Another thing that would be great is that if you could load while it was playing.” — Simon Rycroft

Paddy Free went to great lengths to enable him to move seamlessly between the many songs in a live set list, by carefully bringing all of the material into a single Ableton Live song file. However this was a time consuming process:

“That has taken all that design time and all that construction to do that, but ultimately at the end of the day I step up sequentially through 12 patches to play 12 songs. I can put the next strip on and everything is labelled.” — Paddy Free

More than simply being time consuming, this could also prove to be a complex and frustrating process:

“In [Ableton] Live, making up the megalosal live [performance] gigs, the two show loads for the two computers is an absolute head-fuck. Absolutely. Now in Live 5 you can drag clips back and forth which helps. [With the] paradigm in Live the easiest thing to do is to have everything in one song. And realising that, if you’ve got complex songs . . . [like] for us, we want to have eight outputs which means we need to have eight tracks at least per song.” — Paddy Free

Evaluation questions

- Can material be represented in a hierarchy of temporal blocks up to the set level, allowing information hiding when desired?
- Can the producer work with multiple levels of the temporal hierarchy simultaneously?

8.2.1 Ambiguous temporal structure

As in the case of voice abstraction, there can be ambiguity in how listeners will perceive temporal abstractions. This is explicitly addressed in the GTTM by the use of weighted preference rules [95]. A musical passage may have multiple contradictory (but still valid) interpretations as to how the musical events could be grouped, or various interpretations of time-signature and tempo. Figure 8.5 shows how a section of Mozart’s Sonata in A K. 331 has two possible groupings.

Figure 8.5: A passage from Mozart’s Sonata in A K. 331 with two possible temporal groupings, as described in a General Theory of Tonal Music [95, p.63]

At the extreme, sequencing systems can even represent temporal abstractions that do not match any reasonable perception of structure. This can occur when the grouping structure reflects how a piece was spliced together, rather than the listener or composer’s perception of grouping. Paddy explained how compositional devices can be used to make an intended structure obvious to the listener:

“You need markers There is some sample on a sixteen bar cycle coming in to just hold the hand of the listener through the thing.” — Paddy Free

This structural element of a composition may or may not be represented in the sequencer software.

In practice, as revealed in observations, producers constantly break and merge blocks together in order to serve their current objectives.

The GTTM described two of these complications in hierarchical descriptions of music, where material may be in more than one group at a given level. This occurs in cases of overlap and elision. Overlapping groupings occur when two neighbouring blocks share events. This would commonly manifest where the final material in one phrase also serves as the material for the beginning of the next. For example, the last note of an arpeggiated run of notes could be held, acting as the first note in a series of long tones.

When events from the beginning of a block interrupt the end of the previous block we call this elision. In this case, each event belongs to a single block, but the blocks coincide for a time.

“I would join the two folder tracks together. I do what I needed to do, and then if I needed to resplit them I would manually split it again.” — Clinton Bradley

Evaluation question

- Can multiple contradictory temporal abstractions be represented?

8.3 Linearisation

We define “linearisation” as the actions of creating a linear arrangement from the multitude of possibilities afforded by the composition material. At the heart of this component of production activity is one of the most pronounced differences between many DAWs: their choice of linear or non-linear temporal models, and the consequent abstraction mechanisms that they provide. DAWs implementing the multitrack recording metaphor utilise a linear model. Moving between the two models could be problematic for participants:

“That is a disadvantage of the MPC. I would really like to be able to get out of [linear] song mode and go back into [non-linear] sequence mode [and get back into song mode again]. That would be awesome. That is one of the things I would love to see on the MPC”. — Simon Rycroft

In our field studies, this dichotomy revealed each model had its own benefits and liabilities, which we now describe.

8.3.1 Forming arrangements

One of the most obvious needs served by systems using linear models is providing the ability to form finalised arrangements. Paddy Free explained linear systems as enabling the creation of structure, rather than keeping material in a state of flux. Linear models help ensure that all work is leading towards a single concrete arrangement. One advantage of this is that this structure becomes directly visible. For this reason, Troy favoured the linear model for his arrangement work:

“...If you want to visually see the verses and the choruses and stuff like that [you need a linear view]. I’m not really thinking about creating it [in the non-linear view] because I find that a bit time consuming, whereas when it is [linear] you can create the song how you want it.” — Troy Kelly

While Paddy Free was a strong proponent of the non-linear approach, he also used systems with a linear timeline model, due to its strength in representing a final structured arrangement:

“Music by definition is structure: time, air-pressure waves, arrangement, whatever. Music exists in time and so if I can at least rewind the tape, go back, because you are always striving for some form of structure. Structure is the goal... Ideas start

out in the [non-linear] view, and then get hammered out into rough arrangements in Ableton Live’s [linear] view”. — Paddy Free

Like Paddy Free, Simon Rycroft from Rhombus also used a non-linear approach much of the time, but favoured the linear model imbedded in Pro Tools for the arrangement process, as it enabled him to lay out the structure of a piece. Without tools to enable a linear description, producers tended towards loop based approaches, with structure depending more on varying the layering of repeating parts, rather than varying the parts themselves:

“After the fact you’re just turning tracks on and off to arrange . . . which is not the most beautiful way to write a song because you always throw things in as loops. . . . You’re not doing a linear path of what a song should be. . . . That arranging side of it was always hard. It was always way too miserable to think about. It is so much simpler in [Apple] Logic in a linear format.” — Christiaan Ercolano

Like many participants, Emile also used a layering approach to composition, even though he almost exclusively used a linear DAW:

“Usually because for me structure involves adding or taking away parts or clips.” — Emile De La Rey

However, much of his time was spent creating one-off variations in repeated parts in the linear timeline which demonstrated the strength of the linear model, even when working with a layering compositional aesthetic.

Evaluation question

- Can the producer build and view a piece’s entire linear structure?

8.3.2 Material in arrangement context

Apart from enabling producers to create and visualise a structured arrangement, systems using the linear model can also make it easier to create and manage transient variations. During a piece, many small variations in parts were often created, either to add variety, or to ensure events in one voice are well matched to those in other voices.

Without a linear representation it can be difficult to create these variations. Producer Christiaan Ercolano described how in a non-linear environment each of these variations would create an explosion in the number of temporal blocks:

“So your simple two bar turnaround has turned into one [block] with a cymbal at the front, one clean, one with a drop out for the special word [in the lyrics], and then another one with the fill. So what started off on day one as four [repetitions of] two bars is now four different sequences. . . . Suddenly bars have to be thought of as a part, and that is a much slower [process]. . . . That kind of one-off stuff we do in Pro Tools.” — Christiaan Ercolano

This is because in the linear model found in Pro Tools, all of these blocks are concrete representations in a specific temporal context and can be easily varied on an individual basis to match coinciding content.

Paddy Free explained that linear descriptions used in the studio allowed for more precise control, as opposed to the spontaneity required in a live environment, and Troy felt that using the linear model was less time-consuming. Clinton Bradley also spoke about the importance of being able to vary material depending on its location in the development of a piece, and how this could be more difficult for those using non-linear triggered systems:

“The triggered guys run into problems all the time. . . . As far as I’m concerned all songs need to evolve so I definitely make variations as the song goes on. It should begin simply and end

more dramatically. I do that all the time, repeating elements doesn't mean they have to be the same every time, the filters might open up more ... it gets more intense." — Clinton Bradley

One vital aspect of arrangement was ensuring that compositional decisions reflect the full context in which material is performed. The principle example of this was that arrangement timing decisions need to take into account the context of what has come before. Paddy Free and Troy both described how making the correct decision required them to listen to the piece from the beginning so as to hear where the material might be used in its full context. In Troy's words:

"It is about not just putting something in there and saying 'yeah that is it'. It is about listening to it and [deciding] what happens next." — Troy Kelly

Additionally, producers could be very concerned with how material transitions from one block to another. For example, Paddy Free discussed how he would spend "... an hour getting one dub sound to go from verse to chorus just right". In other words, the immediate context of any block determined the important details of how neighbouring material should be performed in order to facilitate desired transition effects and sense of flow. Both Troy and Clinton Bradley spoke about this at length:

"It makes me think about when you read music you'll have a repeat, but it will have a D.S. al Coda or a B ending that is the ending for the second time. I'd probably just record the arrangement [into a linear sequencer] and do something with automation." — Troy Kelly

"Obviously if you are going to go to a different [section] you are going to play the previous part differently [than] if you are looping. — Clinton Bradley

Clinton also described how instrumentalists will anticipate the chorus in various ways, also emphasising the importance of individual transition points between sections. Simon Rycroft also had relevant comments:

“[When arranging with a linear sequencer] you will have to go to each individual track so that the transition is occurring where there is no audio, or an appropriate place for a cross fade [and you will drag out] the anacrusis for any given instrument. Ideally you’d have to drag out the automation.” — Simon Rycroft

Simon then discussed how it could be problematic to achieve the same results in a non-linear sequencing system:

“With [non-linear] pattern based arrangement you wouldn’t be able to effectively build the end of a verse into any particular transition point. You might have a certain fill that is perfect for the chorus, but if it was running into a bridge you might not need the fill in that case, or there might be a different fill. . . . [When] changing the form you would need to have clear start and end points that line up on the beat, so if you have an anacrusis that could make things difficult. Or if you had sustained effects coming into the chorus, that could cause problems. ” — Simon Rycroft

Finally, as a follow-on consequence, changes in any one part of a composition also change the context of other material, and can create significant side-effects. As a result, arrangement changes may be needed to correct the balance of the composition, further emphasising the importance of arrangement flexibility as discussed in the proceeding section.

These examples illustrate the importance of context on compositional decisions which are in many ways better served by a linear approach, rather than systems using a context-free non-linear model.

Evaluation questions

- Is it possible to specify variations in repeated material to ensure diversity and to ensure coinciding parts mesh?
- Can neighbouring and global context of repeated material determine the details of how it is performed?

8.3.3 Flexibility

The most pronounced ability of non-linear DAW systems is enabling spontaneous rapid rearrangement. This flexibility of action approaches the experience of working with musicians in a band situation, where specific arrangement decisions have a high degree of **provisionality**. As already quoted, Paddy Free expressed it in exactly this way, that “it is the equivalent of being in a band saying ‘take it to the bridge’”.

This idea of matching the flexibility of a band is a crucial one. In a band environment each musician naturally develops an understanding of the temporal grouping abstractions in the piece and being able to compose, rehearse, and perform at this higher level can make things much easier. We can contrast this flexibility to the poor experience of attempting to arrange with low-level blocks in a linear DAW:

“Say if I have a big Apple Logic song, if I want to say ‘take it to the bridge’ I’d have to cut the bridge out separately, find it, drag it, cut and paste it around my timeline. Whereas with Live I just [cue] the scenes. Or when I get to a point I go ‘capture scene’ and now I can jump to that state at any time.” — Paddy Free

“It gets messy when you are doing lots of changing around with big arrangements with lots of things [going on]. You do a lot of cut and paste, and then ‘naa, I changed my mind’, cut and paste again, and then ‘naa, changed my mind’... In a linear timeline situation [like this] it is hard to move big chunks around. Quite

often you have nowhere to put them. It is hard with Pro Tools if you are tracking on a vocal, you just keep bringing up another track but they're all on the same track but you have different play lists. So if you are dealing with a whole band you have eight tracks of drums, and you have a bass, guitar, and then you've got a playlist and you start getting more things that can go wrong. . . . But it probably stops me experimenting too much because it is just a bit of a headache." — Andy Morton

"Pro Tools doesn't have much capacity for real-time rearrangement of loop snippets of audio the way that Live does." — Simon Rycroft

Systems providing this level of flexibility of arrangement have a strength in facilitating the producer's creativity. By being able to arrange in real-time, producers can try out new ideas very rapidly and create results they would have otherwise likely overlooked:

"Live lets you Jam an arrangement. You can't really jam an arrangement in Logic and have it recordable and fully editable. . . . Once you've got all your elements immediately triggerable new arrangements reveal themselves." — Paddy Free

"In [Ableton] Live I try to jam my arrangement and try some different arrangements." — Troy Kelly

"[Real-time arrangement] means you can work with live musicians, and jam on a track and decide communally when you will go to the chorus, it gives you instant manipulation of form. It means you can be creative and get a good feel for the track." — Simon Rycroft

The other benefit of this flexibility of arrangement is that it works well with the state of flux in which electronic music often needs to remain. As

described in the discussion of process abstraction (section 6.5) Simon Rycroft, Module, and Paddy Free all spoke about how effected material could radically inform the possible arrangements of a piece. This state of flux can remain late into the creation process:

“With Pitch Black, everything is in a state of flux right up to the end. . . . If Mike makes a big build [up], that will change the feel of a section, so the whole piece will have to be rearranged.” — Paddy Free

“We’ll often go back. We don’t often say ‘well that is done, lets move on’. Nothing is ever set in stone.” — Simon Rycroft

The form of arrangement flexibility provided by high level temporal grouping and a non-linear model is very useful when performing in a live context. All but one of the participants used systems with these capabilities when playing live. One participant was so dissatisfied with the dominance of linear sequencers that he had developed his own music programming environment for live performance:

“It is quite difficult to make live performances with electronic music because it lends itself towards more post-production.” — Alex McLean

It is important to remember that live performance often comes towards the end of the production process, and so rearrangement possibilities are almost always important at this very late stage.

This also facilitates interaction without graphical display (e.g. for stage performance) where arrangement blocks are assigned to various hardware triggers and activated through gesture. Section 6.7 described some of the listening biases that can be introduced when producers rely on visual descriptions of the material they are working on. Manipulating arrangement without graphical feedback in this live context, regardless if in the studio or on the stage, can help the producer experience the composition as a listener:

“I always found once we got off the computer onto the MIDI sequencer, taking away the visual domination of the thing always lead to new ways of relating to the music because you are only hearing it.” — Paddy Free

“Rather than looking at it visually you can feel it musically. It is better from the outset, creatively it is better for sure. . . . If Pro Tools had it, yes I would use it, although I can’t see how it could work in Pro Tools.” — Simon Rycroft

Clinton Bradley also felt he would use an eyes-free arrangement performance mode if it was available in Apple Logic, his DAW of choice.

Evaluation question

- Can arrangement decisions be radically changed throughout the entire production process, including in a live eyes-free improvisational context?

8.3.4 Lead-ins

Material temporally grouped into blocks needs to be placed onto the metrical grid. In cases where material has a lead-in, either an anacrusis or a small quantity of lead-in sound, the point at which this block “begins” is an arbitrary distance before the metrical pulse with which it should be aligned. This was a problem for Christiaan when working on the MPC. If a sample began before beat *one* of the bar he would have to chop the sample into two blocks, one of which would be placed at the end of the previous bar. As Christiaan pointed out, with linear sequencers the producer can just pull the sample slightly into the previous bar:

“[With non-linear sequencers like the MPC] everything has to start on a one. . . . In Pro Tools you can willy nilly just go a

cross-fade things over and start them a little bit before the one.”

— Christiaan Ercolano

Some linear sequencers like Logic provide a “Region Anchor” that allows temporal blocks to be placed in reference to this internal reference point, rather than by the point at which they begin. Non-linear sequencers traditionally do not provide a good solution to this as described by several of the participants:

“[In Ableton Live] you just have to take a 2 bar phrase.” — Troy Kelly

“That is a real pain.... All these clips are triggered on beat one, but you have a particular sound that is triggered slightly before beat one. It makes it difficult to manipulate the clip in real-time in a live environment.... I’ve often thought about [this] in Ableton [Live], the sound might be half a beat or quarter of a beat before being on the one, but you still need to decide to trigger that thing four bars earlier.” — Simon Rycroft

“Humans don’t play exactly to the downbeat, so when you chop up [material you will lose the upbeats]. So what I do is take a little bit before, maybe a quarter note before because you can then always drag the contents of that a bit earlier.... That is more of an issue for triggerer guys. Triggerer guys are always dealing on the one, you definitely do have to take into account anticipation. It is really a live music issue. The second you add a vocal you have a real musician who is not going to be on the one. If you are just chopping things into very hard sections that can be an issue.” — Clinton Bradley

Evaluation question

- Is placement of material against a metrical abstraction mediated by a system to take lead-ins into account?

8.4 Rhythmic structure

The second component of temporal abstraction introduced at the beginning of this chapter is based on metrical structure. To recap, with metrical structure perception “the events of the piece are related to a regular alternation of strong and weak beats at a number of hierarchical levels.” [95].

8.4.1 Tempo and Time Signatures

Tempo determines the number of pulses over time, expressed as beats per minute (BPM). Participants varied tempo between songs. Regarding changing the metrical structure of a song, Andy Morton said:

“It is a little bit tricky going from time signatures or tempos. But I don’t do that very often.” — Andy Morton

Andy’s underlying reasons for not varying tempo and time signatures are difficult to unpack, but they may in part be due to the socio-technical history of the hip-hop style in which he works:

“Quite apart from the sound capabilities of drum machines such as the TR-808, the characteristics of the instrument’s operating system also appears to have had a direct influence on the musical style of rap during this period. Tom Silverman of Tommy Boy Records has stated that ‘the 808 forced you to program in a hiphop style You couldn’t program in real-time ...you had to drop the beats into a certain framework. Everything sounded ultra-mechanical. That’s partly how the hiphop sound originated”’. — Théberge [133, p.198]

In any case, other producers did find a need to manipulate metrical structure as described shortly below.

Time signatures determine the pattern of strong and weak beats. As with most popular music, participants predominantly used the common 4/4 time

signature, a repeating pattern of four foot-taps with stronger beats falling on the first and third. However, participants also used various other time signatures during observation.

In the most basic case, producers sometimes work without explicit support for metrical abstraction. This occurred during observations when producers recorded audio without synchronising it with a metronome. In these cases, the principle difficulty is that editing and moving material without a metrical grid is a very laborious process:

“It is harder because you are working with exact audio. There is more work involved in cutting it and moving it to the right point.” — Emile De La Rey

“You can’t move shit around if you don’t use a click track.” — Troy Kelly

Simon Rycroft sometimes found that Pro Tools would play material out of time as it did not lock it to the metrical structure:

“When you are in grid mode the timing is not accurate. If you copy and paste an eight bar loop through an entire song, by the end it will be slightly out of time, whereas the MPC is constantly quantising everything correctly.” — Simon Rycroft

In observations with Eric Vani and Jeramiah Ross, both edited material to fit when it had been captured without being locked to a metrical grid.

Most systems provide at least a basic level of temporal abstraction where tempo and time signature information can be entered to create the metrical grid for the piece. If material is captured against this grid it can then be moved and edited according to the abstraction that this grid provides — in terms of bars, beats, and sub-beats; rather than raw units of seconds. Unfortunately this relatively low level of abstraction in of itself does not allow producers the flexibility to easily change metrical decisions after material has been captured.

“Like most sequencers you have to decide on the tempo first.” — Andy Morton

“When we are doing production work we don’t use the MPC because massive changes are very difficult. Tempo changes and things like that, woah big time pain in the ass.” — Christiaan Ercolano

“The hardest thing to do would be to change the tempo. Especially if you are using heaps of audio.” — Emile De La Rey

“... you’d have to say what is the tempo going to be to start with, and after that certain things are undoable.” — Paddy Free

Another problem can occur when producers are not able to vary metrical structure over the course of a piece. Ableton Live only provides block triggering on a single metrical grid, with a static tempo and time signature. This forces producers to come up with workarounds that are less than ideal:

“Dave Hill from Ableton is a drummer, so he solves [Ableton] Live’s inability to switch meters by setting quantization to quarter notes, which lets him trigger sophisticated rhythmic changes live. (I’m practising so I can get to that point instead of using the bar.)” — Kirn [87]

Recent advances in practical real-time DSP have allowed the tempo of recorded material to be changed independently of pitch. This allows producers to make changes while working with the abstraction of tempo and have the concrete audio change appropriately:

“[Ableton] Live has come closest in that pitch and time are malleable instantly with no preparation ... so that has blown the doors off two huge paradigms... That is why I’m so buzzed about it, from knowing the effort required to do that... In Live

you can make as fundamental decisions as tempo right at the very end so that is pretty wicked: allowing you the leeway of time to make major decisions.” — Paddy Free

But with these advances, music software still has progress to make in providing flexibility of metrical control. For example, in a band situation the musicians adapt to tempo fluctuations and in fact make them part of the musical expression of performance:

“To pick up and slow down tempo as a band would [be great]. . . . To be able to tie that in more spontaneously, be it the band shifts up 2 BPM as it goes into the chorus subconsciously, or whatever. . . . For music software or technology to be able to track that from me more accurately would be cool.” — Paddy Free

Evaluation questions

- When material is captured without correct metrical abstraction information, can it be added after the fact?
- Do the metrical abstractions provide facilities to change tempo and time-signatures at any time?

8.4.2 Rhythmic effects

By allowing producers to manipulate metrical abstractions, they are able to create important metrical effects.

Andy Morton spoke about how he would use asynchronous repeating loops to create variation as they phased in and out with each other. He described the resulting material as being “repetitive without being repetitive.” This technique creates polyrhythms which are an important part of much popular music.

“Polyrhythms are quite difficult in sequencers. I tend to enjoy using those a lot. Very long term changes over time. Make a pattern to change it over time.” — Alexander Jensenius

Another important rhythmical effect used by several producers is ‘swing’. Swing is created by a systematic shift in the timing of the metrical pulse with the duration of pairs of notes being increased and decreased respectively. Swing was a vital concern for Andy Morton and Christiaan Ercolano in giving the music they created the right ‘feel’. During observations, to create a more human sounding feel Andy Morton and Christiaan Ercolano both moved samples earlier or later against the metrical grid so as to feel ‘sloppier’ and more natural:

“To make things feel sloppy and loose on snares and stuff I’ll always leave a little bit of air on any of my samples so it leaves things late. So often real micro tuning or timing with start points rather than the sequencer.” — Christiaan Ercolano

This kind of manipulation led to issues in how events are placed in relationship to metrical structure.

The creation of rhythmic effects depends greatly on how events are located relative to the operation of metrical structure. We discuss this in more detail below.

8.4.3 Event and metrical structure relationships

A vital distinction in musical abstraction is the difference between measuring a location in terms of time (such as in seconds), as opposed to the more abstract notion of location in terms of a metrical grid (such as beat or bar). Any event in a composition can be associated to the abstraction of a particular bar, beat, and subdivision. In this section we explore participants’ requirements to operate at multiple temporal levels, and then go on to describe how exploiting the relationship of events to metrical structure can improve the situation for producers.

Working at multiple temporal levels

Modern sequencing systems predominantly present material on a linear horizontal axis representing the passage of time. The advantage of this model is that there is a simple and intuitive relationship between the location of material on that timeline, and the relative time at which it will be heard. Unfortunately, this visualisation is not particularly adept at rapidly conveying precise timing information for such relatively sparse events. If we think about a producer creating a one bar drum beat, they will be placing most events at no more than sixteen divisions of the bar, and many at eighth and quarter divisions. At a common tempo of 120 beats per minute (bpm), that is equivalent to a maximum resolution of 125 milliseconds between divisions, and more commonly working on the order of 250 and 500 milliseconds. However, nuanced rhythmical performance has been shown in studies to require resolutions down to the 10-20 millisecond range [67]. Slight variance in timing at this almost imperceptible time scale are vital for the ‘feel’ of a rhythm, creating effects such as swing, and the unique character of particular instrumentalists.

“[Minute timing changes are] important for creating a groove, and the way that certain beats fall on the bar. The grid is exactly divided, but to get a good groove you often need to move certain elements of the track slightly ahead or behind of the beat. I achieve that either by getting it right beforehand in the sequencer, or I will use the nudge function in Pro Tools.” — Simon Rycroft

Interviewer: Do you find yourself fine tuning timings?

“Hell yeah, all the time. That is a huge part of music production. It is probably the most time consuming part of music production and it is not necessarily a good thing.... I am working with those all the time for feel and dramatic effect. If you have two kick drums at the same time, not only do you have to [equalise] them properly, you have to make sure they hit at exactly the

same time or not exactly the same time to make sure they don't cancel out." — Clinton Bradley

As a result, producers using a linear scaled timeline with a zoom level appropriate for editing at a sixteenth note time-scale are working at a level ten times greater than that required for nuanced rhythmical editing. The resulting **visibility** issues are often much more pronounced where producers are actually working with building rhythmical phrases over four bar or larger time scales, at the relatively vast 16000 millisecond zoom scale.

"I'll zoom right in on the snare, I'll listen to it thoroughly and then zoom right *into* the snare [modify the timing slightly] and listen to it again." — Troy Kelly

The reverse is also true, as zooming into an extremely fine granularity makes it impossible to perform edits on the order of sixteenth and quarter note divisions, let alone multiple bar patterns:

"With these visuals you can't see the difference. If you are down into this minute resolution you can't really see the difference. You need to have something that has more scope for change." — Andy Morton

"The great thing about the nudge thing is you can work at a higher level, select a whole track and use the nudge [tool]. You are working at such a high level you can't actually see it moving. [It would be good to have two levels of zoom so] you would be able to see a beat but you could see where it is in relation to [its context]." — Simon Rycroft

"I've got key commands [to jump between] zoom levels. Like when I do my phase alignment, so I can zoom right in and have a look at a phase, zoom right out, move to the next section, and zoom in. I'll eventually [group the larger section into a part object with the] event to part [command]." — Troy Kelly

“I’m working at all levels all the time. The beauty of [Apple] Logic is it does let you view the minutiae and the whole thing at the same time.” — Clinton Bradley

Editing and metrical abstractions

The common music notation system bypasses this problem entirely by simply ignoring it. In this case, the notation system only provides a resolution at the macro-level, and leaves the minute matters of feel to the skilled interpretation of performers. Of course scores in the common music notation system are augmented by performance notes describing the desired interpretation to influence the performer in an appropriate direction. For example, the term ‘ritardando’ instructs the performer to expressively slow the tempo, while a ‘swing’ instruction would result in a change to minute but crucial timing details of the performance. As another example, it is not uncommon to see instructions such as ‘lazy’, which would result in slight delays applied to certain events.

Apart from avoiding the need to describe the details of micro-timing, traditional notation is so useful precisely because of the way in which it employs abstraction of metric structure. In fact, it is due to the use of higher level abstraction that traditional score notation is able to avoid micro-timing details in the first place. Common music notation cannot specify an event be performed at an arbitrary point. Instead, each rhythmical inscription must have a start point and duration that is directly related to the metric structure of the current context in which it is performed. This allows composers and performers to work in the currency of bar, beat, and sub-beat locations; and quarter, eighth, and sixteenth beat durations. On the downside, this useful level of abstraction comes at a learning cost, as it does not have a direct mapping to a linear timeline view. Additionally, when common music notation is used in the electronic music paradigm, producers are unable to use it to control the fine levels of timing required for nuanced expression.

One notation that allows for editing through the structure of metrical

abstractions is the MIDI event list. Event lists typically take the form of a table, listing various event details. Like traditional Common Music Notation, event lists typically display timing information in the abstract bar, and subdivision format, rather than depending on absolute times. Figure 8.6 shows the event list editor from Atari Notator.

1	2	3	4	5	6	BAR	/	4	16	768	STATUS	CHANNEL	-1-	-2-	Length/Info
■						4	1	1	1	NOTE	1	A#3	92		2 19
■						4	1	1	1	NOTE	1	C4	104		2 47
						4	3	3	1	NOTE	1	A#1	106		47
						4	3	3	1	NOTE	1	D#3	107		1 47
						4	3	3	1	NOTE	1	G#3	106		1 47
						4	3	4	1	NOTE	1	B1	106		47
						4	4	1	1	NOTE	1	C2	105		47
						4	4	1	1	NOTE	1	F3	93		1 47
						4	4	1	1	NOTE	1	A#3	99		1 47
						4	4	2	1	NOTE	1	B1	98		47

Figure 8.6: This event list editor from Atari Notator allows producers to work with metrical abstractions of bars and beats.

In this particular example, all of the events shown occur in the fourth bar of the displayed sequence, as shown in the ‘BAR’ column. The following two columns determine the event’s placement in that bar, the quarter, and sixteenth respectively. The additional ‘768’ column controls the fine variance of each event (at a resolution of 768 parts) enabling the producer to manipulate micro and macro timing values in the same interface. Andy Morton explained why this sort of interface was very powerful for manipulating feel:

“I think it is more accurate. If I see a number, it is not on the one. If it is on the beat before, on a forty-eight, it is a bit early. If it is on a six it is a little bit lazy. I know that you can go up to a fifteen and that is extremely lazy, so you have that scope.”

— Andy Morton

However, as with traditional scores, this abstract notation also has a learning cost:

“I know a few others who used this and it took them ages. They couldn’t understand how I could do it so fast. Once you’ve learnt that, it is really fast.” — Andy Morton

As with these examples of numerical micro-timing values, both Andy Morton and Christiaan Ercolano spoke about specific swing ratio figures and the quantity of audible timing effect that resulted.

Evaluation questions

- Are producers able to manipulate material in terms of bars, beats, and sub-beats through provision of temporal editing and visualisation over metrical abstractions?
- Are metrical structure abstractions provided, allowing producers to learn common parameters for expressive timing?

8.5 The editing / performance divide

There were two predominant modalities used by our participants for interacting with gesture representation systems such as MIDI. The most concrete method was to capture and refine material through performance gesture. The alternative was to enter and refine material through editing an abstract notation (as seen in figure 8.7), typically via a mouse and keyboard.

Our participants used a hybrid of these two modalities, moving between the concrete and abstract manipulation of material as required. The amalgamation of these two modalities creates a divide in producers’ actions as they move between the two:

“My intent, once it goes into the computer, that is now de-coupled from the visual, actual moment of bashing that [piano key] down.”
— Paddy Free

Figure 8.7: The piano-roll editing interface found in Apple Logic.

This divide has a number of consequences for producers, which we discuss in the following sections.

8.5.1 Performance gesture

Interacting with sequencers via pure performance gesture is the more concrete means to create material. In this study, all producers using traditional sequencer software captured at least some performance gesture by hardware controllers such as MIDI keyboards, electronic drum surfaces, turntables and mixing desks. In this modality, the musician works in real-time with any accompaniment playing alongside. Events are created at a given point in time in response to a physical gesture occurring. This mode of input demands a level of virtuosity from the user, although a captured performance can usually be augmented by editing the resulting notation. While performing can be more demanding of physical skill than entering material with an abstract editing interface, there are important advantages to working in this more direct manner, with its closer mapping of action and result.

“[Ideally] you are making a beat by physically tapping out a rhythm in time with your hands. Being able to play a bass line, or

play a synth, or manipulate an effect with a physical movement makes so much more sense. That is how real instruments are played. That is how guitars are played, how pianos are played. It is a rhythmic movement. . . . You are playing the music rather than just allocating values to aspects of it.” — Andy Turley

“I don’t really like composing with a mouse. I much prefer to use the MPC. . . . It is much more like an instrument than a computer. . . . It really is so preferable to working with the computer as far as the initial composition goes.” — Simon Rycroft

Also, at a higher level, Paddy Free, Simon Rycroft, and Andy Turley all spoke about the utility of triggering loops and song sections to through performance gesture to control arrangement structure.

Refinement through performance gesture

Some performance gesture systems not only enable the capture of material, but also provide for the refinement of material through gesture. This is common on older sequencer drum machines that do not provide a graphical user-interface, which instead provide an erase button. Pressing erase while a recorded event is being played back removes that event from future playback. Andy Morton, who is very familiar with this form of working described his thoughts on this:

“It is a love hate thing with it all because it is a drag that you can’t do minute editing. You can’t just go and shift everything. But there is this great discipline that it teaches you, because you have to musically remember it was [for example] the second beat of [the groove]. You can’t go in and find it and edit it. . . . You have to erase it by feeling it. It is a good discipline to learn, and it teaches you a lot just because of that — because of its limitations. A lot of people with computers . . . forget about the performance

elements of it. So this is one that forces you to perform. . . . There is no undo, there is no nothing. Visualising what you want takes a lot more discipline, as a performance which takes a lot more discipline, it's great." — Andy Morton

Despite the positive aspects of using this method, Andy Morton's description also betrays the difficult and painstaking nature of refining material in this manner. Even with the ability to erase events via a performance interface, Christiaan Ercolano expressed how he would sometimes re-record a phrase rather than refine it in this way:

"You tend to be more of a musician on it. You just go 'fuck, I fucked those two bars up, I'm going to do it again.'" — Christiaan Ercolano

This is forced by the fact that available systems provide little means to refine captured material other than through selective deletion and overwriting through re-performance, as already described.

Evaluation question

- Can material be captured and refined through performance gesture?

8.5.2 Editing notation

Editing an abstract notational form provides a very different experience to performance based editing. The abstract nature of notation editing can prove problematic when producers are wanting to work at a concrete level. In these cases there is a conceptual gulf between the producer's intention and the abstract nature of the notation system. This is particularly important when creating material:

"I would like to be able to just program drums really really naturally without having to worry too much about clicking on the

sixteenth, and clicking on the eighth. That's a really mundane way of programming drums. When you are actually painting, when you have your beats in a bar and you have to click on the first and the forth. You click click click click click, then you play the loop, and you go 'na that's not right', turn that off, turn it on here . . . it is in your head. But with a pad system you can just [perform]." — Andy Turley

However, one advantage of the abstract nature of editing notation to capture material is that it removes the real-time requirements of performance:

"I just find it faster. Fine tuning . . . I can make a beat faster than you can play the beat." — Andy Morton

Refinement through editing notation

For participants, the most common way to refine material was by editing the notation. However, Paddy Free explained that this was far from ideal:

"The editing of mix automation [probably takes the most mental effort] because that is the biggest shift from what I am doing. The divide between recording a 'take' of automation is the most removed from editing and tweaking automation. It is the most mentally intensive because it is the most operations to describe something that is actually very small and quick." — Paddy Free

Here we clearly see the divide between the underlying performance gesture and the mechanism of editing notation. This becomes particularly problematic when a performance is close to the goal performance, but needs some refinement:

"You don't always want to just do another take because what you've got there is 70% perfect and you don't want to lose that."
— Paddy Free

Troy Kelly used editing to refine the captured performance of instrumentalists in his role as producer. In one case he modified the octave (pitch) and timing of a keyboard performance. The advantages of refinement through editing were several:

“Being able to do it myself later on, being able to sit down with no one else in the room and do it thoroughly and precisely. It is not locked down if [the performer] has made a mistake.” — Troy Kelly

When performance gesture falls just short of the producer’s target they are forced to deal with the operation intensive nature of editing notational forms, rather than being given the option to refine their performance *through* performance. As we already described, the options for refinement through performance gesture are currently quite problematic.

8.6 Summary

Mechanisms for *temporal* abstraction have a significant impact on the activity of producers. Systems that do not provide abstractions for creating multilevelled temporal structures make it difficult to do rapid rearrangements at various scales, something that is very important for producers’ creativity. Producers working in a live context had difficulties when they were not provided with abstractions to manage entire songs for set arrangement and transitions between songs. At smaller time-scales, producers often needed to work with microscopic timing details to manipulate ‘feel’, while simultaneously working at the larger abstract level of metrical pulse — a tension that was not well served by common systems. One of the most thought-provoking aspects in which the provided means of temporal abstraction impacted producers’ activities was in the dichotomy between linear and non-linear sequencing paradigms. Participants’ patterns of work required

them to repeatedly move back and forth between the two models during various phases of the activity, losing temporal structuring information at each step. A final highlight of our temporal abstraction findings was the awkward divide between the mechanisms for recording performance through gesture, and editing it through notation. In particular, while skilled performance could be captured through quick and relatively simple gestures, editing this performance through notation could be awkward and laborious.

Chapter 9

Reuse and Versioning Abstraction

9.1 Introduction

Reuse is a vital component of creating many types of complex digital artefact. In this chapter we describe ‘reuse abstractions’, which provide models of how reused content is represented and manipulated throughout a user-interface. In the case of computer mediated music production a great deal of reuse takes place, and our study of producers’ activity systems exposed the vital importance of the mechanisms that enable this. Reuse abstraction mechanisms are crucially important in music systems primarily due to the role of repetition in common musical forms. Musical styles such as blues, jazz, rock, jigs, country, flamenco and many others derive much of their identity through the repetition of distinctive rhythmical and chordal patterns. The classical composer Schoenberg wrote that “a motive appears constantly through a piece: it is repeated . . . , the repetition may be exact, modified or developed” [113]. As we saw in figure 3.1.E, Common Music Notation highlights the importance of repetition through its provision for repeating section blocks with specialised bar line markings, 1st and 2nd time repeat bars, D.S. and Coda markings, and repeat previous bar markings. The common compositional techniques of

“canon” and “theme” repetition, as well as the repeating verse and chorus forms common to all pop music demonstrate this also.

The use of repetition that is so central to many musical forms is tempered by its partner in the compositional term “theme and *variation*”. The variation of repeated material is a core component of composition allowing for development and much musical interest within a piece, as well as allowing several performances of a piece to differ from each other while remaining recognisable. Another aspect of musical reuse can be found where material from a repertoire is employed at multiple points within a piece or across different compositions. This manifests in computer music systems in “libraries”. We found these three aspects of reuse (repetition, variation, and repertoire) were all crucial to our participants’ musical activity, and are supported to varying extents in the abstraction mechanisms of their software systems.

9.2 Copying versus referencing

Two common approaches for reusing musical material were used by our participants. The first is the ‘copy and paste’ approach where copied material is duplicated and completely independent of the source material. The second ‘referencing’ approach allows the producer to create multiple instances that are kept in sync by all referring to the same underlying material. Participants used various implementations of copying and referencing schemes, which revealed underlying tensions between the two approaches, and how they affect the activities of producers. We describe the most important of these issues now.

9.2.1 Copying

Copy and pasting material was a crucial part of how participants created arrangements. One of the most significant features of using copying was that producers were always working with concrete representations that could be

manipulated directly. Copying was often followed by altering details of the copy to create a variation from the source material.

“Once you start going to the effort of song-writing you are just copying and pasting. Copying entire sequences again and again and again, and making variations of them, and going ‘this is the one with slightly more shaker in it now’. You make a copy of the whole thing, [edit the copy], and call it ‘louder shaker’” — Christiaan Ercolano

“If you’ve copied and pasted [a loop] you can adjust that particular one [without changing others].” — Troy Kelly

“You can always make a new pattern. . . , you copy an old pattern and alter it. — Andy Morton

In some cases, creating these variations was a significant focus of participants’ composition activity. Motivations for this included making material sound less repetitive, as well as to make individual copies fit their *temporal* context. This could lead to cases where producers would end up with sections where the majority of copies were individually edited for variation. This is illustrated in the following quote, which was also used to illustrate the requirements for this related aspect of temporal abstraction (see section 8.3.2):

“So your simple two bar turnaround has turned into one [block] with a cymbal at the front, one clean, one with a drop out for the special word [in the lyrics], and then another one with the fill. So what started off on day one as four [repetitions of] two bars, is now four different sequences. . . . Suddenly bars have to be thought of as a part, and that is a much slower [process]. . . . I tend to write the other way though [with distinct copies of sequences], because you know there will be more tweaks and more tweaks,

so they might as well be separate. Because as you go deeper you go ‘oh you know that third one, that has the drop out, it should actually go [*vocal drum sounds*]’ as well.” — Christiaan Ercolano

Emile De La Rey spent a great deal of time creating many individual variations in rhythm parts through copy, paste, and editing:

“See these drum parts, most of these look pretty similar but there are slight variations in all of them.” — Emile De La Rey

Another reason for creating independent copies of material was to ensure that undesirable changes didn’t pass on to related material:

“You’d make a copy and start attacking it just for safety reasons. When you’ve got the verse that you like you copy it before you do anything to it to make it your verse fill.” — Christiaan Ercolano

In a copying scheme the independence of repeated material prevents changes in one part of the developing composition from affecting related material. While this is often necessary, this also stops strongly related material from staying in sync:

“You can’t change a lot in the arrange [view] that actually changes the session [view] because there is no double ups. In here there is none of that.” — Troy Kelly

Tim Prebble described how a special sound replacer plug-in for Pro Tools attempted to address this by algorithmically scanning for repeated material, and permitting changes to be permeated through repetitions. However this was limited to simple sound replacement changes:

“If you had a region like a drum loop and you decided you wanted to replace the first kick-drum, you could apply it to the first one and then tell it to do it to every one.” — Tim Prebble

9.2.2 Referencing

Referencing schemes provide mechanisms for material edited in one place to change related material elsewhere. For the purposes of this discussion we will refer to these types of linked material as ‘references’. Some implementations use prototype based schemes [109] where a single master parent entity is mirrored in any number of other child entities. In general, only the parent entity can be edited, and changes are reflected in all children. Terms for such referencing entities include “ghost” and “alias”. Another approach to referencing treats each linked repetition as an equal peer to any other. Any one of these peers can be edited, causing the same change in all other repetitions as they are all visual duplicates of the same material, just appearing in different locations. This later approach often provides the foundation for ‘looping’ implementations, and pattern based sequencing:

“[Sequence duplication in the MPC] is not a copy and paste [like] in [Pro Tools] where it is an independent copy. When you use it again it is the same one.” — Christiaan Ercolano

The advantage of such referencing approaches is that many changes can be made globally:

“Well that’s what it does so well, if you are repeating this multiple triggering of any sound, . . . if you are getting a sample that is used all the way up through the song. If you get a snare and you tune it a bit better it is going to affect every track.” — Christiaan Ercolano

Referencing approaches were sometimes very problematic for participants as they found it difficult to determine quickly how various material was inter-linked due to **hidden dependancies**. This could result in increased **viscosity** as participants tried to avoid **errors** where changes to material had unintended consequences:

“It is visible that something is ghosted [i.e. referenced]. What you can’t tell [though] is what it is ghosted to. It just tells you that it is a ghost, but if you have a few different groups of ghosts you can’t tell them apart.”

Interviewer: Is that why you don’t use ghosting much?

“Probably. That is a [common] mistake with ghosting. I forget that it is ghosted to something further along the track. I make the mistake that it is ghosted to something else.” — Emile De La Rey

In regards to determining the consequences of a change, Christiaan Ercolano said “you just have to suck it and see.” He went on to describe how these changes can be very difficult to undo if problems are found further down the line.

In some systems, the only way see at a glance how material was related through referencing was to look at the material’s textual labels. Emile explained how this could be troublesome when references were created from former references, all of which would end up with the same label.

Paddy Free also had bad experiences when editing linked replicated material. Again, without a quick and obvious indication that material was linked and what it was linked to, it was all too easy for the producer to accidentally edit material throughout an arrangement:

“There is a very strange thing that took me a while to get my head around in Logic: that if I pack a bunch of parts into a folder, and then make option-drag copies of that folder, still what I edit inside is going to appear in the other [copy of the] folder as changes when I might not necessarily [want that]. But then they fixed that. [Originally you would make a copy of a folder] but the stuff inside it was still interdependent, and that was a head fuck, it was like ‘oh fuck why is that crapped out down that end of the song when I changed it down this end’.”

Interviewer: Do you ever want that to happen?

“Absolutely, totally. It is better to have that functionality, that’s what you need to happen from time to time.” — Paddy Free

Evaluation question

- Are individual repetitions editable, and do producers have control over how changes to one repetition affect others?

9.3 Variation

One significant disadvantage of the predominant but simplistic referencing schemes is that they make it impossible to create variation without resorting back to copy and paste instances. As a result, any representation of the relationships between variations is lost and manipulation of common aspects of the varied material becomes impossible. For example, Emile would often begin arrangements by using linked references, but over the development of the composition would convert these into concrete copies as variation inevitably needed to be added. The same was also true for Christiaan Ercolano and Paddy Free:

“Aliases serve to a point as no brainer things that truck on by themselves reliably. [However], there are points where you want to commit for the sake of committing, or you want to commit for the sake of change, at which point you turn your alias back to a real copy and edit the real copy so you can change it.” — Paddy Free

One way in which some participants did manage to create variations of references was to take advantage of parametising mechanisms. These allow producers to vary parameters applied to referenced material differently for each individual repetition. One such parameter is volume, which Andy Morton used to create variations without needing to create concrete copies:

“You’ve made a little beat, and then you go over to the arrange side and [are] just making copies. And then you start going ‘ok when it comes in I just want this little bell to be the first one’, and mute everything else.” — Andy Morton

In Atari Notator, Andy could vary material by changing parameters such as the pattern’s MIDI channel, quantise, transposition, velocity, compression, offset, and filtering of high and low notes. Many of these can be achieved in modern DAWs by varying mixer controls and running referenced material through MIDI or audio effects units that can be automated:

“[To repeat a change] you’d just drag it out. Or you’d just do it to the first one and then copy it through. If it was something like volume I’d just do it over here in the volume for that track. If it was an effect I’d probably just do the same. If I wanted it off for one of the loops I’d just automate it.” — Troy Kelly

Unfortunately, there are many things that cannot be varied by these parameterisation methods, which explains why our participants would often work with concrete copies.

As illustrated above, a common action for participants was creating variations of repeated material. Several producers expressed that creating fine-grained variations was important for their compositions, but it could be difficult to create these through editing:

“I’m not a big fan of grabbing an eight bar bass groove and just repeating it.” — Troy Kelly

“Doing fills and that sort of thing [requires a lot of effort, for example] if the music is loop based then deciding when and how you are going to vary it.” — Emile De La Rey

A more attractive solution to creating these variations was to use captured performance data:

“So I’ll get in a real drummer playing those grooves. It will sound so much better with a real drummer playing it. They will put in their little fills and it will just give the track life because at the moment it is very static, it just repeats over and over again.”

Interviewer: The variation would be painful for you to enter?

“Yeah fuck that, because you’ll be drawing in all these tiny little subtleties.” — Jeramiah Ross

Simon Rycroft also favoured replacing many electronically created parts with recordings from instrumentalists. Eric Vani and Paddy Free worked from varying source material, by recording musicians’ performances at an early stage:

“[We] record for a long time so that it is a human that is generating the results, so it is coming from your store of variation. . . . Unless you are setting up serious generative capabilities with random generators and things, . . . with a fellow human you are going to have that ineffable improvisational spark that could always come out . . . you’re not ever going to have that out of a computer.” — Paddy Free

Alex McLean explained how creating compositions by programming in Perl allowed him to easily create these types of variation through parameterisation and randomisation:

“I couldn’t really create it with a sequencer because I’d get frustrated with the lack of control. The repetition you have to go through to make something. . . in a sequencer. . . . One good thing I enjoy doing is breaking down a pattern which you can do very easily in code just by gradually adding random numbers to an otherwise straight forward pattern. . . . [Creating variation] is very easy with code because you can just reduce it to a few variables and write a piece of code that changes those variables over

time.... With sequencers I guess you're changing the output of your musical idea, whereas with code you are working with a musical idea itself, so you are changing the idea." — Alex McLean

Troy Kelly proposed a similar idea, but without the need for coding, which would help him generate variations without painstaking manual intervention:

"[Imagine] you've got a part within a piece of software like this where you can [rank loops] on a grade of one to ten Then you just hit randomise that folder [with a] random groove, and you make a song purely through this kind of prerequisite of classification. In [Propellerhead's] Reason you can create a random pattern and sometimes you get some really cool shit. A bit more craziness [would be cool]." — Troy Kelly

Evaluation questions

- Can references be maintained after derived material has been varied?
- Can varied repetitions captured through performance be represented as such?

9.4 History

Many of the most obvious issues for participants stemmed from managing the history of how material is varied during a project's development. Throughout the activity of creating a musical composition the project develops a rich and deep history and it can be extremely important that producers have access to this. History management includes issues typically addressed by undo and version control mechanisms. The extent to which systems provide mechanisms that facilitate history management for compositional abstractions can have significant effects on producers' activities. Illustrating the current problem with history management, Troy Kelly described four fifths of the process

as “archiving, project management and all of the mundane things”, rather than creating and editing musical works. Clinton Bradley’s many years of professional experience was vital for him in his role as a consultant, training others how to be producers. In his experience:

“You have to be organised or you’re going to lose stuff, bottom line. Musicians are notoriously disorganised. When you are in the throes of music and mad scientist stuff you are not [able to be organised] and you shouldn’t [have to be] when you are being creative... 99% of the time musicians don’t organise, they are more engaged in the creative side and don’t realise you have to back things up, you have to organise.” — Clinton Bradley

Discovering the nature of these organisational problems faced by producers in order to better support them is thus vitally important, and this is what we discuss now.

9.4.1 Versioning history

The highest level of organisation at which history management comes into play is at the arrangement level, where entire arrangements are captured in a ‘snapshot’. The dominant method of creating snapshots used by participants was the ‘save as’ command. This typically created independent project files, although in some systems underlying audio data was shared between them.

Emile captured snapshots of his project as he worked so that he could “...not feel scared about changing it drastically”. The same was true for Paddy Free:

“I’m savvy to [versioning] these days, so I’ll save a version of that... I just learnt to have that snapback to a moment of clarity.” — Paddy Free

In an observation session Eric Vani did the same in order to safely “change up arrangement options”, and Jeramiah Ross also captured snapshots in order to allow himself the security of keeping a version to come back to:

“I’ll save the Live session, so if something is not right I can go back to that and get it out again.” — Jeramiah Ross

Paddy Free found that creating versions of effect processing automation also led him to create snapshots, due to a failure of the provided abstraction mechanisms. As he explained, this was because it was not possible to capture and represent multiple takes of automation in Ableton Live or Apple Logic because they are “tied to a track”. A sub-optimal workaround was to take a new copy of the track:

“There is only one place in Logic’s setup where I can write bus two automation, only one place where I can edit it. I can’t have multiple versions of bus two automation. It is a sort of meta-data that sits at a different level to all the other [data]. It is [inconsistent]. If I want to do multiple takes I have to save multiple versions of the song... [or] you have to [render] the results of the bus two automation.” — Paddy Free

This tendency to capture multiple versions of a project can result in large numbers of snapshots being created, causing complex asset management issues. For example, with one typical track, Emile created thirty snapshots of a piece in the duration of a single week. In Emile’s case these snapshots would typically consist of seven or eight major versions, with intermediate snapshots providing a form of history so that smaller changes could be reverted at any time. Simon Rycroft described how this method also worked for him:

“In Pro Tools we ‘save as’ all the time. By the time you finish a track for an album you might get thirty to fifty versions of it, all in session files... just different arrangements, different effects, different bits and pieces. It is always good to go back in a few weeks and listen to how it has changed from that point. What ideas you dropped and what ideas you added. Sometimes I find that when I go back to a much earlier version it is really fresh and you’ll wonder what happened.” — Simon Rycroft

Crucially, these snapshots often took place when participants needed to create compositional abstractions that could only be approximated through the crude means of rendering. In these cases the producers were purging their project of production data, and wanted to capture a snapshot so they could access this information at a later point if necessary. When Eric Vani rendered multiple voice parts (voice abstraction) with all of their collective effects (processing abstraction) into a single long rendered audio block (temporal abstraction), he first performed a ‘save as’ operation to ensure he would have access to the underlying material if it was needed.

In order to deal with the resulting asset management problems created by manually managing multiple snapshots, participants needed to formalise how they managed their projects. Emile used last modified dates to keep tabs on when each version was created, although this could become confused when older versions were revised. He also used a system of naming for final masters, variations, remixes / large changes, and less structured idea project files he named ‘messes’. Managing all this could become very complicated. At one stage he laughed when demonstrating the multitude of snapshots created in a single project “because of the names, because I don’t know what the hell is going on.”

Emile also described how tracking and naming of a project’s history became even more complicated when multiple projects were merged. The participant felt it would be useful to be able to see how tracks evolved and related together:

“It would be useful to see the development, because of how randomly you were working [across different folders].” — Emile De La Rey

Moving between different audio applications was common, and observed in action with Paddy Free, Jeramiah Ross and Eric Vani. This created additional complexity for file management as each distinct program would create its own branch of project files. In one case, Jeramiah Ross had difficulty

locating the Ableton Live project file that contained source material for a piece he was working on in Acid. He cited this sort of problem as the reason he had to develop a formal organising strategy:

“I’ve actually begun naming songs in dates and data just so I know what the hell is going on.” — Jeramiah Ross

Troy Kelly also used dates and a careful organising strategy:

You soon realise you need to have a really good self made system, so I use the date. So a project at the moment we did two versions of the same album, one in English one in Te Reo. So I [remixed it with a different EQ] and just used the date. I can’t understand why a lot of software doesn’t do that, it just goes ‘version 1’, ‘version 2’. — Troy Kelly

Simon Rycroft found that managing all of these assets required him to develop a paper-based system to document how things were organised:

“I write things down, file management I guess. By the time you’ve got about fifteen bounces of every song and all the versions. We’ve got probably about a terabyte of hard-drive space filled with all sorts of palaver. It’s because we are too scared to throw anything out.... We’ll end up with up to fifty different session files for one track. If you don’t effectively label each one with notes you end up having to trawl through each and every file. It is time consuming.” — Simon Rycroft

One key reason for saving snapshots was to capture arrangement possibilities. However, saving each arrangement to a separate project file meant that edits that should be shared between different arrangements, such as a correction to a vocal in the chorus, or a change to the bass pattern, would not be carried through to other arrangements:

“I usually work with Thomas, and sometimes one of us will listen to a track and rearrange the thing entirely and save it as ‘Simon’s arrangement’ or whatever. We just create new sessions. It can be a hassle because once you have tried an arrangement we’ll then stick with that one for a while. [We then] add more elements and mix it, but when you go back and change to the original [arrangement] you have to re-edit the whole thing. That happens reasonably often. I can see if I had a function to revert to a different arrangement it could be handy to switch between arrangements in a single session.” — Simon Rycroft

Andy Morton spoke about the need to capture different arrangement versions while trying out new ideas. As with Simon Rycroft, different arrangements were often called for when collaborating with others:

“[When] people have different musical ideas [we work through it] by trying things out and listen to the person. If you’ve been putting it off, I’ll just do it, try it. You can then go ‘I don’t like it’ ... or you might go ‘I don’t like it but now I understand what you are talking about’, so go back to another idea and [ask them] ‘what do you think of this?’ ... Being able to go back to old versions comes in handy.” — Andy Morton

Clinton Bradley managed to keep multiple arrangements in a single project file in Logic, with each inside its own track folder. However, again there could be difficulties ensuring changes in one arrangement were shared with others when appropriate.

Evaluation questions

- Can producers capture a snapshot version of any block of material that they can recall later?

- Can different versions of all types of data be maintained inside a single project file?
- Can producers work with the same data across multiple applications?
- Can changes be propagated across multiple arrangements when required?

9.4.2 Take management

Another component of history tracking is take management. In multitrack recording, a ‘take’ is a captured recording of a specific musical performance. Multiple performances of the same material can be recorded as separate takes. A core concept for participants was capturing ‘takes’ of a performance which could be compared, and composited into a final version. As the number of captured takes naturally balloons over the course of a project, mechanisms for managing these can become vitally important:

“If you are doing a twenty-four track band, and every time you record [a take you create another twenty-four tracks] it is a big hassle to manage those tracks without dropping them into folders.” — Clinton Bradley

Some systems provide mechanisms to wrap multiple takes into a single entity:

“Audio events can be displayed and edited directly in the project window, but you can also work with audio parts containing several events. This is useful if you have a number of events which you want to treat as one unit in the project.” — Cubase manual [128, p.26]

One problem for some participants was that the mechanisms provided for managing multiple takes did not adequately support multiple takes of automation data, as described in section 9.4.1. The way in which automation

data is conceptually overlaid onto track data can make it difficult to maintain multiple alternative versions of any one parameter automation. As discussed in section 6.5, participants often used effects parameter manipulation to create compositional material — making this data a form of performance data on the same conceptual level as note event data, not simply a form of meta-data created in a final discrete mixing action. With parameter manipulation a form of performance, it was evident that participants needed the ability to capture multiple takes of these performances.

Paddy Free spoke to this issue, describing how it was a particular problem in his work:

“It’s not possible to make multiple takes of automation in Live or Logic because they are ‘tied to a track’ and you’d have to take a new copy of the track.” — Paddy Free

Under the standard multitrack resource instance model, if each automation take was captured on its own track, each of these tracks would need to have their own instance of the effect unit (with the added CPU overhead that implies). The only alternative would be to create an additional effects channel for those takes adding undue complexity to the project in the process.

Evaluation question

- Is there support for managing multiple takes as a single unit, including takes of automation?

9.4.3 Material history

History can also be very important at a finer granularity below the arrangement and take level. At this lower-level producers are concerned with the history developed through operations such as chopping up, moving, and splicing smaller scaled material; making changes to processing of sections of material;

and making changes to automation. Using the large-grained arrangement level is far from ideal for capturing this form of history. The principal reason for this is that it can make it difficult to find the history of a particular chunk and revert its material back to its prior state without also reverting the state of surrounding material:

“You have to go back to an old version, and how about the work you’ve done in the mean time on other things?” — Andy Morton

Just one example of this was small mistakes (i.e. due to **error proneness**) such as inadvertently moving material, but this also extended to all other forms of editing:

“It’s real easy to grab something, knock it slightly, and depending on how well you know it, it’s hard to take it back to where it was. If you don’t notice it you can’t undo it.” — Andy Morton

Obviously, finding the origin of specific material becomes increasingly difficult as the depth of a project’s history increases, and if material is sourced from multiple projects.

In order to capture this fine-grained history in the multi-track model, participants such as Paddy Free were observed taking a copy of local material’s current state and preserving it on a temporary muted track. If changes turned out to be undesirable the known working version of that material could be reinstated or reused at a later point. Clinton Bradley had developed techniques to store this material manually in related folder tracks so he could access it at later points:

“The good thing about the folders is as I’m mashing stuff together I keep the originals in a folder muted.” — Clinton Bradley

Tim Prebble also revealed how he would keep these small backups around for a significant duration in case he needed to come back to it at a later point:

“I have workarounds. If I wanted to pitch that [noise] down I would tend to stash a copy of it and mute it so that I could always go back to the original, which is a slightly clumsy workaround. It would be better if you had a history.” — Tim Prebble

It was clear here that the participant was describing how it would be useful to have a local history for the abstraction he was editing (in this case a ‘noise’ object) rather than a global undo history for the whole project. Unfortunately, keeping ‘stashed’ copies to capture materials’ history had its own obvious problems. In particular, without more explicit support for capturing this history it could be difficult to keep historical material associated with derived material when a piece is undergoing rearrangement. Additionally, the complexity and confusion created by all of this extra unmanaged material in the project would be undesirable. Emile mentioned how he would “always keep a [separate project] copy with all of the raw takes in it”, and remove them from current working versions. His reason for keeping these separate was “tidiness”, or in other words, to reduce complexity.

Another reason that participants kept track of the history for material was so that they could fine-tune or recreate the same process at a later point. Tim Prebble described how developing new sounds could be a complex process worthy of documentation:

“He has really clever databasing so if he was creating a T-Rex vocal ... he will document that whole process [of how he created the sound] so it is reproducible.” — Tim Prebble

To conclude our discussion here we turn to Paddy Free’s vision of how a richer fine-grained material versioning system could work for him:

“... things are dependent on [other] things in their creation. Saying this was built on that, or this was an adaptation of that or a version of that, that would be great. That’s like not having to save multiple versions of a song. Anything is undoable and changeable at any point.” — Paddy Free

9.4.4 Dependencies

As material is reused, both as shared underlying material in multiple snapshots of a single project, and across distinct projects, then complex inter-dependencies (often **Hidden dependencies**) are created. Tracking these dependencies through the file system had become a major headache for some participants. For example, Emile was forced to write a script that would scan his Cubase project files to determine which projects use any given audio file on his file-system. This demonstrated the problem with forcing producers to use just standard file-system tools to manage project assets, as they don't provide access to the necessary meta-data required for dependency tracking. Simon Rycroft, Troy Kelly and Clinton Bradley all encountered the same issue with meta-data and dependencies:

“I guess one of the most difficult things to deal with my head is managing files. Managing all these different bits that make sense when you put them through a certain interface.” — Simon Rycroft

“My general rule is once you have multiple projects [using] one bag of audio files is don't delete anything, you just can't afford to, and it is a pain in the ass. . . . I'll tend to save a new copy of the project that includes all the audio files.” — Troy Kelly

“Here is the deal: the only really safe way is to make copies and start again. It is not a good idea to use the same material [in more than one project]. You could use the same source files, but it is asking for trouble. I've seen this many times, I've seen pros delete all the drums on an album by mistake. . . . For a major change you want to do a 'save as', but save it as a project so it makes copies of everything. You don't want to refer to all that other stuff. If I am going to make major changes I'll generally save it as a whole new name.” — Clinton Bradley

Clearly these producers have learnt the hard way that they must duplicate all assets shared between projects. This is due to the failure of software systems in supporting the management of interrelationships between project files. With the vast quantities of hard-drive space used by audio projects it is easy to imagine the cost. Also, every-time the producer performs a ‘save as’ they break the links to the rich material history in past project files which we have already described.

Evaluation question

- Does the system allow producers to easily track dependencies between material and the various projects it is used in?

9.5 Library management

A key component of producers’ music production activity is creating, managing, and accessing a personal repertoire of musical content in a library. This can include short audio samples, loopable phrases, longer recorded material, virtual instruments, effects units, gesture phrases (typically MIDI), all of their related presets, and potentially all of the associated abstractions we have described thus far.

9.5.1 Archiving previous work

An important component of these personal libraries is a record of previously created material that can be pulled into new projects. In some cases producers will want to start with a relatively blank slate:

“I don’t always want to refer to old projects. It is sometimes the most inspiring to have a blank page.” — Emile De La Rey

However, even in these cases producers will often reuse smaller components, such as instrument sounds and effect presets. More significant library

mining can arise when producers have created a large number of concepts which they may or may not develop further:

“[My collaborator] was talking about [creating] ten or twenty beats and I only liked three of them. I said that is usually the ratio, twenty to two. You have to do twenty to find the two that are good.” — Christiaan Ercolano

“Whether it is for good or for bad, I am probably making ten times as many [grooves] per month. Constantly you think of something new, and you try it, it works, you hang onto it, if it doesn’t work you throw it away. Just churning through things much quicker.” — Andy Turley

This producer explained how even some of the material that wasn’t working in its original context could sometimes be successfully used later in a new context.

Paddy Free described the initial stages of developing a new piece would often involve pulling a lot of material from his personal library. A subset of these concepts from producers’ libraries, sometimes in combination, could become the basis for entire compositions:

“Quite often I’ll just boot up [Ableton] Live and go through my sketch archive and get audio files that I’ve recorded in for various things.... I’ve written a whole track out of just scribbles from other things that I’ve never finished. Lately I’ve been doing that live.” — Jeramiah Ross

One potential key to this will be allowing the producer to access the rich context in which material is created:

“Meta-data for sounds is going to become critical as the amount of data you are accessing becomes larger.” — Tim Prebble

Evaluation question

- Can the producer archive and reuse all types of material abstractions from past projects, while retaining links to their previous contexts?

9.5.2 Locating material

A library is only useful if material can be found easily. Equally, being able to locate material inside a project is crucial to producers' efficiency. If producers come back to a project after some time, or as Paddy explained, if it has been created by a collaborator, locating material can be even more difficult. In interviews and observations, producers demonstrated various techniques to improve their ability to locate material, which we examine now.

One common method utilised by producers was labelling, which is a form of **secondary notation**. As described in chapter 8, musicians and composers have long referred to blocks of material with terms such as “Verse One”, “Chorus” and “Bridge”. This form of naming also persists at lower levels of the temporal block hierarchy too. In observation sessions producers named the various abstractions in their projects, from effect processing units, chains of these units, tracks (voices), and audio and MIDI blocks, all to make them more memorable and locatable. Systems that don't provide labelling facilities can cause the producer to make **errors**, even with a relatively small number of objects to differentiate:

“In Redrum, which is really simple, it is fairly rudimentary but you can paint your simple drum patterns in it. It just gives you a real easy structure, and because you can switch between the banks so quickly, you can go ‘this is pattern A, B, C, and D’... If I could label those it would probably help me an awful lot, because I [make mistakes when I] switch between them.” —
Andy Turley

Labelling can be vital, and if producers leave material unlabelled for too long it can become very difficult to remember what is going on later:

“When your first session is over and you haven’t actually had time to label you’re fucked. Labelling will make all the difference to a big job.” — Christiaan Ercolano

Paddy Free was observed labelling material in observation sessions and made it clear how important this was to his projects’ organisation:

“Often part of the whole project process for me is after you’ve been playing a whole bunch of music and you’ve gone ‘where the hell is this going?’ is to go through and do labelling and naming. It is a really necessary process.” — Paddy Free

This producer would go as far as interrupting collaboration with others in order to take time out to label material in a current project before things became too confused:

“Give me ten minutes guys, I’m just going to do some labelling and house keeping.” — Paddy Free

The importance of labelling also extended to project files and resources, and these names often needed to be kept **consistent** with one another:

“You need a working name very very quickly. Especially if you have like a computer and a MPC locked up everything has to have the same name. It is critical that folders are made with that name and sequences and samples with that name, and maybe the sample you took is bounced into the computer just as a bulk recording just in case you can’t find the record again.” — Christiaan Ercolano

This **consistency** was difficult to maintain when projects were renamed:

“Every song ended up with a different name from what it started. It would be ‘Kenny dope loop’. Then it would be called ‘Dope

number 2', Once [the vocalist added the lyrics] deeper love it was called 'Deeper love' so 'dope beat' never made any more sense. When you'd come back you'd have to make a chart of what they ended up as, and what they are now." — Christiaan Ercolano

Emile indicated how this could also occur when multiple projects were merged, with the result that "the naming starts getting confusing".

Labelling mechanisms were not a panacea for locating material. For example, as previously quoted, Emile looked through a number of project names he had carefully created and laughed, saying he was confused "because of the names, because I don't know what the hell is going on".

Simon Rycroft expressed a deeper dissatisfaction with systems that forced him to rely on labelling over other means:

"Something as basic as importing an audio file [forces me to think], because I have to look at a list. All of a sudden I'm looking at a list of words, and it is really difficult for me to actually go back into that way of working where I'm not dealing with a graph, and moving stuff around. . . . [It is the] same for looking at lists of plugins. And it really gets to me, I have to get out of my zone, and it destroys my train of consciousness to have to actually go through the filing system of my computer and wade through. I find that really removes me from the process." — Simon Rycroft

Forcing the producer to remember the name of an object is less than ideal if they could rely on the more salient features of the objects, such as their spacial-visual characteristics. Emile described how spacial-temporal location provided a powerful means to locate material:

"Usually I know what is going on in my track. I'll just go to that point in time. [It is] fairly easy because it is time based. If you know where it occurs then you go [there] and look at the track that it is in." — Emile De La Rey

Christiaan Ercolano emphasised the importance of good labels in time constrained contexts such as live performance:

“The naming thing is big. . . . [Labels] are bloody everywhere. . . . When we play live, all the song sequence names are where you are in the song. . . . Without them there is no hope.” — Christiaan Ercolano

Having rapid access to this library material in a studio has one meaning, but the requirements on a library for use in a live performance situation are even more demanding. Paddy Free explained how Ableton Live has come some of the way towards those requirements:

“[Ableton] Live can time compress or expand any tightly cut loop off your hard-drive in real-time. That is the cool thing, so immediately you can audition anything against anything [else] which would have been quite a complex process before that: looping up every sound in every song from our back catalogue so now we can pull on absolutely everything we have done.” — Paddy Free

However the ultimate goal is yet to be reached:

“Make an interface that is as spontaneous as saying ‘you remember that part you played 6 months ago? [Play] that.’ . . . If an iPod supposedly delivers you your music on demand, non-linear, when you want it, mobile, what would be amazing is to have that iPod recall but still you can deconstruct from within those mixes completely. You can open out the full project down to its component wave forms again. . . . If you could get to the equivalent of saying to the band, ‘let’s do that number from [any point in the song] Ideally we’d like to have every song we’ve ever done all loaded up into our show-load. We can’t because of the limitations [of having them all loaded in Live].” — Paddy Free

Evaluation questions

- Is there adequate support for rapid naming, searching, and name-versioning of all types of data?
- Can material be located spatially and visually rather than solely by name?

9.5.3 Local libraries and working space

It was interesting to observe how participants not only used a global library, but also organised a library of material inside individual projects. Our participants repurposed voice and temporal abstraction mechanisms in order to store this library content locally in a manipulatable and viewable form. Assembling this local library involved both bringing material from external sources (as detailed above) as well as creating material directly within the project. In chapter 5 we introduced actions to generate material, which provided an example of this library building in Christiaan Ercolano's work:

“You are fucking round. That is a huge chunk, that is the first day. You don't do any song writing the day you are inputting samples and trimming them, and getting ready for song writing.”
— Christiaan Ercolano

For one piece Paddy Free captured multiple takes from a session guitarist, which provided a library of raw material for that piece. Each take had a different style so he could use them for different musical purposes.

“[This is] only the first process of assembling the ammunition.”
— Paddy Free

Clinton Bradley argued for the importance of building a large quantity of material to draw from in the production process:

“I find that the more that I keep the better result I get at the end because I’ve got far more [material to choose from]. Particularly, guitars will make a lot of [small mistakes] and that quite often can become interesting audio fodder for a key moment in the track.”

— Clinton Bradley

There were four ways in which participants organised these local libraries. The first, as observed in a composition session with Eric Vani, was as a selection of generic audio clips on the file-system that could be accessed by multiple audio applications being used in that specific project. In an observation session, Eric captured at least half an hour of rhythmic drumming. The material was then systematically searched, promising material identified, and each block saved in individual files on disk. Some of this material was then imported into a different DAW where it was used as the basis for a new composition. Unfortunately, exporting each block to the file-system did not preserve their history and context, as each was essentially severed from its original place in the captured audio. This file-system approach also provided only limited means for organising and marking up the audio.

In linear multi-track systems participants used temporal space before and after the canonical arrangement to store local library material. This was the second way in which participants were observed organising a local library. This requires tracks to be created to contain this material with appropriate effects processing (or lack thereof), and for them to remain in place and unmodified for the duration of the project’s development, independent of the multitude of other tracks being used actively for the arrangement.

“Often editors, and I do it myself, go an hour down the track and have a library of all the stuff they are using. They will tend to jump down there, grab some stuff, and jump back to where they cut it. The same way where in excel you can split a [sheet] in half and jump between two non-linear bits. . . . It depends on the kind of project as to whether you assemble a perfect library or it is just a space for cutting things up.” — Tim Prebble

“As I create the song, if I go ‘oh that is a good loop, that is going to be part of the song’ I’ll copy the loop and put it at the beginning of the track so I have a library of all my loops. [For example] if we drop a guitar line in and we are like ‘yeah that is it nailed’, I’ll copy it and have a bank down the left hand side. If I’m finished the song and [I think] I’ll probably want to use this live, I’ll copy it, and flip [to the session view] and use it live.” — Troy Kelly

The use of the library in a live interactive environment is also important. In systems such as Ableton Live, as this local library is developed it can be interactively shaped and arranged in real-time:

“Live allows the conditions for spontaneity. This is the ammo I might draw from, and you don’t have to play in a set way.” — Paddy Free

Using space before and after the main arrangement allows the space for blocks of material with substantial temporal durations:

“The advantage of having it down the end is you can access the whole.” — Tim Prebble

Here, producers can even have the space to build mini arrangements, creating larger temporal blocks for the local library they are building. In this way, the local library also acts as a scratch pad for trying out and manipulating compositional ideas without compromising the existing arrangement. This served Troy, who recorded sections of material from the interactive session view in Ableton Live beyond the end of a piece:

“I’ve just gone down here [later in the song] and recorded from the [session view].” — Troy Kelly

The third way in which several participants were observed creating local libraries allowed for context, although would be less well suited to storing material with longer durations. Material was placed on disabled tracks so that it was temporally located near related material.

“You can disable tracks but I can still chuck things on [it]. That is kind of in context where down the end it is a resource.” — Tim Prebble

The classic example of contextual library material was witnessed when participants captured multiple performance takes. A final version could be stitched together from the multiple takes, or as in the Paddy Free example given earlier, blocks of material from these takes could be used in locations related to the original temporal location.

The final way in which participants were observed creating local libraries was building material in several different project files. This could be problematic when systems don’t allow producers to work on several project files simultaneously:

“The fact that you can only have one session open makes that more difficult in that if I go away and make some weird sounds in another session I then need to close that and open my real session to put it in context. It would be much easier if I could have two sessions open. . . . Sometimes they are not time based. For the master [project file] it is critical how it is structured, where as the others can be as random as you like.” — Tim Prebble

Evaluation questions

- Can the producer access the local library content from multiple project files, including contextual library material and larger mini-arrangement style material?

- Can the producer create local library content without adding complexity to the composition?

9.6 Summary

This chapter revealed how producers' activities were influenced by *reuse and versioning* abstractions — or typically lack thereof.

We saw difficulties in the coarse-grained 'either-or' approach, in which repeated material was strictly either discrete copies *or* linked references. This division forced the loss of all semantic ties between repeated but varied material in the digital representations that our participants were developing. Also, versioning and snapshot abstractions were practically non-existent in participants' software tools. Our participants had been forced through bitter experience to work around this by reshaping their activities to manage this *manually* at a large cost of increased complexity. Participants experience had also led them to modify their work process due to the lack of dependency management where material was reused across multiple project files. We also found interesting difficulties in the work of producers where systems failed to provide means of incorporating local or foreign library content into a project without adding additional complexity to the global channel structure of a piece.

Chapter 10

Evaluation and Discussion

10.1 Introduction

In chapters 5 through 9 we presented our analysis of the activity of computer mediated music production and the abstraction mechanisms that play such an important role in producers' work. These resulted in a number of evaluation questions for examining music production systems in light of our findings. In this chapter, we apply these evaluation questions to two software systems, and then examine a number of implications for future design of such systems. We conclude by examining the ways in which our findings relate to similar domains.

10.2 Evaluation of systems

In the early stages of this research we published an an evaluation of Ableton Live and Propellerheads Reason based on the user-interface metaphors employed [57]. Now that we have completed a detailed account of abstractions and music production based on our field research, we can use this to perform a detailed evaluation of such systems grounded in these new empirical findings.

In order to evaluate two prominent music production systems we began

with our evaluation questions in appendix C. We applied each evaluation question to the software system in turn, and summarised the most interesting and problematic results, which we present below. Following the evaluation sections below are tables demonstrating how we systematically applied each evaluation question to the systems under examination. We found this to be a relatively quick and thorough way to assess how the abstractions supported in the systems would interact with the music abstraction and activity principles we have discovered, and should be similarly accessible to practitioners working in this domain. In the evaluations below, the relevant evaluation question is cited by its unique code. For example (P.1) signifies the first processing abstraction question “Can producers avoid manually working around CPU limitations?” as listed in the appendix.

10.2.1 Apple Logic 7

Apple Logic 7 (shown in figure 10.1) offers some of the more advanced abstraction mechanisms of any linear sequencer, and as such provides a good benchmark for evaluation.

Figure 10.1: Apple Logic.

Folder tracks

Apple Logic's 'folder tracks' provide the facility for creating abstractions over processed content (P.6), multiple voices (V.4), and temporal groups (T.1). In many ways, the resulting folders act much like concrete audio regions on the timeline. In particular, this allows many of the editing and arrangement operations that are typically utilised when working with concrete audio.

As useful as these folder tracks are, there are a number of ways in which they fail to meet the abstraction requirements uncovered by our research findings. The principle limitation of folder tracks is that they cannot always be used interchangeably with rendered mix-down versions of the material they represent (P.6). Unlike rendered audio, they do not support detailed editing tools that search or manipulate their aggregate audio, because they are processed in real-time from their component parts. For the same reason, cuts and splices to track folders containing processed material do not preserve the state of effects at cut points, resulting in dramatic changes to the audible result (P.2). As we now know from this research, this is unacceptable to those using effected results as first class compositional material. Additionally, track folder abstractions fail to provide the crucial waveform visualisations that many producers get so much utility from (V.5). Finally, effects cannot be directly applied to folder tracks, requiring re-routing to achieve desired results, and adding to the complexity problems we have seen in our studies.

While folder tracks do support several of the key forms of abstraction identified by this research, it is unfortunate that their somewhat coarse-grained nature limits opportunities to utilise them. We have seen that producers need to compare the contents of abstractions with external material on a common timeline (V.6). Many of the abstraction requirements we have discussed do not *require* information hiding. For example, the ability to rearrange a piece through the manipulation of larger temporal structures should not necessitate the material be concealed inside some other object. It is unfortunate then, that to gain any benefit from folder track abstractions, Logic forces the user to hide material in a separate timeline. If material had been allowed to

remain inline it might better meet users' needs, while simultaneously allowing for abstract operations. This observation was borne out in producers' own accounts. Our final critique of Logic's folder tracks is that they fail to address the problem of option dilemma (G.1), as there is no mechanism to mark off material as complete, without resorting to rendering the material to raw audio.

Effects processing

When Logic users create compositions employing large quantities of synthesis and effects processing, they must employ workarounds to resolve CPU limitations (P.1). Freezing and unfreezing operations cause long delays that interrupt creativity and prevent most editing operations, which we found to be unacceptable to some producers. Rendering material is the only practical alternative, which as well as being similarly slow, disconnects material from its rich history and prevents changes to underlying source data (P.6).

In Logic, arbitrary effects cannot be associated with individual blocks of material (P.4). Instead, effects must be associated with an entire track, resulting in additional confusion and complexity. When an effect is shared among a number of tracks, Logic provides no simple means to allow the parameters to temporarily diverge (P.4). Additionally, the real-time processing model behind the system inherently creates unpredictable results, determined by the arbitrary start point of play back and initialisation state of effects units (P.11).

Unfortunately, transient musical events, such as one-off phrases require the addition of a whole new track to the entire duration of timeline, or grouping the material with otherwise unrelated elements (V.2).

Aliases and versioning

While Logic provides a system of linked copies (aliases in their terminology), these links cannot be maintained if even small changes are made to an alias (R.2). As a result, modified aliases become true copies and changes can no

longer be made across the set. It can also be difficult to determine which particular aliases will be affected by a change to a referenced region (R.1).

We have described the importance of capturing various types of versioning snapshots, and note that some Logic users still rely on manual ‘save as’ and file system management of fragmented project files (R.5). Also, changes to one ‘save as’ arrangement option cannot be propagated to others when desired (R.7). We discovered that inside project files, it is all too easy to lose the creation history of sub-mixed and comped material, with producers being forced to ‘stash’ muted source material inside folder tracks and manually organise and keep them together (R.4).

Non linear performance

Logic also fails to provide a seamless and integrated mechanism for eyes-free interactive arrangement (T.7), preventing producers from treating arrangement as another form of performance expression without time consuming preparation of material.

Apple Logic evaluation table

Logic: General	
G.1: Are there facilities to help producers deal with option dilemma, allowing them to mark off decisions and move forward?	Only by rendering.
Logic: Processing	
P.1: Can producers avoid manually working around CPU limitations?	Only by freezing which prevents many forms of editing.
P.2: Can producers easily change materials' voice and location in time without losing processing context and how it will sound?	Only by rendering.
P.3: Can automation parameters be associated with material or points in time as required?	Yes.
P.4: Can short term or diverging effect processing be created without adding complexity or CPU overhead?	No.
P.5: Do the object editing facilities allow for tail effects?	No.
P.6: Can results of effected processing be edited and viewed as first class audio without regard for underlying effects processing, while allowing underlying processing to be viewed and modified even after the results have been edited?	No. Only limited editing through folder tracks with no waveform visualisation.
P.7: Can processing complexity be reduced for use in a live setting?	To a limited extent through track folders and rendering.
P.8: Is it possible to hide edits?	No.
P.9: Can looking ahead on the timeline be disabled?	No.
P.10: Is it possible to create feedback loops and manipulate them interactively?	Yes.
P.11: Is it possible to have predictable results of effect processing?	Only with rendering or freezing.
Logic: Voice	
V.1: Does the system provide voice abstractions to capture both the concepts of instrument parts and perceived parts?	Only for aggregated voices as folder tracks.

V.2: Does the system provide an abstraction for dealing with transient musical events?	No.
V.3: Does the system allow producers to make use of visual spatial layout to organise material?	Only on tracks.
V.4: Does the system provide voice abstraction facilities to reduce voice counts and the resulting overwhelming complexity?	Yes.
V.5: Does the system allow producers to view, edit and arrange multiple voices as if they are a cohesive whole?	Only to a limited extent through folder tracks or rendering.
V.6: Does the system provide access to individual voices inside voice abstractions where multiple voices are amalgamated, and allow them to be compared with other voices from outside of the abstraction?	The contents of folder tracks can be viewed, but they are on their separate timeline divorced from other voices.
Logic: Temporal	
T.1: Can material be represented in a hierarchy of temporal blocks up to the set level, allowing information hiding when desired?	Yes.
T.2: Can the producer work with multiple levels of the temporal hierarchy simultaneously?	Only through manually managed multiple views.
T.3: Can multiple contradictory temporal abstractions be represented?	No.
T.4: Can the producer build and view a piece's entire linear structure?	Yes.
T.5: Is it possible to specify variations in repeated material to ensure diversity and to ensure coinciding parts mesh?	No. Only by breaking alias references.
T.6: Can neighbouring and global context of repeated material determine the details of how it is performed?	As above.
T.7: Can arrangement decisions be radically changed throughout the entire production process, including in a live eyes-free improvisational context?	No.
T.8: Is placement of material against a metrical abstraction mediated by a system to take lead-ins into account?	Yes, through the use of anchor points.

T.9: When material is captured without correct metrical abstraction information, can it be added after the fact?	Yes.
T.10: Do the metrical abstractions provide facilities to change tempo and time-signatures at any time?	Yes.
T.11: Are producers able to manipulate material in terms of bars, beats, and sub-beats through provision of temporal editing and visualisation over metrical abstractions?	Yes.
T.12: Are metrical structure abstractions provided, allowing producers to learn common parameters for expressive timing?	No.
T.13: Can material be captured and refined through performance gesture?	Material cannot be refined through performance gesture.
Logic: Reuse and Versioning	
R.1: Are individual repetitions editable, and do producers have control over how changes to one repetition affect others?	Alias dependancies are not well visualised, and individual repetitions are not editable.
R.2: Can references be maintained after derived material has been varied?	No.
R.3: Can varied repetitions captured through performance be represented as such?	There is no provision for creating relationships between repeated material that has been recorded from an external source.
R.4: Can producers capture a snapshot version of any block of material that they can recall later?	No, not without manually copying it and placing it elsewhere on the timeline.
R.5: Can different versions of all types of data be maintained inside a single project file?	No.
R.6: Can producers work with the same data across multiple applications?	Only through rendering and export.
R.7: Can changes be propagated across multiple arrangements when required?	No.
R.8: Is there support for managing multiple takes as a single unit, including takes of automation?	Yes, but not for automation.
R.9: Does the system allow producers to easily track dependancies between material and the various projects it is used in?	No.

R.10: Can the producer archive and reuse all types of material abstractions from past projects, while retaining links to their previous contexts?	No.
R.11: Can the producer access the local library content from multiple project files, including contextual library material and larger mini-arrangement style material?	Yes, by opening them in a new project.
R.12: Can the producer create local library content without adding complexity to the composition?	Adding library content complicates the project's global channel structure.
R.13: Is there adequate support for rapid naming, searching, and name-versioning of all types of data?	Many types of data can be named, but name changes are not retained.
R.14: Can material be located spatially and visually rather than solely by name?	Yes, but only within the restrictions of the single timeline.

10.2.2 Ableton Live 6

Ableton Live (shown in figure 10.2) offers several novel abstraction mechanisms, which were well appreciated by a number of our participants. Therefore it is appropriate to give it special attention through an evaluation here, based on what we have learnt from this field work. While there are many positive aspects to the design of Ableton Live 6, for brevity we focus here on the limitations in Live's support for abstractions, and how they affected producers' work.

Freezing and rendering effect processing

As shown earlier in figure 3.19, Ableton Live 6 permits an extended variety of editing operations to be carried out on frozen tracks (P.6), allowing users to remove performance overheads while still retaining some editing flexibility. For example, frozen material, including audio tails (P.5) resulting from effect processing, can be split and rearranged. However, any editing of tail effects, and the results of edited effects are lost when the user inevitably needs to

Figure 10.2: Ableton Live.

unfreeze the track to add or fine tune effects. As a consequence, editing the results of effected audio is only practicable if the user is prepared to give up the option of modifying underlying processing (P.6). Additionally, the results of frozen effects are not visualised as a waveform, hindering precise editing. For these reasons, producers are forced to use the consolidate and flatten commands, which have the side-effect of removing all links to the production history of material.

Multitrack-mixing: Channels and tracks

In implicit recognition of the importance of processing abstraction, Ableton Live ensures that when blocks of material are imported from other projects, any required effects units are automatically imported along with them (P.2). In this way, effects are recognised as an integral *part of* material. For example, importing a MIDI drum pattern would result in its associated drum machine, samples, and compression unit being brought into the new context

in which it is being used. These instruments and effect units are automatically loaded into the track on which the material has been added. However, in a clear demonstration of the breakdown of the multitrack-mixing metaphor, any further blocks of drum material imported on that track *cannot* add their required effect processing units as they would conflict with those already in place (R.12). Producers hoping to organise material by perceived instrument parts as a form of *voice* abstraction (with, for example ‘drum grooves’ all being placed on the same track) are foiled by tracks’ role as the principle means of process abstraction (V.1).

This is related to the difficulties we established with multitrack-mixing systems, where short term effects required for a limited block of material must be associated globally to an entire track (adding global complexity) and remain active for the duration of the piece (adding to performance overhead) (P.4). As with Logic and other multitrack-mixing systems, neither is there natural support for parameters of shared effects temporarily diverging, nor transient musical events. Inline edits cannot be hidden without destructive editing (P.8) and consolidation (P.6), and there is no way for producers to easily mark off edits or other changes as complete without making them permanent (G.1). Also, as with other systems, real-time processing can result in unpredictable audio results depending on the start point of the playhead and effect unit state (P.11).

Multiple voices

One of the most significant limitations of Ableton Live’s abstraction mechanisms is the lack of voice aggregation abstractions (V.4). Unlike Apple Logic, there is no way to fold multiple voices (AKA tracks) into a single entity for combined editing, waveform visualisation, and complexity reduction (V.5). This can result in rapid track count expansion and a dependence on significant zooming and scrolling. This lack of voice abstraction also has the repercussion of there being no natural way to group and hide multiple takes (R.8), including automation takes (R.5).

Session view versus arrangement view

In the linear arrangement view, temporal hierarchy is limited to just a single level of blocks, with a second orthogonal level of timeline locators, making instant selection and manipulation of high level blocks (verse, chorus etc.) impossible (T.1). The non-linear session view allows clips across a number of tracks to be organised into ‘scenes’ for triggering, although there is no way to fold scenes into higher level scenes, or other recursive hierarchical organisations. As a consequence there is no support for high level temporal groupings for set organisation, and consequently each piece must be located in its own file to be loaded individually, or manually and painstakingly merged into what Paddy Free termed a “megalosal live set” (see figure 8.4).

Ableton Live echoes the classic dichotomy between linear and non-linear sequencers by essentially implementing two data-compatible, but largely independent sequencers in the one system: the session view (non-linear), and the arrangement view (linear). Each of these sequencers inherits the typical limitations of its class. For example, the linear sequencer is awkward for improvisational and composition actions, and commits producers to particular arrangement possibilities (T.7). On the other hand the non-linear sequencer makes it difficult to establish higher level temporal structure (T.4), develop and link appropriate transitions between musical sections (T.6), or deal with lead-ins occurring before the first beat of the bar (T.8). While the ability to move rapidly between the two sequencing environments is a great step forward, in practice, material in one view is not linked to the other, and limits the effectiveness of this approach. Consequently, producers are required to manually move material from one domain to the other. In Ableton Live there can be substantial loss of *temporal* information (T.1) (T.4) when moving between these two disjoint paradigms. For example, material created in non-linear format will lose its triggerable structure when moved to the linear timeline, all of which must be rebuilt if it is to be used again later in the non-linear format. The process of moving material into the multitrack-mixing models’ linear representation results in a degree of commitment to

a single linear arrangement, making it awkward to explore multiple varying arrangements. In this way any memory of other options of ‘what can come next’ is lost. As a result of the division between material in the non-linear and linear sequencers, changes to material in one format are not reflected in the other (R.7). Conversely, when moving from a linear to non-linear representation, producers are required to forgo the larger linear structures they have composed, breaking material at an arbitrary granularity and losing temporal hierarchy (T.1). For example, larger AABA structures could be triggerable as both a unit in of themselves, or as four individual blocks; but non-linear representations will force a choice of representation between the two. Additionally, as we have seen, specially crafted segues (the transitions from one part to another) which are so crucial to the smooth flow of arrangement are not easily represented in Ableton Live’s non-linear form (T.8), and are difficult to retain in the movement back and forth between linear to non-linear environments.

Reuse and libraries

A disconnection between replicated material is evident throughout the system (R.2). There is no mechanism to edit multiple repetitions simultaneously (other than that facilitated by ‘looping’), even within the same sequencer view (R.1). Alternative arrangements and editing snapshots are best served by the primitive ‘save as’ feature, with each located in detached disjoint files (R.4). As we have seen, this can result in confusing file management issues and dependancies (R.9).

The ‘browser’ allows limited organisation of library material (R.11), but its tree based file-system model leaves little room for creating and manipulating library content without merging it into the channel structure of the current session (R.12). Additionally, there is no support for tracking the naming history of material in libraries or current projects (R.13).

Ableton Live evaluation table

Live: General	
G.1: Are there facilities to help producers deal with option dilemma, allowing them to mark off decisions and move forward?	Only by rendering.
Live: Processing	
P.1: Can producers avoid manually working around CPU limitations?	Only by freezing which prevents many forms of editing.
P.2: Can producers easily change materials' voice and location in time without losing processing context and how it will sound?	Only by rendering.
P.3: Can automation parameters be associated with material or points in time as required?	Yes.
P.4: Can short term or diverging effect processing be created without adding complexity or CPU overhead?	No.
P.5: Do the object editing facilities allow for tail effects?	No.
P.6: Can results of effected processing be edited and viewed as first class audio without regard for underlying effects processing, while allowing underlying processing to be viewed and modified even after the results have been edited?	No.
P.7: Can processing complexity be reduced for use in a live setting?	Only through rendering or effect racks.
P.8: Is it possible to hide edits?	No.
P.9: Can looking ahead on the timeline be disabled?	Only in the session view.
P.10: Is it possible to create feedback loops and manipulate them interactively?	Yes.
P.11: Is it possible to have predictable results of effect processing?	Only with rendering or freezing.
Live: Voice	
V.1: Does the system provide voice abstractions to capture both the concepts of instrument parts and perceived parts?	No.

V.2: Does the system provide an abstraction for dealing with transient musical events?	No.
V.3: Does the system allow producers to make use of visual spatial layout to organise material?	Only on tracks.
V.4: Does the system provide voice abstraction facilities to reduce voice counts and the resulting overwhelming complexity?	No.
V.5: Does the system allow producers to view, edit and arrange multiple voices as if they are a cohesive whole?	No.
V.6: Does the system provide access to individual voices inside voice abstractions where multiple voices are amalgamated, and allow them to be compared with other voices from outside of the abstraction?	NA.
Live: Temporal	
T.1: Can material be represented in a hierarchy of temporal blocks up to the set level, allowing information hiding when desired?	No.
T.2: Can the producer work with multiple levels of the temporal hierarchy simultaneously?	No.
T.3: Can multiple contradictory temporal abstractions be represented?	No.
T.4: Can the producer build and view a piece's entire linear structure?	Yes, in the arrangement view.
T.5: Is it possible to specify variations in repeated material to ensure diversity and to ensure coinciding parts mesh?	No.
T.6: Can neighbouring and global context of repeated material determine the details of how it is performed?	No.
T.7: Can arrangement decisions be radically changed throughout the entire production process, including in a live eyes-free improvisational context?	Yes, in the session view.
T.8: Is placement of material against a metrical abstraction mediated by a system to take lead-ins into account?	No.

T.9: When material is captured without correct metrical abstraction information, can it be added after the fact?	Yes, particularly through warp markers.
T.10: Do the metrical abstractions provide facilities to change tempo and time-signatures at any time?	Yes.
T.11: Are producers able to manipulate material in terms of bars, beats, and sub-beats through provision of temporal editing and visualisation over metrical abstractions?	Yes.
T.12: Are metrical structure abstractions provided, allowing producers to learn common parameters for expressive timing?	No.
T.13: Can material be captured and refined through performance gesture?	Material cannot be refined through performance gesture.
Live: Reuse and Versioning	
R.1: Are individual repetitions editable, and do producers have control over how changes to one repetition affect others?	No.
R.2: Can references be maintained after derived material has been varied?	NA.
R.3: Can varied repetitions captured through performance be represented as such?	There is no provision for creating relationships between repeated material that has been recorded from an external source.
R.4: Can producers capture a snapshot version of any block of material that they can recall later?	No, not without manually copying it and placing it elsewhere on the timeline.
R.5: Can different versions of all types of data be maintained inside a single project file?	No.
R.6: Can producers work with the same data across multiple applications?	Only through rendering and export.
R.7: Can changes be propagated across multiple arrangements when required?	No.
R.8: Is there support for managing multiple takes as a single unit, including takes of automation?	No.
R.9: Does the system allow producers to easily track dependancies between material and the various projects it is used in?	No.

R.10: Can the producer archive and reuse all types of material abstractions from past projects, while retaining links to their previous contexts?	No.
R.11: Can the producer access the local library content from multiple project files, including contextual library material and larger mini-arrangement style material?	Yes, but only by viewing them in a distinct tree view which does not show them in context.
R.12: Can the producer create local library content without adding complexity to the composition?	Adding library content complicates the project's global channel structure.
R.13: Is there adequate support for rapid naming, searching, and name-versioning of all types of data?	Many types of data can be named, but name changes are not retained and are not searchable.
R.14: Can material be located spatially and visually rather than solely by name?	Yes, but only within the restrictions of the single timeline for the arrangement view. The session view has leeway for a vertical ordering.

10.3 Implications for design

Our research and analysis has illuminated a number of significant issues in the abstractions present in computer music user-interfaces, many of which result from designers' adherence to the ubiquitous multitrack-mixing model. We have used the knowledge gained from this process to evaluate aspects of Apple Logic 7 and Ableton Live 6. In this section we detail the implications of these findings for the design of future DAW systems by outlining various design concepts, some of which could be integrated easily into the existing multitrack-mixing model, while others would require a more radical redesign.

10.3.1 Designing for processing abstraction

In chapter 6 we itemised a number of important processing abstraction issues in current DAW systems. Our producers often required the direct manipulation opportunities of audio, but were instead required to deal with underlying effect processing parameters. As a result, they would resort to saving snap-

shots, and rendering to raw audio. In the process, the digital representation would dispose of important causal information and editing possibilities along with it. This leads us to suggest that DAW designers should look to a radical redesign of the ubiquitous real-time processing model, to acknowledge and embrace the crucial role that rendering plays in real world production activity.

V-rendering

At a basic level, this could include support for what we will term ‘virtual-rendering’ (v-rendering) operations. V-rendering would provide all the benefits of rendered audio, including visualisation and editing of effect tails, but retain (and abstract away) the details of underlying material, effects, and edits. Just as so called ‘non-destructive editing’ (as described in section 5.8.2) retains pointers to underlying audio files; non-destructive v-rendering could allow users to explore and modify underlying material and automatically regenerate a cache of the v-rendered result.

Unlike Apple Logic folder tracks, a v-render could be split into pieces and moved on the timeline, shifted to new voices, or placed in completely new projects; with *no change* to its audible result. In contrast to both frozen audio and folder tracks, unfolding such a block in its new context would reveal its generating content, *including the prior timeline information* which determined its audible characteristics. For example, fine tuning a block of v-rendered audio might require modifying effects parameters from what had originally been situated at an earlier point in time. Our proposed use of v-rendering would provide the benefit of creating a mechanism of abstraction that reduces complexity (by containing and hiding effect units, effect parameters, and source material), reduced **viscosity** (allowing movement to new contexts without audible change), and first class editing of what was apparently raw audio, while simultaneously removing performance overheads. Figures 10.3–10.5 illustrate this further.

The file storage requirements of such a system should be comparable to

Figure 10.3: In this mockup we show a stack of parameters and effects (feeding from bottom to top) resulting in an audio waveform. A piano-roll MIDI view (at bottom) feeds through a ‘drum machine’ instrument to create audio visualised as a waveform. A ‘simple delay’ (including a ‘dry / wet’ parameter graph) results in a transformed audio waveform (at top).

Figure 10.4: A v-rendered segment of audio copied from the resulting audio (from figure 10.3), could be placed in a new context in the composition.

Figure 10.5: The segment of audio (from figure 10.4) could be *unfolded* at any time, revealing the originating material and effects that defined it. Changes here to the originating material (such as modification of an effect parameter) will be reflected in the segment of audio, approximated with standard real-time processing until it has been re-rendered.

current systems, as in practice we observed producers manually mimicking such abilities through multiple ‘save as’ copies.

Opportunistic caching

This proposal could be extended more radically if the system automatically rendered (i.e. cached) *all* material in the background, even in the absence of visualised abstractions. Content that had not yet been cached could be processed in real-time as per usual, but other relatively stable content would be cached, vastly reducing processing overhead. This would require careful CPU and file-system I/O prioritisation to ensure real-time recording operations and latency were not negatively affected.

Our observations showed the importance of live interactive manipulation of controller or source data, and any automated rendering approach would need to allow for this. To accommodate this, if the user modified material on a track the system could initiate live processing and cross-fade from the cached material to the real-time version to provide audio feedback. Such transitions could be jarring, but their impact could be minimised by using various strategies. For example, any material that producers were actively working on would continue to be processed in real-time. Producers could also use a hinting mechanism, so instead of freezing material, waiting for rendering, and then being locked out of many editing options; they could mark material as a candidate for rendering that would be performed in the background.

Any change to material could demand a great deal of re-caching. For example, a change to the last note of a song might require regenerating the effect processing for an entire track. This issue could be mitigated by caching the internal state of effect units at various strategic points. Thus, this regeneration of cached material could commence from the last point where effect state was recorded. As a ballpark estimate, working with an effect requiring 10 seconds of state, using 44.1 kHz 16-bit stereo audio would require 1.68MB of storage to capture the state at any point of time. Some

effect units require no state, and so would not incur this caching overhead.

Implications for use

The approaches discussed above would offer the advantages of removing the creative blocks and interruptions presented by system resource limitations, converging on a predictable and deterministic audio result (not subject to the vagaries of playhead start point and effect state), allow fully fledged editing of abstracted audio without breaking ties to creation history, and still allow live editing of material as required.

We would expect this to work particularly well under the assumption that producers work on only a small proportion of material any one time. This assumption fits with our observations of Paddy Free working for several hours on editing a guitar track, Eric Vani focusing on editing vocal takes, Jim Frazier recording many vocal takes, and Andy Morton developing layered drum grooves. In all of these cases, our participants' actions were highly focused on editing a specific subset of the entire composition. The rest of the material could have been automatically cached. Detailed *quantitative* study of producers or instrumentation of music production tools could be used to validate our assumption that only a small proportion of material is edited at any one time.

10.3.2 Designing for voice abstraction

Chapter 7 described our findings relating to the abstraction and organisation of voices, and these too have implications for future DAW design. Firstly, we saw the impact that transient and duplicated voices had on track counts in traditional multitrack-mixing systems, and also the negative aspects of scrolling and zooming across a large number of tracks.

Track duration

One approach to address this might be to revise the current multitrack-mixing model where tracks must last the duration of an entire piece regardless of the length of their content. Instead, when the user is zoomed into a particular area of the timeline, the system could show just the voices of the material active at that point in time. As playback continued, new voices could be displayed only as required — although this would necessitate some animation of material in order to accommodate newly appearing voices while retaining ordering, as depicted in figure 10.6. This figure shows how playback of voices 1, 2, 3, 5, 6 and 7 could transition to voices 1, 4, 6 and 8. The voices located below 3 would have to move down to accommodate voice 4 as they would still be visible in the scrolling viewport. Once voices were no longer in the viewport, the remaining voices could move to reclaim the unused screen real estate.

Without the visual representation of tracks persisting throughout an entire piece, it would be necessary to provide new mechanisms to associate material with a particular set of voice characteristics for reasons of organisation or effect processing. As Paddy Free described:

“You have to organise at some point. If you call it tracks or channels — once you are adding more than one thing, there has to be a way of keeping a handle on how do I access that drum part or that bass part.” — Paddy Free

This could be accommodated through voice objects, where assigning material to a particular voice object would allow it to be correctly grouped; and provide the appropriate effect processing, audio input and output mappings, and record arming controls.

Such a scheme would be less vulnerable to the problems of representing transient material. Additionally, compositions in which different song sections feature distinct instrumentation would only show relevant material when viewing the chorus or the verse, removing large numbers of empty

Figure 10.6: Here we show the animations required as voices join and leave during playback of a piece. With eight total voices we can reduce complexity and scrolling by only showing those voices playing back during the time frame of the current viewport.

tracks and improving the ratio of useful information displayed. This proposal would need to be evaluated against the potential disorientation caused by the animation of voices, and measured against quantitative data regarding the regularity of voices joining and leaving playback that might trigger these animations. This design could even be provided as a complement to existing multitrack-mixing user-interfaces, with the user being able to toggle between the two views.

V-rendering voices

The v-rendering approach already described in the preceding section also shows promise for use as a voice abstraction mechanism. Our new-found knowledge of the reasons why producers render multiple voices into one provides us with clear motivation for a smart virtual rendering system. We can envisage a system where any number of voices are grouped and folded within each other, with the result being cached for uncompromised audio editing and visualisation purposes. Any one of these abstracted v-rendered voices could then be included in other v-renders recursively. We have demonstrated the importance of allowing producers to ‘unfold’ voice abstractions and work with them in the temporal context of other material, and would therefore provide means to allow this. Ideally, individual voices from an unfolded voice abstraction could be positioned next to any other voice, until it was time to be instantly folded back into the original abstraction without requiring the user to rebuild it from scratch. Changes to the contents of aggregated voice abstraction would cascade to the top level of the abstraction.

10.3.3 Designing for temporal abstraction

One of the clear results of our research is a rationale for re-examining the dichotomy between linear and non-linear sequencing paradigms. Our participants revealed the strengths and weaknesses of both linear and non-linear approaches through their various actions in the activity system, and in many

situations demonstrated their dependance on them both. In the worst case, we saw the labour intensive process of manually preparing and moving material back and forth between the MPC's non-linear sequencing environment to Pro Tools' linear sequencer. Each of these paradigm transitions resulted in significant data-loss as material was grouped and rendered down. In the improved case, participants using Ableton Live were able to move material between the two paradigms without losing processing information, such as associated effect units and object automation. Unfortunately, as we describe in our evaluation of Ableton Live in section 10.2.2, there is still significant information lost in the transition between the two paradigms.

Merging linear and non-linear approaches

This leads us to conclude that there could be significant advantages in attempting to merge aspects from both paradigms in a new system, rather than forcing producers to move material back and forth “through the portal” (as Paddy Free's described it). This might suggest a system in which material is in a non-linear triggerable form, but which we can interactively perform or edit into larger linear sections that are themselves interactively triggerable in a similar manner. In a system that retained the representation of the various levels of temporal hierarchical structure we could trigger both lower and higher level temporal groupings. The latter could soon converge towards a full linear arrangement, while retaining instant access to the advantages of non-linear triggering when appropriate.

One of the advantages of this approach is that it would allow multiple arrangements to coexist without becoming fragmented across multiple files on the file system. Secondly, it would allow the creation of segues between blocks of material as a natural part of linear editing. These segues could then be conveniently presented to the producer in a non-linear triggering scenario when moving from one block to another. With multiple arrangements embodying a huge wealth of data about how various musical sections can be combined, it is reasonable to suggest that these should be accessible

to producers encountering similar combinations in either new arrangements or live non-linear triggering contexts. This same logic applies equally to recalling general arrangement options. When we consider that each arrangement of the same material implicitly contains a wealth of information on the ways in which blocks of material can be combined, we conclude that this could become a useful database to be exploited in interactive performance situations. Utilising such information could allow producers to figuratively ‘navigate’ through an arrangement, being presented with likely transitions at each junction based on existing knowledge in the system. Such a process would more closely mimic the advantages of working with a group of live musicians and would exploit the structural compositional knowledge that accrue over the course of composing a song. This sort of approach contrasts dramatically with traditional non-linear triggering systems in which the quality of various possible transition are undifferentiated, and the only limited means of recording the fitness of transitions is to locate material adjacently.

Lead-ins, set lists, micro-timing, and gesture

Other findings from our investigation lead us to suggest the importance of including support for lead-ins into non-linear sequencer design, and the ability to easily load the next song while the current one is still being performed, or ideally, manage entire set-lists. Another design concept that stems from our research would be a mechanism for visually representing micro-timing information (vital for controlling the rhythmic ‘feel’ of a phrase) while viewing material at a macro beat and bar temporal level. While the numerical representation of micro-timing used so powerfully by Andy Morton could prove to be too complex for some users, a graphical representation might be more effective. One possible visualisation of this micro-timing is shown in figure 10.7. This example would allow the producer to see the early first snare drum event, followed by ‘laid-back’ (late) snare events that contrast with the early kick drum events. Just as Andy Morton developed an association between a micro timing number and a certain ‘feel’, producers could associate the

horizontal bar’s magnitude in a similar manner.

Figure 10.7: An adaptation of Ableton Live’s piano-roll MIDI event view with time running from left to right. The horizontal black lines have been added to represent micro timings that would not usually be visible at this zoom level.

Lastly, in section 8.5.2 we revealed a problematic dichotomy between entering temporal effects automation via performance gesture, and editing it graphically with a mouse. This could be perhaps addressed by allowing differential automation recording, where manipulating the controller during playback could shift the parameter in question from the baseline of the previous take.

10.3.4 Designing for reuse and versioning abstraction

In chapter 9, we described our findings surrounding issues of reuse and versioning abstraction, and these too have implications for future design of music production systems. Our interviews and observations exposed the startling lack of provisions for tracking versioning information, and the difficulties this can cause for producers.

Retaining patterns of reuse

At the level of individual blocks of material it is clear that manually ‘stash-ing’ older versions is an awkward and **error prone** workaround. Even using

such techniques, the system is essentially disposing of useful information by not explicitly retaining these links between source and derived material. Nowhere was this more evident than in participants' use of referenced copies (called 'aliases' in Apple Logic). As we described, our participants inevitably needed to vary referenced copies and hence break these references (using the application-specific equivalent of Logic's 'Turn to Real Copy' command). As a result, almost all material would become a 'real copy' isolated from its versioning and reuse history.

Lerdahl argued that it has been difficult to pin down theories regarding the associative structure of music (see section 3.2.1):

“...it is difficult to make a substantive theory of associations.
Within music theory this remains relatively uncharted territory.”
— Lerdahl [94, p.6]

In contrast, unlike the tradition of pure music theory, the activity of computer mediated music production exposes a variety of associations between musical material through the specific actions of the user. For example, copy and paste of material declares a clear associational structure without requiring a fully fledged theory to account for it. This observation leads us to propose a system that would retain this information, and in doing so, allow producers to access related material when required. This would allow the system to naturally develop a library of variations of material, and provide them as replacement options when requested by the user. For example, Emile De La Rey's labour intensive process of developing and laying out drum groove and fill variations might be eased if each referenced copy could be rapidly swapped out for a known repetition from this naturally developing library. Further, with variation information retained we might begin to address Troy Kelly's suggestion of a system to classify and semi-automatically randomising the selection of variations when requested, while retaining a strong foundation in the declarative sequencing tradition. In section 9.2.1 Tim Prebble described his use of a linked editing plugin that provides a basic form of audio

‘find and replace’ to crudely manipulate a shared aspect across multiple ‘real copies’. With the mechanism we have suggested above, such compensating plugins would be unnecessary, and common properties of reused and varied material could be edited when desired by cascading changes to related material. Of course, the consequences of such cascading edits would require the provision of adequate visualisation of which material would be affected, and allowances made to limit the scope of such shared changes. More advanced implementations could allow visualisation and manipulation of inheritance hierarchies, such as that shown in figure 10.8. These types of versioning history relationships could even be maintained between varying arrangements, and used to selectively propagate changes from one arrangement to another.

Figure 10.8: An early prototype showing just one possible visualisation of the inheritance relationship between repeated but varied material.

Variations for performance

Retaining reuse and variation linkages could have applications for the temporal abstraction design described in the previous section. If, as we have

suggested, the system retained information on the various ways in which material could be linearised, there might also be an application for reuse and versioning information in this temporal context. For example, a producer might develop a phrase with the form AABA, and elsewhere create a variation of the B phrase named B'. In a non-linear environment, when playing through an AA pattern the system could suggest the known phrase BA to follow (to complete the AABA form), and the alternative of B'A on the assumption that B' might be a viable replacement for B (to complete an AAB'A form). This would require the system to clearly differentiate speculative suggestions (such as AAB'A) from those that are known to work (such as AABA). Increasingly tentative suggestions (such as AAB''A and AAB'A'') could be ranked accordingly. This has obvious links to code completion tools for programmers, and text input prediction systems [53]. This mechanism could bring the producer's activity system closer to that of the instrumentalists in a band who naturally draw from a pool of available and known variations in live and composition situations.

Snapshot versioning

Another consequence of our findings is a strong motivation for implementing advanced forms of snapshot versioning, to allow producers to recall material in a specific known state. Freeing our participants from the difficulties of 'save as' snapshots and the resulting complexities of file system management could prove to be a major advance for their productivity and creativity. While version tracking of reused material (as described above) provides one form of history recall, it would be important to allow snapshot versioning of material even in the case where it was not being used in a unmodified form elsewhere. Instead of requiring producers to continue to 'stash' copies or use 'save as' to preserve historical state, our findings lead us to suggest a mechanism whereby any material at any abstraction level (including an entire arrangement or a single block of material) could be snapshotted for later recall through the context of that abstract object. This would en-

able the user to edit without undue concern, and provide a form of localised undo through that object's recorded history. We can imagine how such a system, when combined with the numerous mechanisms already described, could positively impact the situation Simon Rycroft described wherein previous arrangements were divorced from later compositional developments. Instead, our inbuilt snapshotting mechanism would allow Simon to retrieve the previous snapshot, and selectively introduce recent refinements that would have been retained as the variations accessible through each piece of ancestral material.

Library material

Chapter 9 also outlined the ways in which representations of reuse and versioning abstraction were at the heart of many library management issues. We saw how participants could be forced to store shared library material on the file system. Local project libraries also suffered as there was often no way to naturally store them in a project without complicating the track/channel structure of the developing composition, and they would need to be placed at some arbitrary point further up or down the single timeline. Additionally, library content best stored in relation to existing material in an arrangement was sometimes placed on muted tracks in the same temporal location, potentially complicating the editing and rearrangement of the audible material.

To better address these library issues there is a clear justification for allowing a more free organisation of library material outside of the multitrack-mixing model's single linear timeline, while retaining timeline relationships between library content where it exists. One way to address this would be to allow material to be placed on a 2D spacial substrate independent of temporal or track organisation. Material that *is* temporally related could be visually linked by a horizontal timeline, while unrelated library material can be spatially organised freely to aid location with spacial memory:

“Spatial abstraction reduces psychological loads by using our latent abilities in recognition, orientation, and positional memory,

thus shifting the challenges of navigation and assimilation from reasoning to simple perception.” — McCullough [100, p.145]

In this way, library content could be added to a project without adding full length tracks to a single monolithic timeline. Introducing external library content (or simply material from any other project) would bring along with it the rich history of reuse, variation, temporal grouping and structure, multiple voice structure, and inner processing structure; all hidden within an abstraction that would behave as a simple rendered (cached) audio region unless explored. One advantage of retaining such links is that it would be possible to address Christiaan’s problem of naming material, as the system would naturally retain naming information for all related material (differing versions, higher and lower temporal grouping names, song names, project names etc.). These could be used when searching for material by name.

With such rich context being retained within material, we could also provide the ability to jump to the original contexts in which material has been previously used, something that could be easily provided on top of such a system. Extending this further would allow this same functionality to cross between independent project files. This does however raise the problem of dependency tracking. As we have seen, the common technique our participants employed to deal with dependancies was to always take full copies of all project resources when forking a project, and to keep almost everything to avoid the risk of losing material and history. Providing inter-project dependency tracking facilities could remedy this situation, or development of specialised version control repositories modelled on those used in software engineering [45] or other disciplines [40].

10.4 Creativity and crafting with digital artefacts: Wider applications

At various points in this thesis we have framed our examination of computer mediated music production as one example of creative professionals crafting of complex digital artefacts. At this point, we can now reflect on how our approach — that of identifying how the provided abstraction mechanisms interact with the activity systems of users — can be applied to related domains.

Despite our focus on music production, we can already see signs of common abstraction and activity patterns that cut across to other digital artefact crafting activities. The cognitive dimensions present a set of vitally useful, but relatively low-level aspects relevant to notations. Extending beyond the scope of the cognitive dimensions of notations, there is value in developing a higher level pattern language specific to the digital artefact crafting systems used by creative professionals. Such a pattern language might include a description of the commonly occurring abstraction issue, examples, and known design solutions to address it. These patterns, documenting both the reoccurring tensions as well as potential solutions, could become a valuable resource to system designers working in these fields. In order to demonstrate the patterns that might emerge in such an analysis, we now present some preliminary observations about the links between our research and other systems outside of the music production field.

The influence of legacy user-interface metaphor

Just as we have seen with music production systems, mainstream cinema editing tools also suffer from a dependance on user-interface metaphor from legacy physical systems:

“This first generation of computer-based postproduction tools is still a very literal translation of their analog equivalent; they were

not redesigned to take advantage of the very distinct properties of digital media.” — Lew [97]

The linear / non-linear dichotomy

In an academic context, research on real-time cinema editing has grappled with issues of linearising temporal material, mirroring those we found in the dichotomy between linear and non-linear sequencing systems. One approach in the cinema context has been to allow narrative structure and constraints to be captured during the pre-production process, allowing appropriate material to be easily accessed in response to the particulars of live context [28, 97]. Retaining this higher level of abstract semantic information is analogous to the approach we have suggested for music production tools, allowing rapid real-time rearrangement. As we have seen, such “real-time” requirements are in fact also important for creativity in a studio environment, and we would expect the same might be true for video editing. Many similar domains also face issues with temporal abstraction and linearisation. Software for designing fireworks displays, narrative organisation systems for the management of novels during writing, interactive lighting display systems, and even project scheduling software contain timeline editing facilities that might face similar issues.

Timelines with tracks

Many applications also provide facilities analogous to the abstraction mechanism of ‘tracks’ that we explored in music production systems. Apple’s iMovie ’08 consumer grade video editing software has recently moved away from a track-based model, allowing sound-clips to be placed at any point without the need for a full duration track for that material. This change is illustrated in figure 10.9. Research into how well this abstraction mechanism fits with the demands of video editors’ activities would also inform the design of similar track-based systems, as found in our music production systems or the layered timeline in Adobe Flash and various 3D animation software.

Apple iMovie 5

Apple iMovie '08

Figure 10.9: A traditional track display from Apple iMovie version 5 above, and a new trackless display from iMovie '08 underneath.

Reuse and snapshots

In addition to timeline abstractions, Adobe Flash also provides abstractions for reuse that correspond to the issues we uncovered in music production systems. Flash allows a single “movie” object to be embedded in multiple places on the timeline. Changes to that shared movie object are replicated throughout each occurrence in the project. This type of abstraction mechanism (and all the issues that come with it) is present in many such systems such as DoView, shown in figure 10.10. DoView is a causation modelling tool for strategic planning and social program evaluation [60, 132], in which the user draws large networks of causes and effects across numerous inter-linked diagrams. While the users of such a system differ somewhat from what is associated with the term “creative professional”, many of the same issues arise here also. For example, as the same causal entity might need to appear on multiple diagrams, copy and paste has been augmented with an additional “paste as clone” option.

Figure 10.10: Screenshot from the outcomes logic modelling tool DoView.

Interestingly, our description of producers “stashing” backups of blocks

of musical material (in section 9.4.3) has been mirrored in practical use of DoView. Figure 10.11 shows a real world example of a nested list of diagrams including multiple versioned snapshots (“backups”) of each.

Slice List
Introduction	1
Software overview outcomes	2
Marketing outcomes	3
Marketing backup 16 Jul	3.1
Marketing backup 3 Jun	3.2
Marketing backup 28 Mar	3.3
Marketing backup 20 Feb	3.4
Software engineering outcomes	4
SE backup 16 Jul	4.1
SE backup 3 Jun	4.2
SE backup 28 Mar	4.3
SE backup 20 Feb	4.4
User interface outcomes	5
UI backup 16 Jul	5.1

Figure 10.11: A list of DoView diagrams from a real world project demonstrating manually saved backups of individual diagrams. — *Parker Duignan Ltd.*, Used with permission.

‘Save as’ and the file-system

At the file system level, just as with music production, DoView users may also save backup snapshots of models with ‘save as’ (see figure 10.12). Graphic designers can face the same familiar issues with manually managing snapshots. Figure 10.13 shows a real-world example of a graphic designer’s file-system structure, showing a similar manual versioning strategy to that which we observed in music production. From our anecdotal discussions with graphic designers, the lack of undo or permanent snapshots for specific graphical objects can be an issue here also, just as it was for our music producers.

Name ▲	Size	Type	Date Modified
MFE 07 Bus Capability V1-1 7-8-07	32 KB	DoView Document	8/08/2007 11:43 a.m.
MFE 07 Bus Capability V1-2 8-8-07	33 KB	DoView Document	8/08/2007 11:45 a.m.
MFE 07 Bus Capability V1-3 9-8-07	55 KB	DoView Document	9/08/2007 5:05 p.m.
MFE 07 Bus Capability V1-4 9-8-07	63 KB	DoView Document	9/08/2007 5:44 p.m.
MFE 07 Bus Capability V1-5 9-8-07	64 KB	DoView Document	9/08/2007 5:59 p.m.
MFE 07 Bus Capability V1-6 14-8-07	71 KB	DoView Document	15/08/2007 9:00 a.m.
MFE 07 Bus Capability V1-7 14-8-07	80 KB	DoView Document	15/08/2007 10:18 a.m.
MFE 07 Bus Capability V1-8 16-8-07	86 KB	DoView Document	16/08/2007 6:51 p.m.
MFE 07 Bus Capability V1-9 17-8-07	94 KB	DoView Document	17/08/2007 10:57 a.m.
MFE 07 Bus Capability V1-10 17-8-07	95 KB	DoView Document	17/08/2007 8:27 p.m.
MFE 07 Bus Capability V1-11 27-8-07	95 KB	DoView Document	27/08/2007 3:14 p.m.
MFE 07 Bus Capability V1-13 27-8-07	157 KB	DoView Document	31/08/2007 12:32 a.m.
MFE 07 Bus Capability V1-13 30-8-07	158 KB	DoView Document	31/08/2007 12:40 a.m.

Figure 10.12: A number of backup snapshots of a DoView model manually created with ‘save as’ functionality. — *Parker Duignan Ltd.*, Used with permission.

Processing and rendering

Finally, figure 10.14 shows a real world graphic design project in Adobe Photoshop, in which the designer has rendered a number of layers to a flat image for group editing. Just as our music producers had to manually manage, mute, and hide non-rendered versions of processing abstractions when rendering, this Photoshop example shows a hidden folder of non-flattened layers retained for backup purposes.

These preliminary examples and observations of crossovers between our research findings and the issues facing other creative professionals are an encouraging sign that a useful pattern language could be developed for use across these domains. It is also clear evidence that there is a case for examining these other systems (both activities and technologies) utilising a similar methodology to our own, in order to compare results and make contributions to each of these domains.

Figure 10.13: A graphic designer’s file system structure demonstrating manual management of various project versions and snapshots. — *Experimenta Ltd.*, Used with permission.

Figure 10.14: The layers in a graphic designer’s Photoshop project. Layers ‘Layer 75 copy 6’ and ‘Layer 75 copy 2’ are hidden (as the eye icon is turned off), and contained within an ‘Originals’ folder. The flattened (rendered) layer ‘Dull Ship’ was partially composed from these hidden layers that are retained for backup purposes. — *Experimenta Ltd.*, Used with permission.

Chapter 11

Conclusion

The purpose of this research was to learn about the fitness and difficulties created by existing music production abstraction mechanisms through the context of the work of music producers. The goal of this was to provide us with a better understanding of how to design abstraction mechanisms, both in music production tools, and for other digital artefact crafting systems.

This thesis has clearly demonstrated our success in this respect, by presenting a collective case study carried out with twelve professional music producers, and laying out the detailed analysis and implications of what we have learnt.

Our central research question was:

How is the work of professional music producers affected by the particular abstraction mechanisms in the user-interfaces of music production software?

In this thesis we answered this by first providing an introduction to our research in chapter 1 and outlining the vital background necessary to understand the domain and related research areas in chapter 2. We presented existing models and abstractions of music in chapter 3 which were essential in understanding our participants' activities. Chapter 4 provided details of our methodology; specifically our use of collective case study, cognitive

dimensions, and activity theory; and the methodological refinements we undertook in order to tackle this research question. In chapter 5 we presented an activity theory based analysis of computer mediated music production that was founded on our observations and interviews with professional producers. We then explored the interaction between these participants' activity systems and the abstraction mechanisms in their software tools. Real world abstraction and activity issues surrounding audio processing were described in chapter 6. Chapter 7 dealt with our discoveries of the interactions between participants work and abstraction mechanisms for breaking up and manipulating musical material as distinct or combined voices. We then examined how our participants worked with time based structuring of audio through chapter 8, and in chapter 9 we detailed our discoveries about how participants reused material, and the abstractions available to support this work. In chapter 10 we applied the set of evaluation questions which were developed from these analysis chapters to evaluate two music production systems. We synthesised what we had discovered to inform a number of novel design concepts and prototypes and explored the applicability of our findings to other related software tools. Finally, in this chapter we revisit the contributions of this thesis and outline possibilities for future work.

11.1 Thesis contributions

In chapter 1 we introduced the three central contributions of this thesis. We can now revisit these to reflect on what we have learnt.

Contribution 1: Analysis of abstraction mechanisms

Our first contribution is the detailed description of our findings explicating the interaction between existing music production abstraction mechanisms and our participants' work. This is contained in chapters 6 through 9, detailing professional producers' interactions with processing, voice, temporal,

and reuse and versioning abstractions. These four chapters directly answered our core research question:

How is the work of professional music producers affected by the particular abstraction mechanisms in the user-interfaces of music production software?

These findings included, but were not limited to: discovering the crucial role that rendering plays in professional production, despite the provision of abstraction mechanisms such as freezing and folder tracks; the mismatch between provided abstractions and both transient processing and voices; the problem of temporarily diverging effects; the difficulties of exploding track counts and requirements for voice aggregation; the importance of hierarchical representations in professional music production, from organising an entire set to manipulating micro-timing for rhythmic feel; the problematic contradiction between linear and non-linear temporal abstractions; limitations in representing variations in repeated material; producers manually managing snapshot versioning through ‘stashing’ and ‘save as’ copies; and the importance of library material for reuse.

Contribution 2: Analysis of computer mediated music production activity

Our second contribution is presented in chapter 5 and provides a detailed explanation of computer mediated music production through the framework of activity theory. This is based on our interviews and observation field work, and grounds the analysis in examples from the study participants. We presented a full description of the components of this activity system, including the tools we saw in use, the subjects, their differing roles, the types of actions they carry out, and the significance of externalisation of action through the activity’s objects and outcomes. We exposed core conflicts in the activity system: the contradiction between studio and live performance, the tension of

rendering material and keeping options open, the paralysing effect of option dilemma, and the problems of conceptual burden created by the complexity of the digital artefact object. Chapter 3 also provided a vital component of this analysis, as it described the historical development of the multitrack-mixing model which has become a central part of the production activity. This combined activity analysis provides a useful foundation for other research into computer mediated music production, as well as a reference for the activity analysis of similar activities in which creative professionals create and manipulate complex digital artefacts.

Contribution 3: Evaluation criteria, evaluations, and implications for design

To make the above findings more accessible to the designers of music production systems we distilled them into a set of evaluation questions divided into the various types of abstraction identified. These were presented in chapters 5 through 9, and are collated in appendix C. These evaluation questions can be systematically applied to new or existing music sequencing system designs. Doing so will ensure that the concrete findings of this research can maximally inform future designs. We applied these evaluation criteria to two popular DAWs and described the more significant results in section 10.2.

In section 10.3 we illustrated how research into how abstraction mechanisms interact with subjects' activity systems can yield new design possibilities. In doing so, we proposed a number of novel abstraction mechanisms for music production systems including v-rendering for both processing and multiple combined voices, allowance for short term voices, hybrid linear and non-linear sequencing models, anacrusis support for non-linear sequencing, combined micro and pulse-level editing, differential performance editing, automatic history tracking of material, inbuilt snapshot versioning for large and small grained objects, and library material in mixed linear and non-linear sequencing form. Some of these mechanisms would require a radical redesign

of current audio sequencing tools, while others could be added into existing models.

Methodological contribution

In addition to our research findings, we created a reusable methodological tool, called the activity interview. In chapter 4 we presented this new interview script based on the activity theory checklist. The original activity checklist is founded on the rich but complex literature of Activity Theory, and was one of the few significant efforts to make it accessible to Human Computer Interaction practitioners. While both Activity Theory and the checklist provide an ideal foundation for understanding the work of creative professionals we found significant practical difficulties in applying the checklist in a field research environment (see appendix A.1). Our activity interview script published elsewhere [59] and also provided here in appendix B, was systematically developed from the checklist. This was created for the purposes of this research in order to provide a resource that can be understood by interview subjects in any domain, both free of theoretical terminology and instantly actionable. Our interview script also has the advantage of being accessible to practitioners without a formal understanding of activity theory, while allowing them to still take advantage of the perspective it provides. This interview script demonstrated its utility in our research through the information it elicited from participants, directly informing not only the activity analysis in chapter 5, but also providing crucial contextual data for chapters 6 through 9 which analysed the impact of abstraction on participants' activity.

11.2 Generalisability

Having concluded this study we can make some final remarks about the extent to which our findings might be generalised.

Firstly, our results were based on interviews and observations with twelve professional producers that were chosen as cases that exemplified the computer mediated music production activity system described in chapter 5. The reoccurring abstraction issues found across these cases provide compelling evidence that these are likely to be serious issues for at least a significant proportion of such producers.

We expect these findings to hold for novice and developing computer music producers. These producers would lack the detailed knowledge and experience of how to work around problematic or missing abstraction mechanisms. For example, professional producers' use of manual backup versioning and 'stashing' techniques, in combination with rendering, allowed them to accommodate for the failings of their software's abstraction mechanisms. For novice producers without these hard earned skills, it seems likely that these problems would be exacerbated.

In addition to our twelve professional producers, interviews with our two general audio editing participants demonstrated a number of overlapping issues with those of our core participants. This is an encouraging sign that at least a subset of our findings are relevant to this larger class of audio editing activity, although we would also expect these cases to have a number of their own distinct abstraction issues.

Our last three interviews with experimental computer music programmers uncovered a small number of identical issues, and a number of related issues with abstraction mechanisms in common with our professional producers. Despite their radically different approaches and tools, these participants also dealt with processing, voice, temporal, as well as reuse and versioning abstractions. However, in most cases the abstraction mechanisms themselves were different, resulting in a different configuration of issues in their activity.

The methodology and analysis presented in this thesis also provided a template for application to the many other forms of computer mediated activity involving creative professionals crafting digital artefacts. In section 10.4 we described our initial observations regarding the many commonali-

ties between abstraction issues in computer mediated music production and these other domains and how our findings could be generalised. The research findings we have presented here could form the basis of a pattern language of abstraction issues common to these various digital crafting activities, which would prove useful to interaction designers developing such systems in the future.

11.3 Future work

This research leads to several other avenues for future work:

1. Firstly, this research does not claim to exhaustively cover all possible types of musical abstraction, only those that were the most pertinent to our participants. The most prominent example outside of our work would be the abstraction of pitch, which has its own implications for the music production activity. Examination of the entire range of potential musical abstraction mechanisms and their impact on the music production activity is an area for future work.
2. The detailed qualitative experience of producers that we gathered for this research would be well complemented with quantitative information on system use. For example, corpus analysis of a large number of music production files would provide significant information on project organisation techniques and semantic structure.
3. In this thesis we presented a number of design mock-ups and concepts in response to our findings. Further development and user testing of functional prototypes would provide a wealth of data for continued refinement.
4. The study we have conducted revealed interesting aspects of the relationship between abstraction mechanisms and music producers' activity systems. Similar studies of other digital artefact creation systems

and their users' activities would be useful contributions to their own domains, as well as allowing comparison between them.

5. Patterns emerging from these comparisons could produce a useful pattern language, where solutions from one domain could be reused in others.

Appendix A

Activity Checklist to Interview

This appendix provides additional details on the development of the new activity interview, continuing the discussion found in section 4.4. In this appendix, we provide the critique of the activity checklist that motivated our interview tool, examples from the development of our activity interview, and reflections on our experiences in its use in the field.

A.1 Checklist critique

While both activity theory and the checklist provide an ideal foundation for understanding the work of creative professionals, we found significant practical difficulties in applying the checklist in a field research environment. Some of these difficulties are as follows:

Operationalisation The first issue with the checklist is that the items are not presented in an operationalisable form. The checklist items are worded in a manner which does not lead the mind to apply the checklist item. The following item is typical wording from the checklist:

Eval 3.4: Self-monitoring and reflection through externalization

This phrase in itself does not pose a question, or impel the HCI practitioner to action. Having to interpret a list of more than forty such

phrases creates barriers to the comprehension and accessibility of the checklist. In an interview or other field research setting such as ours, practitioners need the content of a checklist to be rapidly accessible so that they can quickly determine the relevance of various items to the current context.

Theoretical terminology Another problem with the checklist is that while it attempts to make the ideas of activity theory “concrete”, this very goal is negated by the use of specialist activity theory terminology. In particular, the checklist uses specialised meanings of the terms *action*, *operation*, *internalisation*, and *externalisation*. The lengthy checklist preamble uses these terms but does not define them clearly, and would be impractical to refer to in a fieldwork context.

Number of items Depending on the version used, the checklist has between thirty-seven and forty-three items. In order to make the checklist easier to apply and internalise, it would be desirable to minimise the number of checklist items where any redundancy is present. This is primarily important as the checklist is expected to be used in a field study context [85, p.30]. A number of the items in the checklist closely relate to each other and could be described as different specialisations of a single activity theory concept or question.

Design versus evaluation Kaptelinin and Nardi’s approach divides the possible usage of the checklist into two camps, evaluation and design. Providing two versions of the checklist creates a false dichotomy, making it unwieldy for use in the in-between cases. In our situation we need to use the checklist to understand current tool use, with the explicit goal of motivating new and *as yet undefined tools*. As such, we would find a need to use elements from both versions, which is difficult to achieve with the existing checklist.

Readiness for application Another potential concern with the checklist

is that even though it claims to provide a concrete activity theory resource, there is some conceptual work needed before it can be practically applied. As a first step to using the checklist in this way, practitioners need to rework the checklist into a number of coherent questions presented in language that subjects in the domain of study will easily understand. For practitioners focused on rapid results, this could be a challenging and lengthy process in itself.

Instead of having to develop a series of interview questions based on a theory they have not fully internalised, practitioners could make good use of a series of checklist based interview questions using everyday language. In applying these questions in their domain of investigation, we could expect practitioners to develop a more concrete understanding of activity theory. In the next section we introduce a new series of interview questions which have designed to fulfil this purpose.

A.2 Developing the activity interview

Section 4.4.1 outlined the three steps used to systematically develop the activity interview, and figure 4.1 illustrated the mapping from the checklist to interview questions. This section presents a number of illustrative examples from this process.

Several groups of original checklist items were combined for the final question list. Activity theory's highly related notions of the Activity, Action, Operation; and Motive, Goal, Condition hierarchies are addressed by several items. The following items all call on the practitioner to determine the elements of this dual hierarchy in their domain:

Design 1.2: Goals and subgoals of the target actions (target goals)

Design 1.4: Decomposition of target goals into subgoals

Design 1.9: Integration of individual target actions and other actions
into higher-level actions

The distinction being made between the first and second item is far from clear. Additionally, since goals and actions are so closely related it does not seem ideal for them to be addressed in separate interview questions as seen in the third item. Instead, we can ask subjects to reflect on how they achieve their tasks, and how these actions relate to their goals. This allows the third item to be integrated with the other two into the following compound question:

Interview 1.2: *Can you take me step-by-step through the process of how you complete your activity, and tell me how this process can vary.*
<Ask about sub-goals during the account where appropriate>

By presenting several closely related checklist items as a single compound question, we can expect the subject to reflect on the important relationships between their goals and sub-goals, and the actions that they use to achieve them. Addressing these issues as separate questions could lead to a fragmented and incomplete response.

Another example of this type of amalgamation is our approach to several checklist items that deal with externalisation:

Eval 3.8: Coordination of individual and group activities through externalisation

Eval 3.9: Use of shared representation to support collaborative work

Eval 3.10: Individual contributions to shared resources of group or organisation

Each of these items overlaps the others, but it is not immediately obvious exactly how. The first item deals with both individual and group activity,

which means all activity. It emphasises coordination through externalisation. The second item refers to shared representation, which is a form of externalisation, and focuses on just group activity. The first item's issue of coordination through externalisation is simply a special case of the second item — a shared representation to support collaborative work. The third item deals with contribution to shared resources, which is another product of externalisation and group activity. All of these items can be essentially captured by an umbrella question eliciting what is externalised, and how externalisations are used — including use for coordination. After removing the theoretical terminology, we can express this to subjects in the following question:

Interview 3.4: How do you use representations of your work — documents, notes, software, and talking etc. — to collaborate and coordinate with others?

To use the checklist as a unified tool to support evaluation for prospective design, it was necessary to identify the variations in wording between the evaluation and design versions of the checklist. Often the differences in wording do not make a significant change to the item, and simply create potential confusion for HCI practitioners wanting to utilise elements of evaluation and design. For example, the following two items are based on the same underlying activity theory concept:

Eval 3.3: Time and effort necessary to master new operations

Design 3.2: Time and effort necessary to learn how to use existing technology

Both of these items are attempts to determine how subjects learn to use tools in a specific activity. The same items can both be addressed in an interview context with the following question:

Interview 3.2: How hard did you find it to master your tools, and what should have been easier?

There are several checklist items that include specialist activity theory terminology which would not be understood by interview subjects. One such example that was discussed above is the use of the term “externalisation”. Another example is contained in the following item which uses the technical terms “actions” and “operations”:

Eval 1.13: Support of mutual transformations between actions and operations

Again, it is non-trivial to reword this item as an interview question that can be easily understood. HCI practitioners who are learning activity theory may not be ideally placed to try and frame this in the clearest terms for their subjects. We could address this by asking:

Interview 3.6: Are there things that you do that you used to have to think about, but can now do automatically?

The above interview questions are just a selection of our full set of thirty-two questions in appendix B; a resource that can now also be used by practitioners, and a useful methodological contribution of this research. Further details were published in [59]. Our experience in applying this resource is detailed in the following section.

A.3 Activity interview in practice

In general, the new activity interview tool worked well. Subjects gave meaningful and often insightful answers that related directly to corresponding activity theory concepts. This helped minimise two forms of researcher bias. Firstly, the questions are based on activity theory without being tailored by the researcher to the specific domain being studied. This means there

is more leeway for the participants to draw unforeseen connections between their domain and activity theory concepts. Therefore the researchers expectations do not unduly limit the results. Secondly, basing analysis solely on observation and resulting questions would be inherently limited. Producing a completed piece of music will typically stretch from weeks to years, a time scale that makes detailed observation of the entire process impractical. Obviously, the subjects themselves are the only people with knowledge of their entire activity, and how it unfolds over these macro-timescales. This form of bias stems from the researchers limited exposure to the entirety of the activity, and can only be offset by incorporating the subject's own reflection. With the addition of observation and the researcher's analysis we can hope to get closer to a full picture of the activity.

Process and results

Practitioners should not expect any trivial one-to-one mapping between an activity theoretic analysis, and their system design. However, such an analysis is a crucial foundation for successful design. In our design process we found that our activity interview helped us ensure that we had covered all of the major issues in understanding the activity system with each interview subject.

The information that we uncovered from these interviews was invaluable for our analysis. For example, by highlighting the *activity* as the unit of analysis, rather than just lower-level tasks, the interviews brought out the contradiction created in many tools between the studio and live performance. Producers' compositions are typically being crafted for dual use — both for distribution on audio media such as CDs, but also for performance in a live context. Additionally, producers often expressed that their work was never truly completed, and that album, single, and re-mix versions of songs were common. Therefore a design goal of our system has become to allow producers to manage multiple variations of a composition, and allow modifications of one to naturally be reflected in the others where appropriate.

The activity interview also brings out issues relating to collaboration. For example, we found that often more than one person would want to interact with the computer representation of the composition. This raises interesting design possibilities of having multiple people working on different aspects of a shared composition simultaneously. With collaborators often developing a shared composition in separate locations, it is also important to facilitate merging diverging versions from different sources.

Higher-level questions

In addition to the checklist items, the checklist paper also provides 23 “sample questions”. The sample questions demonstrate the type of questions that the HCI practitioner could ask about an activity. An example of one of the provided questions is:

Sample 3.4: Does the system provide representations of user’s activities that can help in goal setting and self-evaluation?

This question combines checklist items concerning externalisation *and* goals. The sample questions were compelling enough for both Macaulay [85] and Fjeld et al. [65] to use explicitly in their research. Macaulay described them as being “particularly useful.”. However, these are sample questions that are representative of the *type* of questions that should be asked, but far from a complete resource of the questions that should be considered. By using the sample questions as a primary resource there is a very real danger of failing to consider other important questions not included in the samples. The underlying problem here is that the checklist paper provides no guide on how to derive similar questions. In fact the main body of the checklist paper does not mention them at all.

On closer examination, rather than being merely the checklist items reworded as questions, the sample questions work at a higher level — each synthesising several orthogonal checklist items. The real value of the sample questions lies both in their operational nature (as opposed to the abstract

checklist items), and in how they make links between activity theory concerns. The unfortunate consequence of this is that attempting to create an exhaustive list of these higher-level questions would result in a combinatorial explosion which would be far from practical.

Our newly developed activity interview addresses this through use in a semi-structured interview context. This takes advantage of the natural ability of people to make linkages between comments over the duration of a conversation. As these linkages naturally occur to the interviewer or subject it is important to follow these connections with followup questions. Such spontaneous questions from the interviewer, and insights from the subject are a natural way to discover relevant replacements for the sample questions.

One of the strengths of our newly developed activity interview is that it provides the vehicle for maximising the number of such connections that will be made. By raising all of the salient checklist items in discussion with subjects, these higher level questions naturally arise. Without such a set of interview questions there is a real danger that the researcher will be limited to only the provided sample questions.

Hierarchical goal analysis

Our initial activity interview included a question asking about goals and subgoals in the activity.

Interview 1.2: What are the goals, subgoals and supergoals you are trying to achieve in your role?

One problem with this question was that it proved to be difficult to get a simple hierarchical account of actions and goals in music production. On reflection it became obvious that this is because there is no simple hierarchical description of music production. This activity is open-ended and non-prescriptive, and as such the actual configuration of actions and goals varies between one musical piece and another, and between each subject depending on their patterns of collaboration, tools, and work habits.

Subjects had some difficulty in knowing how to answer this question. Subjects tended to answer in terms of a “for example” account of the actions they might go through in producing a track. They made it very clear that such an account was merely one possible scenario, and that in actual practice the order and details of their actions varied greatly. As the subject gave their account it was possible to query them about their goals along the way. This example shows the impracticality of asking subjects to develop a high-level and idealised abstract description of their actions and goals. They proved to be much more comfortable giving a concrete account, and providing cues as to how this can vary from case to case.

This interview question needs to be reformulated in concrete terms, with room for the subject to explain how the activity can vary:

Interview 1.2 Can you take me step-by-step through the process of how you complete your activity, and tell me how this process can vary. <Ask about sub-goals during the account where appropriate>

Conflict

The checklist uses the term “conflict” in checklist items dealing with tensions between various goals or various actions. The paper itself does not explain this term in any more detail. The use of this term here is somewhat confusing. Activity theory literature has typically used the concept of *breakdowns* or the wider notion of *contradictions* as popularised by Engeström [62]. Even the book *Context and Consciousness* [106] which was edited by Nardi herself does not include “conflict” in its index.

Despite the unusual choice of this term, we decided to use it in the activity interview. One of the goals of the activity interview was to embody the checklist and the expertise it embodied. Changing the terminology unnecessarily would not be ideal. Unfortunately this term caused much confusion when used in interview format. One of the questions was:

Interview 1.5 What conflicts are there between these goals?

The subjects almost universally interpreted the term conflict to mean *interpersonal* conflict, rather than the more abstract idea a form of tension or contradiction between the goals themselves. It is interesting to note that the checklist had already limited its analysis of conflict to goal and action conflict, only a small subset of the types of contractions that can occur between parts of an activity system. For example, contradictions can occur at different levels between tools, roles, and rules in addition to goals and actions. Inadvertently limiting the analysis to just *interpersonal* conflicts over specialised it even further. In the latest version of the activity interview we instead use the term “contradiction”.

Internalisation

The checklist paper claims to address the problem that concepts such as internalisation and externalisation are “somewhat abstract when it comes to the actual business of working on a design or performing an evaluation”. However, the checklist simply reuses these same terms but in an itemised form, which can cause difficulties. For example, in developing the first version of the activity interview we made the mistake of equating internalisation with the very different concept of a task’s transformation to sub-conscious operations. Both internalisation and sub-conscious operations are important concepts in activity theory, but have very distinct meanings. This confusion reduced the effectiveness of the interviews, and as a result hindered the perceived usefulness of activity theory.

Issues regarding internalisation are highly important in our particular domain. In our other complementary interviews using the cognitive dimensions questionnaire, subjects made statements about how they had been “colonised” by the model and work procedures of their music production software. This phenomenon was not raised in response to our activity interview.

Externalisation

Externalisation is another central concept of activity theory. We found that our activity interview questions about externalisation did not get the detailed answers that we might have expected. Interestingly, the domain of music production is permeated by profound externalisation. Most of the subject's time is spent working with externalised abstract representations of the object of the activity. It is likely that activity in other domains which are more internalised would see an larger benefit from these questions in the interview.

Evaluation versus design

Our activity interview merged the evaluation and design checklists. This suited our goal of analysing the *current* activity and tools, while looking forward towards how we might design a new tools.

In our interview's section on change and development of the activity, we asked about what might be possible in a new tool. This was followed with additional questions on how this tool could change the subject's activity. In this case, our subjects found it very difficult to imagine new tools, which we can attribute to their accounts of having been "colonised" by their current tools. Given the difficulty of imagining new tools, further questioning about the impacts of such hypothetical tools gained us little further insight. It is difficult to know how much we can attribute this phenomena to our specific domain, or if there is a more general problem here. One confounding factor is that producers who extensively use computer music tools for composition have by definition found a fit between their activity and their tools. Other potential computer music producers have no doubt been put off by the lack of fit between their music production activity and the design of current tools.

Limitations of the interview

With the interviews completed we can think about which aspects of the activity they failed to illuminate. The above sections highlight problems

due to various factors, but there are some things that the interview simply omitted. Some of these limitations can be traced back to equivalent gaps in the checklist, or activity theory itself.

An important aspect of computer music production is how the design of tools affect the resulting musical outcome. We can expect similar concerns to exist in many other domains where we might want to use activity theoretic analysis. We found that the activity interview did not directly lead to discussion in this area. The closest the activity checklist gets to raising this aspect of activity is with the following item:

Eval 4.2: Effect of implementation of target technology on the structure of target actions

However, the structure of actions is only a small part of the concerns here. It is of course important to develop an understanding of how tools affect the structure of our actions. We also need to know how the tool interacts with and shapes our goals, and the object of our goals. Tools both empower and constrain our abilities through their specialisation. When using a tool “we can sense some things better, and we can alter some things better, but others not at all” [100].

Another limitation of our interview is it did not give us much insight into the cultural-historical reasons for developments in computer music production. Uncovering this knowledge is probably best captured through complementary research into the historicity of the domain. In chapter 3 we provided a historical perspective to the development of the multitrack-mixing model.

Irestig et al. [81] found that the activity checklist “does not highlight the management of signs and symbols in design as a significant and separate aspect of the process”. We also foresaw this limitation, and dealt with it by also interviewed subjects with the cognitive dimensions questionnaire which focuses on important notational and symbolic features of tools. The detailed and crucial information uncovered in these interviews was not duplicated in the activity interview. This validates the claim that this is a limitation

of the activity checklist, but we should not conclude that the checklist, or our activity interview, should be expanded to include these aspects. The cognitive dimensions questionnaire worked well in tandem with our activity interview. Finding research methods that complement this weakness seems the best course of action.

A.4 The unified activity checklist

This section presents a unified version of Kaptelinin and Nardi's activity checklist [85] that we developed by merging the evaluation and design checklists into one. This provided the basis for our activity interview in appendix B.

Means / Ends

- 1.1: People who use the target technology
- 1.2: Goals and subgoals of the target actions (target goals)
- 1.3: Criteria for success or failure of achieving target goals
- 1.4: Decomposition of target goals into subgoals
- 1.5: Setting of target goals and subgoals
- 1.6: Potential conflicts between target goals
- 1.7: Potential conflicts between target goals and goals associated with other technologies and activities
- 1.8: Resolution of conflicts between various goals
- 1.9: Integration of individual target actions and other actions into higher-level actions
- 1.10: Constraints imposed by higher-level goals on the choice and use of target technology

1.11: Alternative ways to attain target goals through lower-level goals.

1.12: Troubleshooting strategies and techniques

1.13: Support of mutual transformations between actions and operations

1.14: Goal that can be changed or modified, and goals that have to remain after new technology is implemented

Environment

2.1: Role of Target technology in producing the outcomes of target actions

2.2: Tools, other than target technology, available to users

2.3: Integration of target technology with other tools

2.4: Access to tools and materials necessary to perform target actions

2.5: Tools and materials shared between several users

2.6: Spatial layout and temporal organisation of the working environment

2.7: Division of labour, including synchronous and asynchronous distribution of work between different locations

2.8: Rules, norms, and procedures regulating social interactions and coordination related to the use of target technology / actions

Learning / Cognition / Articulation

3.1: Components of target actions that are to be internalised

3.2: Knowledge about target technology that resides in the environment and the way this knowledge is distributed and accessed

- 3.3: Time and effort necessary to master new operations / learn how to use existing technology
- 3.4: Self-monitoring and reflection through externalisation
- 3.5: Use of target technology / possibilities for simulating target actions before their actual implementation
- 3.6: Support of problem articulation and help request in case of breakdowns
- 3.7: Strategies and procedures of providing help to other users of target technology / colleagues and collaborators
- 3.8: Coordination of individual and group activities through externalisation
- 3.9: Use of shared representation to support collaborative work
- 3.10: Individual contributions to shared resources of group or organisation

Development

- 4.1: Use of target technology / tools at various stages of target action "life cycles" — from goal setting to outcomes
- 4.2: Effect of implementation of target technology on the structure of target actions
- 4.3: Transformation of existing activities into future activities supported with the system
- 4.4: New higher-level goals that became attainable after the technology had been implemented.
- 4.5: Users' attitudes toward target technology (e.g., resistance) and changes over time
- 4.6: Dynamics of potential conflicts between target actions and higher-level goals

4.7: History of implementation of new technologies to support target actions

4.8: Anticipated changes in the environment and the level of activity they directly influence (operations, actions, or activities)

4.9: Anticipated changes of target actions after new technology is implemented

Appendix B

Activity Interview

<Discuss and define what the activity is for this person and resolve any questions>

GOALS

1.1: What are the different roles of the people involved in this activity? What is your role?

1.2: Can you take me step-by-step through the process of how you complete your activity, and tell me how this process can vary.

<Ask about sub-goals during the account where appropriate>

1.3: How can you tell when you have successfully completed each goal?

1.4: How could your main goals be achieved differently?

CONTRADICTIONS

1.5: Are there any tensions between these goals?

1.6: What tensions might there be with goals from other activities you or others are involved in?

1.7: How could you / do you resolve these tensions?

TOOL USE

I will now ask you a series of questions about the tools you use in this activity. Tools include all sorts of artefacts such as pencils and paper, the notes we take, the theories behind them, and more traditional technology tools.

<Give other examples from the specific activity domain>

2.1: What tools do you use and what for?

2.2: What other tools do you or could you use in this activity?

2.3: Do you have access to these tools?

2.4: How could the tool be integrated with your other tools?

2.5: Which tools do you use when, and how do you organise them?

WORKING WITH OTHERS

2.6: How are these tools used and shared with others?

2.7: How do you divide this activity between people, and at which points do you have to wait for them to complete their work?

2.8: What are the explicit or implicit rules, norms and procedures influencing how you work?

INTERNALISE

3.1: How have your tools affected how you think and reason about your activity, and how much of your activity do you perform in your head?

3.2: How hard did you find it to master your tools, and what should have been easier?

EXTERNALISE

3.3: How do you deal with problems in this activity when they become too complex to manage in your head?

3.4: How do you use representations of your work — documents, notes, software, and talking etc. — to collaborate and coordinate with others?

3.5: When trying to solve a problem, do you ever simulate the situation to explore possible solutions?

TRANSITION

3.6: Are there things that you do that you used to have to think about, but can now do automatically?

HELP

3.7: When things go wrong, how do / could your tools let you express these problems and request help?

3.8: How does the system provide help to other people?

3.9: What knowledge is there about these tools (other than that provided by the tools), and how can you get access to it?

LIFE CYCLE

4.1: How do your tools fit into your workflow?

4.2: How do your tools shape how you work, and influence what you can and do produce?

CHANGE

4.3: What new things could be possible in a new tool?

4.4: What tensions are there between your goals and how you attempt to achieve them, and does this change over time?

4.5: How might your working environment change - people, technology, rules, work etc. as a result of new tools?

4.6: What could you do differently with better support from a new system?

4.7: How have previous systems you have used affected how you performed your activity and when you moved from older to newer tools in the past, how did this affect the activity?

ATTITUDES

4.8: What are your attitudes towards new technology, and how do you see them changing over time?

Appendix C

Evaluation Questions

General

G.1: Are there facilities to help producers deal with option dilemma, allowing them to mark off decisions and move forward?

Processing Abstractions

P.1: Can producers avoid manually working around CPU limitations?

P.2: Can producers easily change materials' voice and location in time without losing processing context and how it will sound?

P.3: Can automation parameters be associated with material or points in time as required?

P.4: Can short term or diverging effect processing be created without adding complexity or CPU overhead?

P.5: Do the object editing facilities allow for tail effects?

P.6: Can results of effected processing be edited and viewed as first class audio without regard for underlying effects processing, while

allowing underlying processing to be viewed and modified even after the results have been edited?

P.7: Can processing complexity be reduced for use in a live setting?

P.8: Is it possible to hide edits?

P.9: Can looking ahead on the timeline be disabled?

P.10: Is it possible to create feedback loops and manipulate them interactively?

P.11: Is it possible to have predictable results of effect processing?

Voice abstractions

V.1: Does the system provide voice abstractions to capture both the concepts of instrument parts and perceived parts?

V.2: Does the system provide an abstraction for dealing with transient musical events?

V.3: Does the system allow producers to make use of visual spatial layout to organise material?

V.4: Does the system provide voice abstraction facilities to reduce voice counts and the resulting overwhelming complexity?

V.5: Does the system allow producers to view, edit and arrange multiple voices as if they are a cohesive whole?

V.6: Does the system provide access to individual voices inside voice abstractions where multiple voices are amalgamated, and allow them to be compared with other voices from outside of the abstraction?

Temporal abstraction

T.1: Can material be represented in a hierarchy of temporal blocks up to the set level, allowing information hiding when desired?

T.2: Can the producer work with multiple levels of the temporal hierarchy simultaneously?

T.3: Can multiple contradictory temporal abstractions be represented?

T.4: Can the producer build and view a piece's entire linear structure?

T.5: Is it possible to specify variations in repeated material to ensure diversity and to ensure coinciding parts mesh?

T.6: Can neighbouring and global context of repeated material determine the details of how it is performed?

T.7: Can arrangement decisions be radically changed throughout the entire production process, including in a live eyes-free improvisational context?

T.8: Is placement of material against a metrical abstraction mediated by a system to take lead-ins into account?

T.9: When material is captured without correct metrical abstraction information, can it be added after the fact?

T.10: Do the metrical abstractions provide facilities to change tempo and time-signatures at any time?

T.11: Are producers able to manipulate material in terms of bars, beats, and sub-beats through provision of temporal editing and visualisation over metrical abstractions?

T.12: Are metrical structure abstractions provided, allowing producers to learn common parameters for expressive timing?

T.13: Can material be captured and refined through performance gesture?

Reuse and versioning abstraction

- R.1: Are individual repetitions editable, and do producers have control over how changes to one repetition affect others?
- R.2: Can references be maintained after derived material has been varied?
- R.3: Can varied repetitions captured through performance be represented as such?
- R.4: Can producers capture a snapshot version of any block of material that they can recall later?
- R.5: Can different versions of all types of data be maintained inside a single project file?
- R.6: Can producers work with the same data across multiple applications?
- R.7: Can changes be propagated across multiple arrangements when required?
- R.8: Is there support for managing multiple takes as a single unit, including takes of automation?
- R.9: Does the system allow producers to easily track dependencies between material and the various projects it is used in?
- R.10: Can the producer archive and reuse all types of material abstractions from past projects, while retaining links to their previous contexts?
- R.11: Can the producer access the local library content from multiple project files, including contextual library material and larger mini-arrangement style material?
- R.12: Can the producer create local library content without adding complexity to the composition?

R.13: Is there adequate support for rapid naming, searching, and name-versioning of all types of data?

R.14: Can material be located spatially and visually rather than solely by name?

Appendix D

Additional sources for figures

- **Figure 2.2:**

`http://plaza.ufl.edu/berticus/projects/project2/electronicmusic.html`

- **Figure 2.3:**

`http://www.vintagesynth.org/index.html`

- **Figure 2.4:**

`http://www.digitalmusicworld.com/html/hardware/Samplers/BriefHistory2.php`

- **Figure 2.5:**

`http://www.ghservices.com/gregh/fairligh/`

- **Figure 2.6:**

`Fromhttp://www.satoworks.com/MasterClass/`

- **Figure 2.7:**

`Fromhttp://www.satoworks.com/MasterClass/`

- **Figure 2.11:**

http://www.propellerheads.se/products/reason/index.cfm?fuseaction=get_popup&article=pop_combinator

- **Figure 2.12:**

<http://w1.867.telia.com/~u86705274/pages/code-apps.html>

- **Figure 3.3:**

<http://www.studer.ru/images/j37large.jpg>

- **Figure 3.4:**

From <http://www.ko-on.co.jp/15fader/images/fader.jpg>

- **Figure 3.5:**

<http://www.digidesign.com>

- **Figure 3.6:**

<http://www.soundonsound.com/sos/dec04/articles/logic7.htm>

- **Figure 3.7:**

From the Cubase manual [128].

- **Figure 3.8:**

From the Logic manual [6].

- **Figure 3.9:**

From the Logic manual [6].

- **Figure 3.10:**

From the Cubase manual [128].

- **Figure 3.11:**

<http://tamw.atari-users.net/notat1.gif>

- **Figure 3.12:**

From the Live manual [3].

- **Figure 3.16:**

<http://createdigitalmusic.com/2005/04/17/>

- **Figure 3.18:**

From the Cubase manual [128].

- **Figure 8.1:**

From the Live manual [3].

- **Figure 8.6:**

<http://tamw.atari-users.net/notator.htm>

- **Figure 8.7:**

From the Logic manual [6].

- **Figure 10.9:**

<http://www.macworld.com/2007/08/firstlooks/flimovie/index.php>

Bibliography

- [1] ACM SIGCHI curricula for human-computer interaction. Tech. rep., New York, NY, USA, 1992. Chairman-Thomas T. Hewett.
- [2] ABELSON, H., SUSSMAN, G. J., AND SUSSMAN, J. *Structure and Interpretation of Computer Programs*, 2nd edition ed. MIT Press, 1993.
- [3] ABLETON AG. *Ableton Live Version 6 Reference Manual*. Ableton AG, 2006.
- [4] ABRAMS, S., BELLOFATTO, R., FUHRER, R., OPPENHEIM, D., WRIGHT, J., BOULANGER, R., LEONARD, N., MASH, D., RENDISH, M., AND SMITH, J. Qsketcher: an environment for composing music for film. In *Proceedings of the Fourth Conference on Creativity & Cognition* (2002), ACM Press, pp. 157–164.
- [5] ALEXANDER, C., ISHIKAWA, S., SILVERSTEIN, M., JACOBSON, M., FIKSDAHL-KING, I., AND ANGEL, S. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.
- [6] APPLE COMPUTER, INC. *Logic Express 7 Reference Manual*. Apple Computer, Inc.
- [7] APPLE COMPUTER, INC. Logic pro 7. <http://www.apple.com/logic/>.
- [8] APPLE COMPUTER, INC. *Macintosh Human Interface Guidelines*. Addison-Wesley, 1992.

- [9] ATES, K., KUKLUK, J., HOLDER, L., COOK, D., AND ZHANG, K. Graph grammar induction on structural data for visual programming. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 232–242.
- [10] AVID TECHNOLOGY, I. Pro Tools—HD 3 Accel Systems, September 2007.
- [11] AVID TECHNOLOGY, INC. Digidesign, 2004. <http://www.digidesign.com>.
- [12] AVID TECHNOLOGY, INC. *Pro Tools Reference Guide*, version 7.3 ed. Avid Technology, Inc, 2007.
- [13] BANNON, L., AND BØDKER, S. *Designing Interaction: Psychology at the Human Computer Interface*. Cambridge University Press, Cambridge, 1991, ch. Beyond the interface: Encountering artifacts in use.
- [14] BARR, P. User-interface metaphors in theory and practice. Master's thesis, Victoria University of Wellington, 2003.
- [15] BERTIN, J. *Semiology of Graphics: Diagrams, Networks, Maps*, notation ed. University of Wisconsin Press, 1967/83. W. J. Berg, Translation.
- [16] BIDDLE, R. L., AND TEMPERO, E. D. Understanding the impact of language features on reusability. In *ICSR '96: Proceedings of the 4th International Conference on Software Reuse* (Washington, DC, USA, 1996), IEEE Computer Society, p. 52.
- [17] BLACKWELL, A. Pictorial representation and metaphor in visual language design. *Journal of Visual Languages and Computing* 12, 3 (June 2001), 223–252.

- [18] BLACKWELL, A. See what you need: Helping end-users to build abstractions. *Journal of Visual Languages and Computing* 12, 5 (October 2001), 475–499.
- [19] BLACKWELL, A., AND COLLINS, N. The programming language as a musical instrument. In *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2005)* (Brighton, United Kingdom, 2005).
- [20] BLACKWELL, A., AND GREEN, T. A cognitive dimensions questionnaire, 2000. <http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDquestionnaire.pdf>.
- [21] BLACKWELL, A., AND GREEN, T. Notational systems - the cognitive dimensions of notations framework. In *HCI Models, Theories and Frameworks: Toward a Multidisciplinary Science*, J. M. Carroll, Ed. Morgan Kaufmann, 2003.
- [22] BLACKWELL, A., AND GREEN, T. R. G. A cognitive dimensions questionnaire optimised for users. In *Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group* (2000), A. Blackwell and E. Bilotta, Eds., pp. 137–152.
- [23] BLACKWELL, A., GREEN, T. R. G., AND NUNN, D. J. E. Cognitive dimensions and musical notation systems. CMC 2000, Berlin: Workshop on Notation and Music Information Retrieval in the Computer Age., 2000.
- [24] BODAMER, J. Die prosop-agnosie. *European Archives of Psychiatry and Clinical Neuroscience* 179, 1–2 (January 1947), 6–53.
- [25] BØDKER, S. *Through the Interface*. Lawrence Erlbaum Associates, 1991.

- [26] BOULANGER, R. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming*. MIT Press, Cambridge, 1999.
- [27] BREGMAN, A. S. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990.
- [28] BROOKS, K. Programming narrative. In *Proceedings of the IEEE Symposium on Visual Languages* (1997).
- [29] BROWN, A., AND SORENSEN, A. Introducing JMusic. In *Proceedings Australasian Computer Music Conference* (Brisbane, Australia, 2000), A. Brown and R. Wilding, Eds., QUT, pp. 68–76.
- [30] BROWN, J. Activity theory: Its use and potential in the field of human-computer interaction. PhD Comprehensive, Psychology Dept, Carleton University, Ottawa, Canada, July 2004.
- [31] BURNETT, M. *Encyclopaedia of Electrical and Electronics Engineering*. John Wiley and Sons Inc., New York, 1999, ch. Visual programming.
- [32] BURROWS, J. B. Musical archetypes and collective consciousness: Cognitive distribution and free improvisation. *Critical Studies in Improvisation* 1, 1 (2004).
- [33] BUXTON, W. The use of hierarchy and instance in a data structure for computer music. *Computer Music Journal* 2, 4 (1978).
- [34] BUXTON, W. Chunking and phrasing and the design of human-computer dialogues. In *Proceedings of the IFIP World Computer Congress* (Dublin, Ireland, 1986), pp. 475–480.
- [35] BUXTON, W., SNIDERMAN, R., REEVES, W., PATEL, D., AND BAECKER, R. *The Evolution of the SSSP Score Editing Tools*. MIT Press, Cambridge MA, 1985, pp. 376–402.

- [36] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B. *Readings in Information Visualisation: Using Vision to Think*. Morgan Kaufmann, 1999.
- [37] CARROLL, J. M., AND MACK, R. L. Learning to use a word processor: By doing, by thinking, and by knowing. In *Readings in Human-Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, Eds. Morgan Kaufmann Publishers, Inc., 1995, pp. 698–717.
- [38] CHANAN, M. *Repeated Takes: A Short History of Recording and its Effects on Music*. Verso, London, 1995.
- [39] CHOMSKY, N. Three models for the description of language. *IRE Transactions on Information Theory* 2, 2 (1956), 113–123.
- [40] CHOU, H.-T., AND KIM, W. A unifying framework for version control in a cad environment. In *VLDB'86 Twelfth International Conference on Very Large Data Bases, August 25-28, 1986, Kyoto, Japan, Proceedings* (1986), W. W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, Eds., Morgan Kaufmann, pp. 336–344.
- [41] CLARKE, S., AND BECKER, C. Using the cognitive dimensions framework to evaluate the usability of a class library. In *Proceedings of the Fifteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2003)* (2003).
- [42] CLUTS, M. M. The evolution of artifacts in cooperative work: Constructing meaning through activity. In *GROUP '03: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work* (New York, NY, USA, 2003), ACM Press, pp. 144–152.
- [43] COKER, J. *Elements of the Jazz Language for the Developing Improvisor*. Warner Brothers, 1997.

- [44] COLLINS, N., MCLEAN, A., ROHRHUBER, J., AND WARD, A. Live coding techniques for laptop performance. *Organised Sound* 8, 3 (2003), 321–330.
- [45] COLLINS-SUSSMAN, B., FITZPATRICK, B. W., AND PILATO, M. *Version Control with Subversion*. O'Reilly Media, Inc., 2004.
- [46] CONSTANTINE, L. L., AND LOCKWOOD, L. A. D. *Software for Use: A practical Guide to the Models and Methos of Usage-Centered Design*. ACM Press, New York, 1999.
- [47] CRAFTON, D. *The Talkies: American Cinema's Transition to Sound, 1926–1936*. No. 4 in History of the American Cinema. Charles Scribner's Sons, New York, 1997.
- [48] CRAIK, K. *The Nature of Explanation*. Cambridge University Press, 1943.
- [49] CRESWELL, J. W. *Qualitative Inquiry and Research Design: Choosing Among Five Traditions*. Sage, 1998.
- [50] CROSS, I. AI and music perception. *AISB Quarterly*, 102 (1999), 12–25.
- [51] DANNENBERG, R. Music representation issues, techniques, and systems. *Computer Music Journal* 17, 3 (1993).
- [52] DANNENBERG, R., RUBINE, D., AND NEUENDORFFER, T. The resource-instance model of music representation. In *Proceedings of the 1991 International Computer Music Conference* (October 1991), pp. 428–432.
- [53] DARRAGH, J. J., WITTEN, I. H., AND JAMES, M. L. The reactive keyboard: A predictive typing aid. *IEEE Computer* 23, 11 (1990), 44–49.

- [54] DE ROURE, D. C., CRUICKSHANK, D. G., MICHAELIDES, D. T., PAGE, K. R., AND WEAL, M. J. On hyperstructure and musical structure. In *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia* (2002), ACM Press.
- [55] DiSESSA, A. *Models of Computation*. Lawrence Erlbaum Associates, 1986, ch. Cognitive engineering, pp. 31–61.
- [56] DUIGNAN, M. Cudas: The new class of composition sequencer. Tech report, Victoria University of Wellington, Wellington, New Zealand, 2004.
- [57] DUIGNAN, M., NOBLE, J., BARR, P., AND BIDDLE, R. Metaphors for electronic music production in reason and live. In *6th Asia-Pacific Conference on Computer-Human Interaction* (2004).
- [58] DUIGNAN, M., NOBLE, J., AND BIDDLE, R. A taxonomy of sequencer user-interfaces. In *Proceedings of the 2005 International Computer Music Conference* (Barcelona, Spain, September 2005), pp. 725–728.
- [59] DUIGNAN, M., NOBLE, J., AND BIDDLE, R. Activity theory for design: From checklist to interview. In *Proceedings of Human Work and Interaction Design* (Madeira, Portugal, 2006).
- [60] DUIGNAN, P. Strategic evaluation manifesto. In *6th European Evaluation Society Biennial Conference* (2004).
- [61] EMMERSON, S., Ed. *The Language of Electroacoustic Music*. Palgrave, 1986.
- [62] ENGSTRÖM, Y. Activity theory as a framework for analyzing and redesigning work. *Ergonomics* 43, 7 (July 2000), 960–74.
- [63] FAGERHEIM, P. Activity theory in jazz research. <http://musikk.hinesna.no/Ansatte/Fagerheim/Paal%20Fagerheim%20art..htm>.

- [64] FIELD, T. Policy-gradient learning for motor control. M.sc., Victoria University of Wellington, Wellington, New Zealand, 2003.
- [65] FJELD, M., MORF, M., AND KRUEGER, H. Activity theory and the practice of design: evaluation of a collaborative tangible user interface. *International Journal of Human Resources Development and Management* 4, 1 (2004), 94–116.
- [66] FOWLER, M., BACK, K., BRANT, J., OPDYKE, W., AND ROBERTS, D. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.
- [67] FRIBERG, A., AND SUNDSTRÖM, A. Swing ratios and ensemble timing in jazz performance: Evidence for a common rhythmic pattern. *Music Perception* 19, 3 (Spring 2002), 333–349.
- [68] GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.
- [69] GOLIN, E. J., AND REISS, S. P. The specification of visual language syntax. *Visual Languages, 1989., IEEE Workshop on* (1989), 105–110.
- [70] GRAY, W. D., AND SABNANI, H. Why you can't program your VCR, or, predicting errors and performance with production system models of display-based action. In *Conference Companion on Human Factors in Computing Systems* (1994), ACM Press, pp. 79–80.
- [71] GREEN, T., AND BLACKWELL, A. Cognitive dimensions of information artefacts: a tutorial, 1998. <http://www.ndirect.co.uk/~thomas.green/workStuff/Papers/>.
- [72] GREEN, T. R. G. Instructions and descriptions: Some cognitive aspects of programming and similar activities. In *Proceedings of the*

- Working Conference on Advanced Visual Interfaces*. ACM Press, 2000, pp. 21–28.
- [73] GROVE, G., AND SADIE, S. *The New Grove Dictionary of Music and Musicians*, 2nd edition ed. Macmillan, London, 2001.
- [74] HARRIS, M., SMAILL, A., AND WIGGINS, G. Representing music symbolically. In *IX Colloquio di Informatica Musicale* (Genoa, Italy, 1993).
- [75] HILL, P., HOLLAND, S., AND LANEY, R. C. Applying aspect oriented programming to music computing. In *Proceedings of Sound and Music Computing '04* (2004).
- [76] HINDLEY, G. *Music and Technology in the Twentieth Century*. The Johns Hopkins University Press, Baltimore, MD, 2002, ch. Keyboards, Crankshafts, and Communication: The Musical Mindset of Western Technology, pp. 33–42.
- [77] HOCKER, J. *Music and Technology in the Twentieth Century*. The Johns Hopkins University Press, Baltimore, MD, 2002, ch. My Soul is in the Machine - Conlon Nancarrow - Composer for Player Piano - Precursor of Computer Music, pp. 84–96.
- [78] HOLMES, T. B. *Electronic and Experimental Music*. Charles Scribner's Sons, New York, 1985.
- [79] HOWELL, P., WEST, R., AND CROSS, I. *Representing Musical Structure*. Academic Press, London, 1991, ch. Musical Structure and Knowledge Representation.
- [80] HUME, D. *A Treatise on Human Nature*. Longmans, Green, and Co., London, 1882.
- [81] IRESTIG, M., ERIKSSON, H., AND TIMPKA, T. The impact of participation in information system design: A comparison of contextual

- placements. In *PDC 04: Proceedings of the Eighth Conference on Participatory Design* (New York, NY, USA, 2004), ACM Press, pp. 102–111.
- [82] JOHNSON, J., ROBERTS, T. L., VERPLANK, W., SMITH, D. C., IRBY, C. H., BEARD, M., AND MACKEY, K. The Xerox Star: A retrospective. In *Readings in Human-Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grundin, W. A. S. Buxton, and S. Greenberg, Eds., second ed. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1995.
- [83] JOHNSON, S. *Interface Culture: How New Technology Transforms the Way We Create and Communicate*. Harper San Francisco, 1997.
- [84] JOHNSON-LAIRD, P. *Mental Models*. Cambridge University Press, 1983.
- [85] KAPTELININ, V., NARDI, B. A., AND MACAULAY, C. Methods & tools: The activity checklist: a tool for representing the “space” of context. *Interactions* 6, 4 (1999), 27–39.
- [86] KAY, A. Doing with images makes symbols: Communicating with computers. Video, 1987. Apple Computer, Inc.
- [87] KIRN, P. How do you set up your ableton live sets for performance / DJing? <http://createdigitalmusic.com/2006/10/18/>.
- [88] KRUMHANSL, C. L. Music psychology: Tonal structures in perception and memory. *Annual Review of Psychology* (1991), 277–303.
- [89] KUUSKANKARE, M., AND LAURSON, M. Recent developments in ENP-score-notation. In *Proceedings of Sound and Music Computing '04* (2004).
- [90] LAKOFF, G., AND JOHNSON, M. *Metaphors We Live By*. University of Chicago Press, Chicago, 1980.

- [91] LAKOFF, G., AND JOHNSON, M. *Philosophy in the Flesh*. Basic Books, New York, 1999.
- [92] LEE, C. S. *Representing Musical Structure*. Academic Press, London, 1991, ch. The Perception of Metrical Structure: Experimental Evidence and a Model.
- [93] LEONT'EV, A. *Activity, Consciousness and Personality*. Prentice Hall, Englewood Cliffs, NJ, 1978.
- [94] LERDAHL, F. *Tonal Pitch Space*. Oxford University Press, 2001.
- [95] LERDAHL, F., AND JACKENDOFF, R. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachusetts, 1983.
- [96] LEVITIN, D. J. *This is Your Brain on Music: The Science of a Human Obsession*. Penguin Group, 2006.
- [97] LEW, M. Live cinema: Designing an instrument for cinema editing as a live performance. In *Proceedings of the 4th Conference on New Interfaces for Musical Expression (NIME 04)* (Hamamatsu, Japan, 2004).
- [98] LOCKE, J. *An Essay Concerning Human Understanding*. 1690.
- [99] MCCARTNEY, J. Rethinking the computer music language: Super-Collider. *Computer Music Journal* 26, 8 (December 2002), 61–68.
- [100] MCCULLOUGH, M. *Abstracting Craft: The Practiced Digital Hand*. The MIT Press, London, England, 1996.
- [101] MERRIAM, S. B. *Case study research in education: A qualitative approach*. Jossey-Bass, 1988.
- [102] MICROSOFT CORPORATION. *The Windows Interface Guidelines for Software Design: An Application Design Guide*. Microsoft Press, 1995.

- [103] MIRANDA, E. R. *Composing Music with Computers*. Focal Press, Oxford, 2001.
- [104] MITCHELL, W. J. *The Logic of Architecture: Design, Computation, and Cognition*. MIT Press, Cambridge MA, 1990.
- [105] NARDI, B. A. *A Small Matter of Programming: Perspectives on End-User Computing*. MIT Press, 1993.
- [106] NARDI, B. A., Ed. *Context and Consciousness*. MIT Press, Cambridge, Massachusetts, 1996.
- [107] NARDI, B. A., AND ZARMER, C. L. Beyond models and metaphors: Visual formalisms in user interface design. *Journal of Visual Languages and Computing* 4, 5–33 (1993).
- [108] NEWCOMB, S. Standard music description language complies with hypermedia standard. *IEEE Computer* 24, 7 (July 1991).
- [109] NOBLE, J., TAIVALSAARI, A., AND MOORE, I., Eds. *Prototype-Based Programming: Concepts, Languages, and Applications*. Springer-Verlag, 1999.
- [110] NORMAN, D. *User-Centred System Design*. Lawrence Erlbaum Associates, 1986, ch. Cognitive Engineering, pp. 31–61.
- [111] NORMAN, D. A. *The Invisible Computer*. The MIT Press, 1998.
- [112] NYMAN, M. *Experimental Music: Cage and Beyond*, second ed. Cambridge University Press, Cambridge, 1999.
- [113] OCKELFORD, A. *Representing Musical Structure*. Academic Press, London, 1991, ch. The Role of Repetition in Perceived Musical Structures.
- [114] PARNAS, D. L. On the criteria to be used in decomposing systems into modules. *Commun. ACM* 15, 12 (1972), 1053–1058.

- [115] PAYNE, J. The software cronicles. EQ Magazine, March 2006. <http://www.eqmag.com/story.asp?sectioncode=36&storycode=13747>.
- [116] PENNYCOOK, B. W. Computer-music interfaces: a survey. *ACM Comput. Surv.* 17, 2 (1985), 267–289.
- [117] PHON-AMNUAISUK, S. Logical representation of musical concepts (for analysis and composition tasks using computers). In *Proceedings of Sound and Music Computing '04* (2004).
- [118] PLATO. *The Republic*, vol. 150. Project Gutenberg, Champaign, IL, 1994.
- [119] POLANSKY, L., AND ROSENBOOM, D. HMSL (hiearchical music specification language): A real-time environment for formal, perceptual and compositional experimentaion. In *Proceedings of the International Computer Music Conference* (San Francisco, 1985), International Computer Music Association, pp. 243–250.
- [120] POPE, S. T. Music notations and the representation of musical structure and knowledge. *Perspectives of New Music* 24 (1986).
- [121] POPE, S. T. *Squeak: Open Personal Computing and Multimedia*. Prentice Hall PTR, 2001, ch. Music and Sound Processing in Squeak Using Siren.
- [122] PUCKETTE, M. Max at seventeen. *Computer Music Journal* 26, 4 (December 2002), 31–43.
- [123] RICHARDS, I. A. *The Philosophy of Rhetoric*. Oxford University Press, 1936.
- [124] SATO, T. SIGGRAPH: Maya master class, 2005. <http://www.satoworks.com/masterclasspage.html>.

- [125] SHNEIDERMAN, B. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *IUI '97: Proceedings of the 2nd International Conference on Intelligent User Interfaces* (1997), ACM Press, pp. 33–39.
- [126] SHNEIDERMAN, B. Creating creativity: User interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction (TOHI)* 7, 1 (2000), 114–138.
- [127] SMAILL, A., WIGGINS, G., AND HARRIS, M. Hierarchical music representation for composition and analysis. *Computing and the Humanities Journal* (1993).
- [128] STEINBERG MEDIA TECHNOLOGIES GMBH. *Operation Manual: Cubase 4: Music Creation and Production System*, 2006.
- [129] STONE, K. *Music Notation in the Twentieth Century : A Practical Guidebook*. W. W. Norton, New York, 1980.
- [130] SUCHMAN, L. *Plans and Situated Action*. Cambridge University Press, New York, 1987.
- [131] TAIVALSAARI, A. *Prototype-Based Programming: Concepts, Languages, and Applications*. Springer-Verlag, 1999, ch. Classes vs. Prototypes: Some Philosophical and Historical Observations.
- [132] THE IDEAS WEB LTD. Doview. <http://www.doview.com>.
- [133] THÉBERGE, P. *Any Sound You Can Imagine: Making Music / Consuming Technology*. Wesleyan University Press, 1997.
- [134] THÉBERGE, P. The network studio: Historical and technological paths to a new ideal in music making. *Social Studies of Science* 34, 5 (October 2004), 759–781.

- [135] TIDWEL, J. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Inc., 2005.
- [136] TOOMIM, M., BEGEL, A., AND GRAHAM, S. L. Managing duplicated code with linked editing. In *VLHCC '04: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 173–180.
- [137] TRUAX, B. *The Language of Electroacoustic Music*. Palgrave, 1986, ch. Computer Music Language Design and the Composing Process.
- [138] TUKIAINEN, M. Evaluation of the cognitive dimensions questionnaire and some thoughts about the cognitive dimensions of spreadsheet calculation. In *Proceedings of the Thirteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2001)* (2001), G. Kadoda, Ed., pp. 291–301.
- [139] TURNER, P., AND TURNER, S. A web of contradictions. *Interacting with Computers* 1, 14 (2001).
- [140] VON EHRENFELS, C. Über gestaltqualitäten. *Vierteljahresschrift für Wissenschaftliche Philosophie* 14 (1890), 242–92.
- [141] VYGOTSKII, L. *Mind in Society : The Development of Higher Psychological Processes*. Harvard University Press, 1978.
- [142] WANG, G., AND COOK, P. R. Chuck: A concurrent, on-the-fly, audio programming language. In *Proceedings of the 2003 International Computer Music Conference* (2003).
- [143] WEINSTEIN, M. TAMS analyzer. <http://tamsys.sourceforge.net/>.
- [144] WIKIMEDIA COMMONS. Wikimedia commons, 2007. <http://commons.wikimedia.org>.

- [145] WINKLER, T. *Composing Interactive Music: Techniques and Ideas Using Max*. MIT Press, 1998.
- [146] WITTGENSTEIN, L. *Philosophical Investigations*, 3rd edition ed. Prentice Hall, 1999.
- [147] YANG, S., BURNETT, M., DEKOVEN, E., AND ZLOOF, M. Representation design benchmarks: A design-time aid for VPL navigable static representations. *Journal of Visual Languages and Computing* 8 (1997), 563–599.
- [148] YIN, R. K. *Case study research: Design and Method*. Sage, Newbury Park, CA, 1989.