The Media Materiality of Max: Rhizomatic Organology and the Genealogy of Abstractions

Jaehoon Choi
Rensselaer Polytechnic Institute
jsonchoi@jsonchoi.io

ABSTRACT

This paper analyzes the Max software environment through the lens of media materiality and digital organology, focusing on its design principles, musical implications, and theoretical foundations. Through close readings of foundational texts by Miller Puckette and David Zicarelli, the study examines Max's emphasis on real-time interaction, modularity, and process-oriented programming. Central to this inquiry is Zicarelli's notion of incompleteness, which positions Max not as a content-based system, but as an open platform for creative interconnection and abstraction. Drawing on theories from Umberto Eco, Deleuze and Guattari, and Thor Magnusson, the paper explores how Max maintains a rhizomatic organological paradigm internally, while simultaneously archiving abstracted genealogies of musical ideas across contemporary music. Ultimately, Max is presented as a paradigmatic example of how software's technical design both shapes and is shaped by broader artistic and cultural networks. This exploration contributes to media studies and music technology discourse by foregrounding how computation, when embedded in artistic systems, functions as both a material and cultural agent.

1. INTRODUCTION

Since its invention by Miller Puckette, Max has been critical in the history of computer music, and understanding its wide-ranging contributions requires a multifaceted research approach. Among the various avenues, this paper focuses on the media materiality of Max and analyzes its broader musical implications. To investigate this materiality, this research conducts a close reading of two articles—each by Miller Puckette and David Zicarelli—published in the *Computer Music Journal*, Volume 26, Number 4 (2002).[1, 2] Although over two decades old, these texts remain highly relevant because they articulate Max's foundational design principles from its creators' perspectives, and how they were engineered in the software. This reading serves as the foundation for the theoretical discussions presented in later sections.

Section 2 explores Max's design implementation, drawing primarily from Puckette's article and partially from Zicarelli's. Section 3 introduces Zicarelli's concept of *incompleteness*, which he identifies as Max's most defining characteristic, along with the role of abstraction in software design. Section 4 then examines how Max's materiality, as discussed earlier, and how its internal mechanism aligns with the notion of rhizomatic organology when implemented in digital software. Section 5 then analyzes how the rhizomatic nature of internal Max manifests genealogies of musical ideas externally, which are abstracted from the complex network of contemporary music scene, obtaining its own material agency. The paper concludes with reflections on the implications of these characteristics for media studies and digital musical instruments.

2. THE MAX PARADIGM

Puckette introduces the idea of the Max paradigm as "a way of combining pre-designed building blocks into configurations useful for real-time computer music performance," encompassing "a protocol for scheduling control- and audio-rate computations, an approach to modularization and component intercommunication, and a graphical representation and editor for patches."[1] Pure Data (Pd) and Max, the main subjects of this conference, are the most prominent applications of this paradigm.

Max paradigm, from its early works from Puckette, had a clear emphasis on real-time musicking, and they are realized through various design implementations. Inspired by Max Matthews' RTSKED, audio and control stays separate inside the Max environment. Here, the scheduling of events and interaction becomes a critical task, and Max achieves this through a trigger-based approach.[1] In Max, objects communicate via messages-"linear lists of atoms, which in turn may be either numbers or strings."[1] These messages are simple, standardized, and serve as both data carriers and triggers, allowing intuitive object-to-object communication.[1]

The simplistic approach to messages, and the unification of trigger and message, holds broader significance in Max. It enables Max to function like a musical instrument played by the musician. Puckette illustrates this with a metaphor of the piano:

> Max follows the piano metaphor also in that objects do not specify what they wait for, but instead they explicitly choose which other objects to trigger. In other words, the choice is up to the triggering object, not the triggered one (the player, not the piano).[1]

This clearly indicates that control in Max is push-based rather than event-listening or pull-based, resembling how a performer chooses which keys to press on a piano. In

this model, Max uses logical time as the basis for triggering actions.[1] It operates in a "purely deterministic way, irrespective of any real-time delays engendered by CPU load or the operating system."[1] These features enable Max to facilitate real-time interaction between various objects, functioning seamlessly and intuitively—much like a musical instrument—through its unique scheduling mechanisms.

This focus on real-time interaction and a musical instrument-like triggering mechanism is further reinforced by Max's emphasis on process over data.[1] To support this design, Max avoids using scoping or namespaces, meaning that all symbols and their associated bindings exist within a single flat space.[1] Additionally, despite certain drawbacks, the Max paradigm adopts a non-uniform and heterogeneous approach to data handling, which allows for greater flexibility and modularity.[1]

In sum, the Max paradigm places greater emphasis on implementing interactions among various objects, tasks, and modules than on encoding musical semiotics or data. Puckette explicitly articulated this design philosophy in his article, as follows.

> Rather than a programming environment, Max is fundamentally a system for scheduling real-time tasks and managing communication among them... the expressiveness of Max comes from its interconnection and intercommunication facilities, whereas the contents of the boxes themselves are usually hidden from the user.[1]

A distinctive characteristic of the Max paradigm is that content remains meaningless unless it is connected; by default, Max minimizes stylistic bias in how those connections are made. The blank surface of Max's initial workspace exemplifies this.[1] Zicarelli described this as *incompleteness*, a concept he elaborates on in detail in his article.[2] This notion of incompleteness will be further explored in the following section, alongside the concept of abstractions.

3. THE HIERARCHY OF INCOMPLETENESS AND ABSTRACTIONS

The title of Zicarelli's article published in the *Computer Music Journal* is *How I Learned to Love a Program That Does Nothing*. As the title suggests, Zicarelli also expresses the view that Max is a paradigm less centered on content or data. He described Max as a "universal language of meaningless numbers,"[3] emphasizing that the arbitrary relationships between elements in Max are more important than the elements themselves.[2]

Zicarelli further expands this idea into a broader scope, which he called it as the *hiearchy of incompleteness*.

> One way to think about Max is that it exists as a set of hierarchically related layers of incompleteness. In short, constructing a system is not enough: one must still operate it to achieve results. But this incompleteness extends to lower levels-most importantly, within the software itself. Rather than solve a particular problem in a particular way, Max creates an ecology within which combinations of elements can form a solution. I like to think Max resembles an "ecosystem" more than a language.[2]

Zicarelli's concept of the hierarchy of incompleteness operates on two levels. First, similar to Puckette's view discussed in Section 2, the Max paradigm places stronger emphasis on interconnection and process rather than on data or musical content. As such, Max is inherently 'incomplete' without real-time interactive use or without a user to engage with it. Second, Max is designed to actively incorporate objects and packages developed by external system builders, which constitute a significant portion of the Max ecosystem. In this sense, base Max is incomplete by design.

To support the notion of incompleteness for system builders, it is crucial to enable a constellation of objects to coexist without overriding the entire Max ecosystem.[2] Zicarelli introduced several design decisions in Max to achieve this. [1] The first is the use of dynamic binding, "in which the base class handles only the most basic tasks of messaging, construction, and destruction. Almost everything else is handled with C functions that are associated with symbols at runtime when classes of Max objects are initialized."[2] Dynamic binding makes it possible ''to extend the environment itself without breaking anything," and is used extensively throughout Max.[2] The second is Max's stack-based execution model for nonsynchronous message passing. In a stream of events, "each subsequent step is triggered by a function call inside the previous step, meaning that an execution stack grows with each successive step."[2] Max sets up objects to communicate directly, allowing a function call in an output object to immediately trigger the corresponding function in an input object with minimal overhead.[2]

These technical implementations have enabled people from diverse backgrounds to contribute their own versions of Max objects and packages through a process of computational abstraction. Computational abstraction exhibits two contrasting characteristics. Internally, it is self-contained, driven by the formal semiotics of logic and mathematics.[4] Externally, however, computation also abstracts surrounding cultural and social dynamics while simultaneously "partaking" in them.[4] As a result, computationally abstracted entities acquire material agency—understood as "the social associations of networked practices, which aggregate and shape both the technological and the cultural."[4]

In the case of Max, musical intent and ethos from diverse communities become abstracted into software materiality—through patches, packages, C/C++ code, and more. These objects encode the musical character envisioned by their creators. However, because Max allows such user-generated objects to be easily interconnected and exist without hierarchy, it opens up new musical possibilities for users. Through this process, Max becomes situated within a broader network of relations and community.

The Max paradigm was shaped by several key design goals: a focus on real-time musicking, an emphasis on process over data or semiotics, non-hierarchical relationships among functional elements, and Zicarelli's notion of in-

[1] Zicarelli did mention that additional mechanisms are not revealed in the paper.

completeness. These principles were concretely realized through specific technical implementations, as detailed in the writings of Puckette and Zicarelli. Interestingly, these characteristics have positioned Max within a broader network of social and communal relations, making it a compelling case study within media studies. As a software application with an artistic purpose, Max's unique materiality has directly fostered an interactive relationship with its user community and the surrounding social infrastructure. The following section introduces the theoretical context relevant to the Max paradigm, focusing on how its materiality, affordances, and networked relations intersect.

4. NETWORKED LABYRINTH, RHIZOME, AND MEDIA MATERIALITY

Puckette's vision of Max, as described in his article in the *Computer Music Journal*, closely resembles the characteristics of a musical instrument. Beyond the explicit metaphor of the piano, the strong emphasis on real-time operation and its triggering/scheduling mechanisms reinforce this analogy. As such, examining Max from a critical organological perspective offers valuable insights for its theoretical inquiry.

Thor Magnusson offers a relevant theoretical framework for understanding the organology of digital musical instruments, which he describes as a "heterarchical method of analysing, archiving, and representing instruments."[5] This approach is heavily inspired by Umberto Eco's semiotic research on the structures of knowledge organization, particularly the distinction between the dictionary and the encyclopedia. In his book *From the Tree to the Labyrinth: Historical Studies on the Sign and Interpretation*,[6] Eco traces the evolution of conceptual and semiotic structures from a hierarchical tree model, exemplified by the dictionary, to a labyrinthine model, represented by the encyclopedia.

The tree structure seeks to organize concepts hierarchically based on differences. The genus of these hierarchical categories is defined by differences that serve as "necessary and sufficient conditions to distinguish one being from another and to make the definiens or definer coextensive with the definiendum or definee."[6] However, this closed system of taxonomy revealed its limitations over time, gradually giving way to a more open-ended model: the labyrinth. Eco describes the labyrinth as "a maze of ambiguous routes, of deceptive appearances of things and signs, of winding and complicated nodes and spirals—and we will see eventually, apropos of the rhizomic nature of an encyclopedia."[6] He identifies three types of labyrinths: the unicursal labyrinth, which has a single path; the Irrweg labyrinth, which contains multiple paths including false ones; and the network labyrinth, in which every point can potentially connect to every other.[6]

As Eco also acknowledges, the network labyrinth resembles the *rhizome* as conceptualized by Deleuze and Guattari.[6, 7] Deleuze and Guattari summarize their notion of the rhizome as follows in their book *A Thousand Plateaus*.

> unlike trees or their roots, the rhizome connects any point to any other point, and its traits are not necessarily linked to traits of the same nature; it brings into play very different regimes of signs, and even nonsign states ... It is composed not of units but of dimensions, or rather directions in motion. It has neither beginning nor end, but always a middle (milieu) from which it grows and which it overspills ... The rhizome is an antigenealogy. It is a short-term memory, or antimemory. The rhizome operates by variation, expansion, conquest, capture, offshoots ... In contrast to centered (even polycentric) systems with hierarchical modes of communication and preestablished paths, the rhizome is an acentered, nonhierarchical, nonsignifying system without a General and without an organizing memory or central automaton, defined solely by a circulation of states.[7]

Drawing on Deleuze and Guattari's metaphysics of the rhizome and Eco's semiotic concept of the network labyrinth, Magnusson proposes a new organological framework—what he calls a "rhizomatic organological framework of instrument analysis."[5] This approach emphasizes how a constellation of elements is queried, connected, and interacts, rather than offering a hierarchical classification based on inherent features.[5]

Even at first glance, Max's user interface resembles a network labyrinth, with objects interconnected through patch cords. This visual metaphor is reinforced by the internal design characteristics of Max, which align with the labyrinthine and rhizomatic models in several ways. Max objects are designed to be hierarchy-free, existing on a single flat layer. As discussed in Section 2, Max does not utilize scoping or namespaces, meaning that all symbols and their associated bindings operate within a unified symbolic space.[1] Zicarelli explicitly affirms this, stating that "none of [the] hundreds of objects is technically more fundamental than any other."[2] Moreover, Max is implemented to maximize interconnectivity among objects through nonuniform and heterogeneous data handling.[1] These design features constitute the media materiality of Max and contribute to its rhizomatic nature. Perhaps most significantly, Max's defining characteristic is its unique incompleteness, as discussed by Zicarelli—a consequence of its emphasis on process over data. This calls for a deeper theoretical engagement with Deleuze's notion of difference.

In his book *Difference and Repetition*, Deleuze argues that the notion of difference in Western philosophy has traditionally been "conceived of as an empirical relation between two terms which each has a prior identity of its own (x is different from y)."[8] Deleuze inverts this priority: identity still persists, but it is now understood as something produced by a prior relation between differentials—"dx rather than not-x."[8] He reveals the paradox of the unrepeatable nature of the Same and posits repetition as an expression of difference. For Deleuze, "Being is the difference itself," and this difference operates not through negation, but through affirmation.[9] This is because negation presupposes a categorical relation grounded in preexisting identity—precisely what Deleuze critiques. Difference, in his view, is affirmative in its capacity to synthesize and generate. This concept is famously illustrated through his analogy of lightning.

> However, instead of something distinguished

> from something else, imagine something which distinguishes itself - and yet that from which it distinguishes itself does not distinguish itself from it. Lightning, for example, distinguishes itself from the black sky but must also trail it behind, as though it were distinguishing itself from that which does not distinguish itself from it. It is as if the ground rose to the surface, without ceasing to be ground. There is cruelty, even monstrosity, on both sides of this struggle against an elusive adversary, in which the distinguished opposes something which cannot distinguish itself from it but continues to espouse that which divorces it. ... difference is made, or makes itself, as in the expression 'make the difference'.[9]

This analogy encapsulates several key concepts of Deleuzian notion of difference; the difference prior to identity, its generative and synthetic nature, and difference being the fractured origin of Being.

This theoretical inquiry offers a clearer understanding of Puckette's notion of "process over data"[1] and Zicarelli's concept of *incompleteness*.[2] As noted in Section 2, "the expressiveness of Max comes from its interconnection and intercommunication facilities,"[1] and content or data remains meaningless unless it is connected. In this regard, Max implements a software-based rhizomatic organology by privileging the relationships between differentials over fixed data or objects. In this way, Max operates within a dynamic repetition of incompleteness and difference, and it is intentionally designed to amplify this condition by maximizing interconnectivity and extensibility. In this sense, the Max paradigm can be understood as a *rhizomatic musical instrument that embodies Deleuze's inversion of the traditional relationship between difference and identity—to the extent that such a relation can be realized within a computational environment*.

Zicarelli's notion of incompleteness can be understood also in this context, and as he mentioned, this incompleteness extends to the broader range of community, especially to the system builders. This is also done computationally such as a release of a new package or a different version of Pd. This process is mediated through software, which functions based on its own materiality, and it operates in a different dynamic than the rhizomatic nature inside Max.

5. GENEALOGIES OF ABSTRACTIONS

In Max, the notion of incompleteness extends beyond the software itself through abstractions. Contributions to Max as a system builder can occur either within Max's own environment or through lower-level C/C++ programming. Max offers a lower barrier to entry and enables rapid development, whereas C/C++ requires more advanced engineering skills and additional resources. However, C/C++ development allows for the creation of more expansive and sophisticated software components. As a result, projects that is focused on the system building wing of Max often pursue the C/C++ route—typically in the form of external packages or alternative versions such as Pd. That being said, C/C++ is not a rhizomatic programming environment as Max, but requires greater formality, rigor, and hierarchy of structure. Therefore, it inevitably requires a pre-conception for what to build before writing the code.

As discussed in Section 3, software abstraction in artistic applications contributes not only functionality but also artistic intent and ethos—each carrying its own form of agency. Furthermore, large-scale projects of this kind often establish their own paradigms and, in some cases, even form genealogies. For example, Rebecca Fiebrink's *Wekinator*,[10] a machine-learning-based gesture mapping tool, has influenced various Max projects, such as *MuBu*[11] from IRCAM and *FluCoMa* from the University of Huddersfield. Similarly, the *Bach Project*[2] inherits the Lisp-based computer-aided composition (CAC) paradigm and applies into Max's real-time environment. It comprises four packages: *Bach*,[12] the core package required by the others; *Cage*, which provides high-level CAC modules;[13] *Dada*, designed for concatenative corpus-based CAC;[14] and *ears*, which supports 'streamlined sound-based offline algorithmic practices'[15] using buffers. The cumulative size and complexity of each paradigm—whether machine-learning-based gesture mapping tools like MuBu and FluCoMa, or the Bach package family—are comparable to that of the base Max environment. The Max paradigm allows them to coexist and remain compatible with one another without significant constraints.

These examples demonstrate the proliferation of diverse musical paradigms surrounding Max, each with its own genealogy, forming a complex interrelation with surrounding communities and institutions. Notably, the coexistence of Max's rhizomatic internal architecture and the externally developed musical paradigms—each shaped through its own genealogy of abstractions—is one of the most compelling aspects of the Max ecosystem. Crucially, this seemingly contrasting coexistence is actually made possible by the rhizomatic design of Max's core, as implemented through digital software. Because Max enables a constellation of objects to seamlessly coexist and interact, it has provided a technological platform for the integration of musical ideas at various scales—from individual creators to large institutions. This dichotomy is somewhat inevitable due to the semiotic difference of lower-level and higher-level, in this case the C/C++ programming environment and Max patch, and its varying complexities.

This implies certain limitations and challenges inherent in the Max paradigm. The formality, rigor, and higher entry barrier of lower-level programming inevitably pose the risk of reifying specific paradigms. This risk can be further amplified by institutional involvement. Such constraints are somewhat inevitable due to the material characteristics of software programming—and are not unique to Max—as software can be extremely fragile, as Born points out.[16] Implementing a musical paradigm and its intended functionalities often requires specialized practices and technical rigor. That said, proper documentation and an active user community are crucial in mitigating this inherent fragility of software code.

That being said, both Max and Pd have maintained active online communities that support a wide range of users. While a comprehensive overview of all community activities related to Max and Pd is beyond the scope of this paper, the Max online forum serves as a notable example. One

[2] https://www.bachproject.net/

particularly active feature is the forum's use of the 'copy to clipboard' functionality, which allows users to easily share patches in response to specific issues or questions. This feature enables the forum to function as a community knowledge base, where various technical and musical approaches in Max are abstracted into patches and circulated within the broader user network. A representative example is a 2008 forum discussion on smoothing, [3] in which several Max users shared different strategies for smoothing real-time input data. These contributions were eventually consolidated into a single patch by a user, with each approach represented as a separate abstraction.

This type of collaborative exchange is facilitated by Max's unique architecture, which allows for high compatibility among objects, as discussed in earlier sections. Such compatibility makes it relatively easy to integrate shared modules into one's own patch with minimal friction—which is often not as straightforward in more standard software engineering languages and environments. In this way, the media materiality of Max significantly shapes the modes of interaction within its online community.

The case of Max illustrates that the influence of media materiality extends beyond technical functionality, shaping the broader cultural and artistic landscape. In other words, Max's materiality is not merely the substrate on which musical ideas run; it is a formative agent that conditions what kinds of musical worlds can be imagined, circulated, and sustained. Max's ethos toward rhizomatic organology makes the coexistence of heterogeneous paradigms—gesture mapping, corpus-based composition, CAC traditions—both inevitable and productive, turning Max into a living archive of contemporary musical thought. At the same time, these musical paradigms form its own genealogies, connected within the complex network of institutions, history, resources, and people. This network imposes its own formality, which are computationally abstracted through a rigid programming environment, manifesting its musical ideas into a material agency.

6. CONCLUSION

In his interview with Darwin Grosse, Puckette stated that his initial motivation for creating Max was to build a "real-time scheduler that can be used in a musical situation [in a computer]." [17] While the significance of Max today extends far beyond that of a computational scheduler, its foundational technological details remain deeply relevant. To understand the significance of a software platform—and media technology more broadly—it is essential to examine its media materiality, analyze its affordances, and consider how it interacts with its users. In this regard, Max has been a particularly unique case. More importantly, it continues to evolve in new directions, warranting further study of its broader implications within the music and digital art communities.

7. REFERENCES

[1] M. Puckette, "Max at Seventeen," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002.

[2] D. Zicarelli, "How I Learned to Love a Program That Does Nothing," *Computer Music Journal*, vol. 26, no. 4, pp. 44–51, 2002.

[3] ——, "Communicating with Meaningless Numbers," *Computer Music Journal*, vol. 15, no. 4, pp. 74–77, 1991.

[4] M. B. Fazi and M. Fuller, "Computational Aesthetics," in *A Companion to Digital Art*, C. Paul, Ed. Wiley-Blackwell, 2016, pp. 281–296.

[5] T. Magnusson, "Musical Organics: A Heterarchical Approach to Digital Organology," *Journal of New Music Research*, vol. 46, no. 3, pp. 286–303, 2017.

[6] U. Eco, *From the Tree to the Labyrinth : Historical Studies on the Sign and Interpretation*. Harvard University Press, 2014.

[7] G. Deleuze and F. Guattari, *A Thousand Plateus: Capitalism and Schizophrenia*. University of Minnesota Press, 1987.

[8] D. Smith, J. Protevi, and D. Voss, "Gilles Deleuze," in *The Stanford Encyclopedia of Philosophy*, Summer 2023 ed., E. N. Zalta and U. Nodelman, Eds. Metaphysics Research Lab, Stanford University, 2023.

[9] G. Deleuze, *Difference and Repetition*. Columbia University Press, 1994.

[10] R. Fiebrink and P. R. Cook, "The Wekinator: a system for real-time, interactive machine learning in music," in *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*.

[11] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, R. Borghesi *et al.*, "MuBu and friends–assembling tools for content based real-time interactive audio processing in Max/MSP," in *ICMC*, 2009.

[12] A. Agostini and D. Ghisi, "A max library for musical notation and computer-aided composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, 2015.

[13] A. Agostini, É. Daubresse, and D. Ghisi, "Cage: a high-level library for real-time computer-aided composition," in *Proceedings of the International Computer Music Conference (ICMC)*, 2014.

[14] D. Ghisi and C. Agon, "Real-Time Corpus-Based Concatenative Synthesis for Symbolic Notation," in *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR)*, 2016, pp. 7–13.

[15] D. Ghisi, "A Library for Offline Algorithmic Audio Manipulation in Max," *Computer Music Journal*, pp. 1–48, 2025.

[16] G. Born, "Computer software as a medium: Textuality, orality and sociality in an artificial intelligence research culture," *Rethinking visual anthropology*, pp. 139–69, 1997.

[3] https://cycling74.com/forums/smoothing-out-numbers

[17] D. Growse and M. Puckette. Podcast 090: Miller Puckette. Youtube. [Online]. Available: https://youtu.be/cqQ8hY9QHkM