THEMA: A MUSIC NOTATION SOFTWARE PACKAGE WITH INTEGRATED AND AUTOMATIC DATA COLLECTION

Peter McCulloch
New York University
peter.mcculloch@nyu.edu

ABSTRACT

This paper introduces Thema, a custom music notation software environment designed to automatically and transparently capture quantitative data into a relational database. The majority of research into musical creativity is qualitative in nature, and this software addresses several areas, such as search and improvisational data, which are difficult to study with current qualitative methods. Thema's database provides advantages over ad hoc file collection mechanisms by providing integrated search; the software also is able to consistently identify musical material via automatically assigned identification codes, and this provides a useful supplement to content-based search. In 2013, a study was conducted of ten graduate-level composers using Thema, and the dataset from this study was used to develop new analytical tools for examining compositional data.

1. INTRODUCTION

Until recently, most research into the compositional process of adult composers has been conducted using qualitative methodologies. Creativity is complex and researchers have rightly appreciated the role that composers play in illuminating their creative process. A variety of techniques have been used by researchers and composers including interviews, verbal protocol, sketch studies, and journals. With the commodification of recording technology, so-called real-time studies of compositional process have become more common. These typically involve audio or video recordings of the composer. To provide insight into the composer's thought process, verbal protocol techniques are often used, either concurrently with composition or retrospectively. This data may then be triangulated with versioned musical sketches or computer files and supplemented by journals and other documentation. While combining multiple sources gives relatively good coverage of activity, utilizes the composers' personal insights, and allows composers to work with familiar tools, it also requires effort on the part of the composer and the researcher to collect and organize data. [1, p. 246-7]. In the short run, this is certainly manageable, but it is difficult to scale these techniques up to larger studies, and, as might be expected, longitudinal studies are rare in the qualitative literature concerning adult composers, as are studies featuring large sample sizes. [1]

Musical informatics have proved useful for addressing questions that concern a large amount of music, and it stands to reason that they could be of some aid in the study of compositional process. While computerized analysis is unlikely to bring the same type of insight that a qualitative study can, it has strengths in complementary areas: namely, it can be pursued over time and at scale because the data can be automatically analyzed. Anecdotal accounts from composers in the process literature suggest that composers benefit from participating in these studies but most composers will never have the opportunity to do so. Quantitative analytical tools could allow composers to systematically examine their own music and behavior on an ongoing basis without a dedicated researcher.

It is increasingly common for composers to use software in the act of composing music rather than purely for typesetting. Though the impacts of this trend are subject to debate, it seems unlikely to change in the near future. The software that composers use to create music is uniquely suited to observing quantitative data about the creative process since it can see not only the document, but also the composer's interactions with the document. Several studies in the literature have used dedicated software to observe compositional behavior [4–6]; these have not, however, utilized Common Music Notation. Other studies have observed composers at work via automatic screen captures or screen recording [7–9]. Peterson's 2008 study presents quantitative results, but the data was acquired through the manual analysis of automatically collected screen captures.

[1] Collins [2] and Donin [3] provide useful reviews of the literature.

2. PREVIOUS AND RELATED WORK

A few projects have used custom software to study compositional process. Otto Laske and Barry Truax used the Observer I program to observe children making music with a synthesizer [4]. Maude Hickey wrote a program to study creativity in children [5]. The QSketcher research project at IBM developed an environment for film music composition that automatically recorded and organized improvisational material, and maintained a persistent view of its environment. [10]. Recently, Chris Nash created reViSiT, a free "tracker"-style audio plugin that captured usage data [6]. He used the software to collect data from over 1,000 musicians; his is by far the largest sample size in this area, but his data set does not include the music that his subjects created.

In considering the use of quantitative data for studying compositional process, it is helpful to consider existing infrastructure for storing data. Standardized open file formats have been a boon to music informatics researchers as they allow musical data to be examined independently of the program that created the data. This is helpful since designing music notation software is a time-intensive task, and standard file formats provide some degree of interoperability between programs. At present, however, there is no standard for how a music notation program should operate, and that also means that standard music file formats such as MusicXML include very little, if any, information outside that already present in the score such as how the composer is using the program or improvisational MIDI data. Additionally, our ability to understand how multiple versions of a score are related depends on effective comparison algorithms. Though comparison works for simple additions and deletions, it becomes less useful as the composer's actions become more dispersed across time.

As an alternative to an ad hoc approach to data collection which combines multiple data formats such as MusicXML and SMF and exists separately from the music software, there is a good argument to be made for an integrated data collection system which operates from within the software used to create the music. Such a system would have a greater understanding of how its constituent elements relate and would be able to integrate the information it collects into its operation. This capability could also be materially useful to composers, particularly in its ability to bridge the gap between improvisational development and transcription. There are certainly drawbacks to such a system, most notably, in that it ties the composer's work to a particular software. Nevertheless, it allows access to data which is not well-served by existing methods; this data may provide insight outside of that available in score and MIDI data.

3. THEMA

Thema is a music notation software environment, written using the Java Music Specification Language's JScore package [11], that has been purpose-built for automatic data collection. On the surface, Thema is designed to operate in a manner similar to existing music notation software. Note entry occurs in step-entry mode, optionally with MIDI input, and most functions in the interface are available via keyboard shortcuts, including score playback. Selections may be made with the mouse, and a variety of commands such as transposition and clipboard operations are available. Thema has two different playback modes: cursor and selection; cursor mode plays the score from the current cursor location, while selection mode plays back only the selected material. Like ENP [12], Thema can make use of non-contiguous selections in the score, and this includes playback operations. Thema's musical representation is relatively limited relative to other software in the field such as Bach [13] and ENP. It cannot, for instance, represent nested tuplets, and it does not support breakpoint function notation or proportional notation in editing.

Thema, like JScore, allows the user to operate on rhythm in relative units via doubling and halving operations, and this works for both selections as well as for setting the cursor's duration. This latter aspect is useful in that it does not require the composer to remember specific key mappings for durations, though those are also available. In step-entry mode, the user may also select a rhythmic motive and then use its durations and ties as a source sequence for a new stream of notes, as shown in figure 1. This simplifies the entry of repetitive figures such as dotted eighth and sixteenth note pairs. Additionally, the composer may advance to an arbitrary position within the sequence or reset to the beginning, and this allows for greater flexibility in applying the current sequence without requiring changes to its content; this can be useful, for example, in creating rhythmic ostinato figures in fluctuating meters.

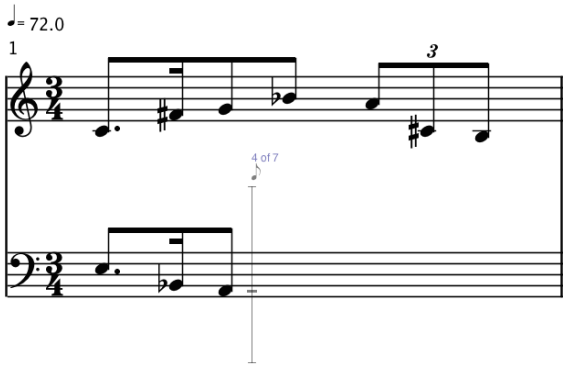

Figure 1. Step entry with rhythmic sequence

3.1 Storage

By design, Thema focuses on automatically and transparently capturing low-level data at a fine time granularity. Where possible, data is stored as it is captured, and all entries are stored with time stamps at millisecond resolution. In order to enable frequent storage, Thema models the score, as well as the state of the program, as a collection of tables within a relational database, and only stores material that has changed over subsequent states. The database maintains previous versions of edited items so that all past states of an item are reachable. In order to easily identify multiple versions of an item across time, each item is tagged with a unique, automatically generated permanent identifier which is consistent across all versions of the item.

Storage in Thema is tightly integrated into to the workings of the program, and it occurs as the result of actions by the composer, instead of at an arbitrary time interval. This makes it easier to discern the composer's actions in the data stream, since any entry in the database is present as the direct result of the composer's actions; it also prevents multiple actions from being condensed into a single entry, as might occur when saving at a particular time interval. A command identifies elements which may have been intentionally changed, and the program proceeds to discern whether those elements were actually changed, as well as any changes to surrounding elements which may have occurred as a result. For example, changing the duration of the first note in the measure has the effect of changing the starting times of subsequent notes within the measure.

3.2 Data Representation

Thema represents the structure of the score in a straightforward, hierarchical fashion: a score contains measures, measures contain staves, staves contain tracks, and tracks contain notes. Following normalized database practices, objects within that hierarchy only reference the class of objects immediately above them in the hierarchy, with the exception that all objects maintain a reference to their containing score.[2] For example, when storing a note in the database, a reference is stored to its track's identifier, but not to its containing staff or measure. This insulates lower level objects from being affected by changes in higher-level objects such as the insertion of a measure earlier in the piece.

With a score changing over time, the amount of data that could be stored is potentially large. It would be inefficient to store the entire document for a small change, so Thema stores only objects that have changed. On load, it reassembles the score from the desired version. This is different than a `diff`-based approach because the state of the entire score across time is visible to search functions without the need for derivation. Accessing earlier versions of a score is as efficient as loading the current score, and it is simple to make comparisons across versions. When undoing commands or reverting to a previous state of the score, the state of the score at the desired moment is loaded from the database and then made current. A detailed description of this mechanism is provided in [14, p.85-109]; one interesting feature of this design is that though it appears to operate like a conventional undo-redo stack, all past states are accessible.

3.3 Processes

Processes are the primary unit of work within Thema. Any action that the composer initiates within the program creates a timestamped entry in the process table, and each entry represents a state in the score or the environment, with the exception of MIDI data, which is stored independently of process information. This distinction is made because the volume of MIDI data is considerable, and may happen in parallel with other actions. MIDI data is recorded with a time stamp, and may be combined with the process table data via an SQL join.

Thema separates processes into two categories: score and environment. Score processes alter the content of the score whereas environment actions, such as adjusting the viewable area, playing back the score, and making drag selections with the mouse, do not. With score processes, the program also records whether or not a command had any effect, e.g., the user attempting to transpose a rest or make a deletion when the cursor is in a blank measure. This allows the program to skip over commands which had no impact when undoing, and also makes it possible to remove empty operations from consideration for analysis.

3.4 Separation of Identity from State

Whether using qualitative or quantitative data, working with multiple versions of a single score often can lead to reference problems. The score is not static over time in these situations, but most of our musical terminology for labeling material depends on it being so. For example, a label such as "the first A5 in measure 7" is tied inextricably to its current state. If the note is transformed, or measures are added in front of it, it is no longer an accurate descriptor. Adding timing information makes it easier to find the particular material as it existed at the moment, but it does not provide a sense of its identity across time. Labeling systems may be used where the software supports it, but these depend on the composer or the researcher to main-

[2] A database can contain multiple scores, and by explicitly filtering based on the score, queries run an order of magnitude faster.

tain consistency, as indicated by David Collins notes in his long-term study of compositional process [1].

Rather than identifying material in the score based on mutable state, it is a better approach to use a permanent identifier that is decoupled from state, e.g., "note #1273". In relational database design, synthetic keys are preferred over natural keys for this same reason in that they preserve the unique identity of a row even when its values change. By using a synthetic key to identify notes and other objects, continuity across time is preserved. It also makes it simple to compare different versions of the same passage which are separated by a large amount of time. This does not preclude searches based on content or comparison, but it reduces dependency on comparison. It is worth noting, however, that this approach is only practical in a situation where the program automatically manages this information.

Thema identifies material by using unique identification numbers. Each object in the score has a unique, permanent identifier as well as a version number. The first value is static and identifies an object across time; it is also guaranteed to be unique across object classes, e.g., for a note with the identifier #1273, no other objects in the database will share the same identifier, even across scores. The second value indicates the specific version of the object within the database table; each version occupies a row in the table. The permanent identifier decouples an object's identity from its state which allows searches to be conducted on the basis of identity rather than musical content. Search based on comparison is certainly still possible, but it is not necessary in order to locate material across time. This is useful because identity-based searches will typically run several orders of magnitude more rapidly than comparison-based ones; for example, it is much simpler to find the history of a particular group of notes by searching for rows with corresponding identifiers than it is to compare thousands of iterations of a score.

3.5 Attribution and Context

When storing data, it is useful not only to be able to identify changed material but also to know how the changes were effected. For example, a "cut" operation is identical to a "clear" operation in terms of the difference between successive states in the score; in the case of a "cut" command, however, it is likely that the material may reemerge as the result of a "paste" command, and it is helpful to identify this relationship as it indicates a larger cognitive process. In this example, Thema will not only indicate the command type during storage, but it will also store a set of parent-child relationships between the source and destination materials so that the link between the two states of the score—however distant—is maintained. The software also records the context of the program, including the location of the cursor, any selections in the score, the current state of playback, and so forth.

Additionally, in storing objects, Thema makes a distinction between directly edited objects and objects that have changed state as a result of edits to other objects. For example, a change in the duration of a note appearing at the beginning of the measure would affect the state of subsequent notes; the first note would be considered to be directly edited, while the other notes would be marked as indirectly edited in the database. Similarly, deleting a measure causes the measure numbers of all subsequent measures to be decremented. By indicating the target of the operation, Thema reduces ripple effects in the data and provides a more accurate picture of the composer's actions.

3.6 Model Objects

Latency can be a challenge for object-oriented applications which use databases for storage. If an object changes before it is correctly stored, the values in the database could become inconsistent with the program values. At the same time, it is important that the user interface for the program is responsive to its user, so it is also impractical to delay processing user input while storage is occuring. To address this problem, Thema uses immutable data objects between the application logic and the database. For object in the score or the environment, the program maintains an immutable data model of each object's current state, e.g., a `Note` object contains a reference to a `NoteModel` object. [3] When an object may have changed as the result of a command, a new model is constructed and compared against the previous state; if different, the new model is added to the storage queue and replaces the previous model. Though the construction of models increases memory requirements, it also allows storage to safely proceed in a separate thread from the user interface, ensuring responsiveness while maintaining data integrity. Because the model objects cannot change once created, they may also be safely cached in memory in order to accelerate loading when undoing or redoing actions in the score. Each model has two fields titled `process` and `kill_process`. These fields serve to mark the lifespan of the specific model. Figure 3 and table 1 show an example of the lifecycle of models in Thema.

4. DATASET

In order to establish a dataset for studying compositional process data collected in Thema, a study was conducted at New York University with ten graduate-level composers creating piano pieces using the software. Subjects were

[3] For a useful discussion of the virtues of immutability, see [15].

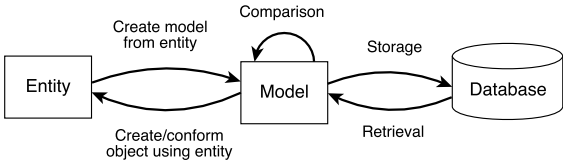

Figure 2. Entities, models, and the database

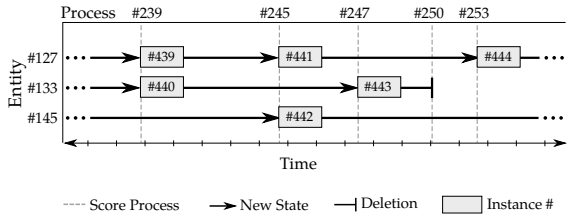

Figure 3. Timeline

introduced to the software via a tutorial session, and were then asked to notate a brief excerpt from Bartok's *Mikrokosmos*; the excerpt was selected because it would require users to perform a variety of tasks, handling time-signature changes, articulations, and multiple voices within a staff. Following this, the remainder of the four-hour session was spent composing a short piano piece for an intermediate-level performer in a style of the composer's choosing. This controlled setting ensured that participants had access to a full-size MIDI keyboard and were able to receive technical support if they had questions about the software. While this arrangement is not ideal from the standpoint of naturalism, it ensured that the data was captured reliably and that composers were able to use the program, and the knowledge gained will allow for future, less-restrictive studies to proceed. Though the composers had relatively little time to work with the software, all of the composers were able to complete the study. At the end of the session, the composers provided a segmentation of the work, indicating major sections, as well as any smaller subdivisions. The composers were compensated for their time and agreed to release their work and data under a Creative Commons license with the option of being attributed for their work or remaining anonymous. [4]

In addition to capturing quantitative data, Thema also recorded screen captures for every entry in the process table. The bounding-box coordinates for currently visible notes, dynamics, and articulations were also stored into a table in the database. This makes it possible to create graphical overlays on the score images without having to use the program itself, and provides a simple means of rapidly browsing through past states of the score. Nonlinear browsing and montaging can be realized via database queries, e.g., "select all activity within a two-minute window of a clipboard operation" or "show all versions of the selected passage."

[4] The terms of the license are available at `https://creativecommons.org/licenses/by-nc-sa/4.0/`

entity	*id*	*process*	*kill_process*
127	439	239	245
133	440	239	247
127	441	245	253
145	442	245	null
133	443	247	250
127	444	253	null

Table 1. Entity states from fig. 3 as represented in the database.

5. VISUALIZATIONS

Thema contains a suite of visualizations in a variety of formats for examining compositional data. These visualizations can be synchronized together via a central time slider. For example, one window might contain the score at the time indicated by the slider, while another window displays the structure of the score over the course of a two hour sliding window, and a third window contains a score which displays incoming pitches from the MIDI keyboard in a two minute window. Each window can also contain multiple graphical overlays, such as, for example, showing the location in the score and duration of playback superimposed on the long-term structural view. Thema also features an API for developing user-defined graphical overlays.

The score overlay section of the API allows programmers to access the drawing subroutines for notes in the score. Figure 4 shows a heat map of the composer's edit activity superimposed on the score.

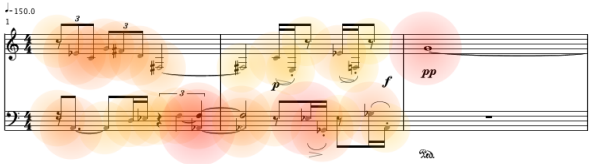

Figure 4. Heatmap Overlay

In figure 5, the arrows between pairs of notes indicate pairs of notes that were edited in close time proximity to each other. The weight of the line is proportional to the number of times they were edited as well as how closely in time they were edited, with simultaneous edits producing thicker lines. As can be seen, the notes in the first two measures are densely connected to each other; they are also connected to the previous (unseen) measure. The notes in the last measure, however, are not connected to the notes in the prior measures, indicating that they were never edited in close time proximity to the notes in the second measure.

This location is also one of the major boundaries in the score indicated by the composer.

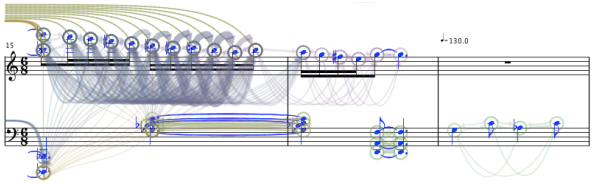

Figure 5. Edit Time Connections Overlay

Observations such as this inspired the development of a novel pitch-agnostic boundary detection algorithm, described in [14, p.183-190]. The algorithm operates on the premise that low-level musical boundaries parallel boundaries in the composer's process as represented by edit times; similarly, areas of musical continuity in the score are more likely to be closely related in terms of editing time. While the premise is naive, when tested against the boundaries indicated by the composers, the algorithm achieved surprisingly respectable results. Future work will address the algorithm and its parameters in depth, and compare the segments found via this method against those found by content-based algorithms.

6. CONCLUSIONS AND FUTURE WORK

The sample size from the NYU study is small, but it demonstrates that Thema can be an effective tool for research. More data in this area is needed, particularly in tandem with current qualitative methods. The planned public release of the software in the Fall of 2015 will allow composers to experiment with the program over time and in a naturalistic setting, with the option of sharing their data with researchers. Development is also underway on a wrapper written in Python to convert Thema's data into Music21 streams and this will provide access for other researchers in computational musicology.

Low-level musical behavior has not as yet received much attention in the compositional process literature, and it is hoped that this tool will provide a new means for studying it and, in so doing, allow compositional process research to connect to existing research in computational musicology.

Acknowledgments

Thanks to Robert Rowe, Juan Pablo Bello, Elizabeth Hoffman, Panayotis Mavromatis, and Sarah Marlowe for their help, as well as to the composers who gave of their time and talents in the study.

7. REFERENCES

[1] D. Collins, "Real-time tracking of the creative music composition process," *Digital Creativity*, vol. 18, no. 4, pp. 239–256, 2007.

[2] ——, "A synthesis process model of creative thinking in music composition," *Psychology of Music*, vol. 33, no. 2, pp. 193–216, 2005.

[3] N. Donin, "Empirical and historical musicologies of compositional processes: Towards a cross-fertilisation," in *The Act of Musical Composition : Studies in the Creative Process*, D. Collins, Ed. Ashgate Publishing, 2012, ch. 1, pp. 1–26.

[4] B. Truax, "For Otto Laske: A communicational approach to computer sound programs," *Journal of Music Theory*, vol. 20, no. 2, pp. 227–300, 1976.

[5] M. Hickey, "Qualitative and quantitative relationships between children's creative musical thinking processes and products," Ph.D. dissertation, Northwestern University, 1995.

[6] C. Nash, "Supporting virtuosity and flow in computer music," Ph.D. dissertation, University of Cambridge, 2012.

[7] J. Peterson and E. Schubert, "Music notation software: Some observations on its effects on composer creativity," *Proceedings of ICoMCS December*, p. 127, 2007.

[8] J. Peterson, "Computer notation-based music composition and the delayed introduction of musical expression markings," *Journal of Education, Informatics and Cybernetics*, vol. 1, no. 3, pp. 37–41, 2008.

[9] B. Eaglestone, N. Ford, G. J. Brown, and A. Moore, "Information systems and creativity: an empirical study," *Journal of Documentation*, vol. 63, no. 4, pp. 443–464, 2007.

[10] S. Abrams, R. Bellofatto, R. Fuhrer, D. Oppenheim, J. Wright, R. Boulanger, N. Leonard, D. Mash, M. Rendish, and J. Smith, "QSketcher: an environment for composing music for film," in *International Computer Music Conference*, 2001.

[11] N. Didkovsky and P. L. Burk, "Java music specification language, an introduction and overview," in *Proceedings of the International Computer Music Conference (ICMC)*. Havana: International Computer Music Association, 2001, pp. 123–126.

[12] M. Kuuskankare and M. Laurson, "Expressive notation package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[13] A. Agostini and D. Ghisi, "Bach: an environment for computer-aided composition in max," in

Proceedings of the International Computer Music Conference. Ljubljana, Slovenia: ICMC, 2012, pp. 373–378. [Online]. Available: http://quod.lib.umich.edu/cgi/p/pod/dod-idx/bach-an-environment-for-computer-aided-composition-in-max.pdf?c=icmc;idno=bbp2372.2012.068

[14] P. McCulloch, "Thema: A software framework for the quantitative study of compositional process," Ph.D. dissertation, New York University, 2014.

[15] R. Hickey. (2009) Are we there yet? a deconstruction of object-oriented time. [Online]. Available: http://wiki.jvmlangsummit.com/images/a/ab/HickeyJVMSummit2009.pdf