Mantautas Krukauskas

Similarities of a Musical Text and Computer Program Source Code: New Music Metrics Possibilities

In the current reality of interdisciplinary development in the field of sciences, humanities and the arts, a new possibilities and approaches arise to discover new ideas and research methodologies.

It's always useful to compare experience and perspectives of different fields, which connect to the same basic principles of creativity. At the very beginning I would like to draw some parallels between the process of music creation and the process of computer software development.

On the very abstract level, we might divide the process of music creation to three general phases:

- Idea of composer is set up;
- Idea is "encoded" into musical text;
- Performer interprets musical text.

Idea of a composer is more or less formal. It might be described as a creative inspiration or expressed as a draft of concept. Finally, it is "encoded", set out with the help of an accustomed notation system. This part of the process includes an important amount of creativity itself, however, the final result – a musical score – is a formal and particular expression of initial idea. It becomes a code for a performer to interpret it and make the idea of composer live. Musical text acts as a medium in this sense.

In the process of software development (creation of a computer program), we can also abstract three similar phases:

- Definition of idea of computer program – algorithm;
- Idea is encoded into source code;
- Source code is being interpreted in a computer with operating system.

Algorithm is an abstract description of a task (or set of tasks) of a computer program. It can be easily compared with the initial idea of composer. In general, algorithm is a program itself, however expressed on a very high level of abstraction. The programming process implements the algorithm by realising it with the help of particular computer programming language. When programming is completed, source code needs to be interpreted in an operating system (i. e., Windows, Linux, MacOS etc.). Program becomes functioning, when it's being interpreted in a particular computer with a particular operating system.

When computer program is being created, a problem is analysed and defined in order to create an algorithm. The initial definition of a musical composition also has quite similar element. In example, composer usually starts from selecting instruments he will use in his work. Instruments can be defined by composer himself or projected in the requirements of a commission for a musical piece. In other instances, inspiration of other art form (Liszt's *Preludes*), events observed (Messiaen's *Catalogue d'Oiseaux*) etc. might serve as a problem or general idea of composition. Script or libretto for a stage work would also serve as a good example. Composer might also set up other predefined elements to have a framework for his creation.

Algorithm of a musical composition might be described as a general concept of the piece, predefining structure, form and/or many other possible aspects. In some cases it might be even drafted as a scheme.

Implementation of algorithm in both composing and computer program creation basically resolves abstract concepts into particular language. Creativity is very important for this process, especially in composing. Programming also involves creative elements, however it is more neutral and impersonal, as it basically determines smooth functionality of a computer program. In music composing the process of a formalisation of an idea involves dense usage of creative approaches, such as particular composing techniques, hardly cognisable intellective manifestations of creative mind etc. It is where music is actually born. However, in both cases the result is a text, where are all thoughts and ideas (from abstract to very specific levels) are laid out.

Process: Software	**Process: Music**
A problem to be solved, analysis and specifications	Definition of composition (i.e., requirements of a commission)
Algorithm	Pre-definition of structure, form, other elements
Implementation (programming)	Implementation (writing musical text)
Parsing/interpreting	Performing
Testing	Rehearsing with both composer and performer(s)
… etc.	… etc.

It is interesting to note, that in computer program creation different persons or groups of persons usually perform all the different parts of the process, by using a "split and rule" policy. In musical composition usually single person is responsible for at least first three steps. However, we might have examples, where one person generates the concept of a computer program and implements it. We also might actually find many cases in the sphere of creative industries, where different persons are responsible for definition, structuring and implementation of musical composition.

The next steps of the process described are connected with decoding of a text (parsing/ interpreting or performing), herewith also with issues of cognition, perception etc. We could also find more interesting parallels there. In example, software functionality testing could be compared to rehearsals of a piece with composer and performer(s) together, when afterwards piece is modified according to the results of the rehearsal. However, I will not focus on these developments, as it is not the main focus of this research currently. The key elements of a text (both source code and musical text) that are important for us in this case are:

- It's a static form of a dynamic idea;
- It's accessible and convenient for analysis;
- It is notated according to the particular, known rules.

To continue this comparison, it is worth noting, that during past 50 years the development of technologies and computers had a strong impact on a process of composition itself. The first steps, role of algorithm and structure definition sometimes are made with the computer assistance. One of the first computer-aided compositions based on this principle was *Illiac Suite for String Quartet* by Lejaren Hiller composer already in 1957. This work used serialism and counterpoint rule sets, which were programmed with *Illiac*, University of Illinois supercomputer. Machine-generated material was performed by string quartet afterwards. It was the origin of a so-called algorithmic composition type, which is still developing now. The main techniques of algorithmic composition include:

- Generation of musical material based on various mathematical algorithms;
- Modification of material;
- Selection of material according to the set of rules.

We can easily notice, that such principles are also common for usual process of creation of a musical composition.

It is further necessary to briefly discuss the issue of the level of abstraction, which is very important both for musical composition and for computer programming.

Composing	**Programming**	**Level of abstraction**	**Composing / programming skills**
Structure, form, principle	Algorithm	High	Low
Operating with structural elements	Object-programming	Medium (High)	Medium
Writing of musical text	Lower-level programming	Medium (Low)	High

From the table provided we could clearly see the parallels of abstraction in composing and programming. Structure, form, principle and algorithm are representations of high level of abstraction. However, particular composing or programming skills needed to define them is low. It means, that basically even a person with minor musical knowledge could devise an abstract form of a composition. Operating with particular structural elements in music (it can be also compared to object programming) requires better knowledge and skills. Writing musical text or source code is less abstract and high-level composing or programming skills are needed.

This short comparison of musical text and source code enables us to be certain, that both have conceptual similarities. Therefore musical text can also be interpreted and analysed as a set of logical functions. The same musical text can be expressed with functions of different levels of abstraction, like structure and form (high abstraction), common elements of a musical text – notes, signs, symbols etc. (medium abstraction), performance data (low abstraction) and so on. One of the musical file formats, which are able to express different levels of abstraction by nesting musical concepts in a logical hierarchy, is MusicXML. This format is under development from year 2000 and is often used as a medium-format between popular musical composition and score-writing environments.

Maurice H. Halstead designed one of the main techniques of analysis of source code, based on logical-semantic analysis and developed it into Software Science. On the basis of similarities of musical text and source code, some of these techniques can be adopted to analyse musical texts.

Source code is analysed by grouping it into functions (operators) and variables (operands). 4 basic metrics are concluded:

- **η**1 – number of distinct operators;
- **η**2 – number of distinct operands;
- N1 – total number of operators used;
- N2 – total number of operands used.

After counting these attributes, more complex measures derive, some of which might be also used in analysis of a musical text. In example:

Program (musical text) vocabulary η:
$\eta = \eta 1 + \eta 2$;
Program <u>vocabulary</u> = <u>distinct operators</u> + <u>distinct operands</u>;

Program (musical text) <u>length</u> N:
$N = N1 + N2$;
Program <u>length</u> = <u>total operators</u> + <u>total operands</u>;

Program (musical text) volume V:
$V = N \log_2 \eta$;
Program <u>volume</u> = <u>length</u> x $\log_2$ <u>vocabulary</u>;

Program (musical text) difficulty D:
$D = (\eta 1 / 2) \times (N2 / \eta 2)$;
<u>Difficulty</u> = (<u>distinct operators</u> / 2) x (<u>total operands</u> / <u>distinct operands</u>).

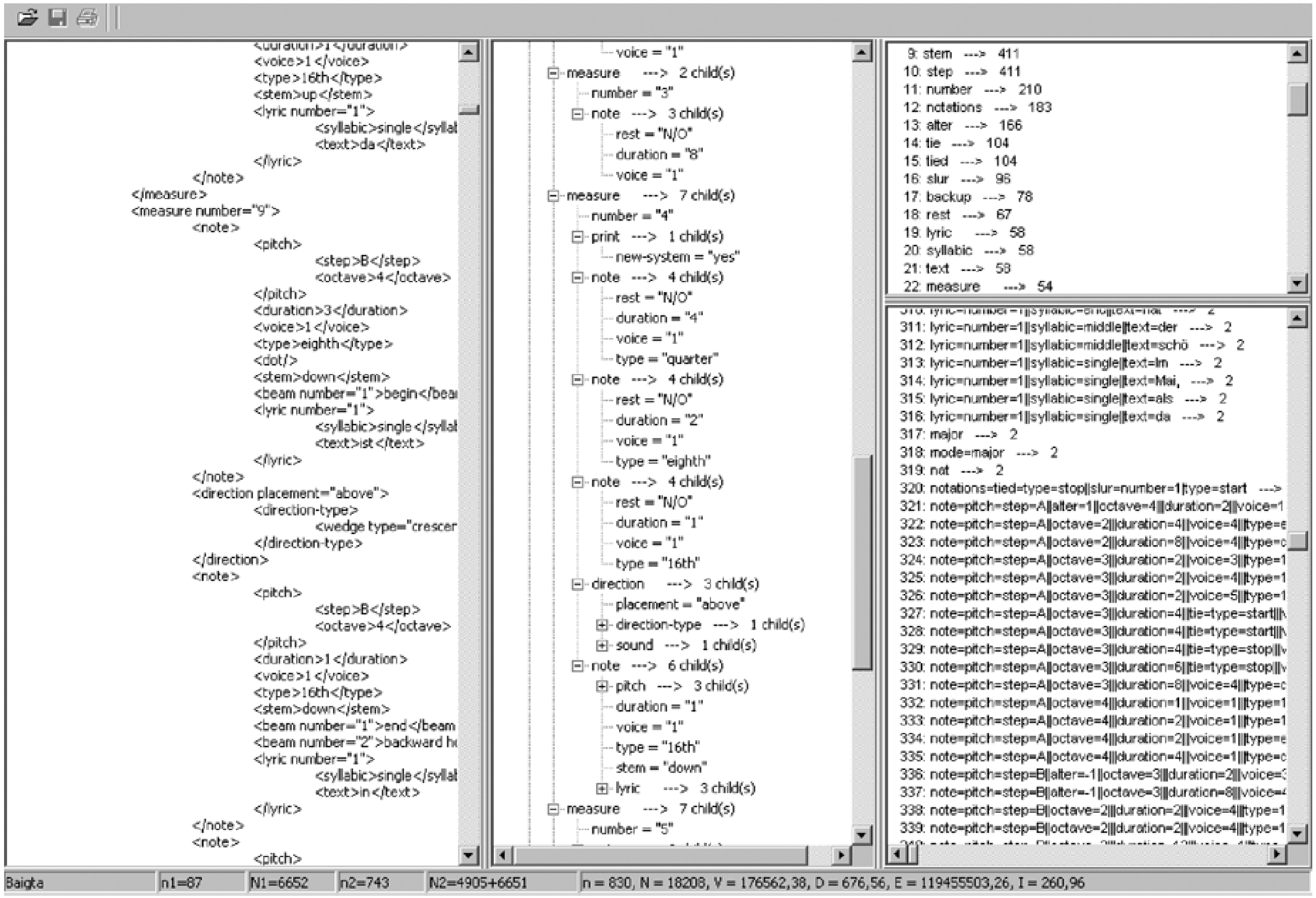
Author of this article conducted a series of experiments, where usage of such metrics was tested with MusicXML format. In Picture 1 screen of a prototype computer program is displayed, where fundamental elements of Halstead metrics are calculated from a musical score in MusicXML format. Already now it is possible to confirm validity of the results, and further development of this research is being done at the moment.

Such analysis might add to the development of impartial metrics for a musical text. To conclude, let's remember famous sentence of Sir William Thomson Kelvin: "I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be."

References

1. Manning, P. Electronic and Computer Music. Oxford, Oxford University Press, 2004.
2. http://www.recordare.com/xml.html.
3. Halstead, M. H. Elements of Software Science. New York, Elsevier, 1977.
4. Thomson Kelvin, William. Popular Lectures and Addresses, Volume 1. London, Macmillan & Co., 1889.

Picture 1. Screen of a prototype computer program



```
                <voice>1</voice>
                <type>16th</type>
                <stem>up</stem>
                <lyric number="1">
                    <syllabic>single</syllab
                    <text>da</text>
                </lyric>
            </note>
        </measure>
        <measure number="9">
            <note>
                <pitch>
                    <step>B</step>
                    <octave>4</octave>
                </pitch>
                <duration>3</duration>
                <voice>1</voice>
                <type>eighth</type>
                <dot/>
                <stem>down</stem>
                <beam number="1">begin</bea
                <lyric number="1">
                    <syllabic>single</syllab
                    <text>ist</text>
                </lyric>
            </note>
            <direction placement="above">
                <direction-type>
                    <wedge type="crescer
                </direction-type>
            </direction>
            <note>
                <pitch>
                    <step>B</step>
                    <octave>4</octave>
                </pitch>
                <duration>1</duration>
                <voice>1</voice>
                <type>16th</type>
                <stem>down</stem>
                <beam number="1">end</beam
                <beam number="2">backward h
                <lyric number="1">
                    <syllabic>single</syllab
                    <text>in</text>
                </lyric>
            </note>
            <note>
                <pitch>
```

```
        voice = "1"
measure    ---> 2 child(s)
    number = "3"
    note  ---> 3 child(s)
        rest = "N/O"
        duration = "8"
        voice = "1"
measure    ---> 7 child(s)
    number = "4"
    print  ---> 1 child(s)
        new-system = "yes"
    note  ---> 4 child(s)
        rest = "N/O"
        duration = "4"
        voice = "1"
        type = "quarter"
    note  ---> 4 child(s)
        rest = "N/O"
        duration = "2"
        voice = "1"
        type = "eighth"
    note  ---> 4 child(s)
        rest = "N/O"
        duration = "1"
        voice = "1"
        type = "16th"
    direction    ---> 3 child(s)
        placement = "above"
        direction-type   ---> 1 child(s)
        sound   ---> 1 child(s)
    note  ---> 6 child(s)
        pitch  ---> 3 child(s)
        duration = "1"
        voice = "1"
        type = "16th"
        stem = "down"
        lyric    ---> 3 child(s)
measure    ---> 7 child(s)
    number = "5"
```

```
9: stem  --->  411
10: step  --->  411
11: number  --->  210
12: notations  --->  183
13: alter  --->  166
14: tie  --->  104
15: tied  --->  104
16: slur  --->  96
17: backup  --->  78
18: rest  --->  67
19: lyric    --->  58
20: syllabic  --->  58
21: text  --->  58
22: measure    --->  54
```

```
311: lyric=number=1||syllabic=middle||text=der  --->  2
312: lyric=number=1||syllabic=middle||text=schö  --->  2
313: lyric=number=1||syllabic=single||text=Im  --->  2
314: lyric=number=1||syllabic=single||text=Mai,  --->  2
315: lyric=number=1||syllabic=single||text=als  --->  2
316: lyric=number=1||syllabic=single||text=da  --->  2
317: major  --->  2
318: mode=major  --->  2
319: nat  --->  2
320: notations=tied=type=stop||slur=number=1||type=start  --->
321: note=pitch=step=A||alter=1||octave=4|||duration=2||voice=1
322: note=pitch=step=A||octave=2|||duration=4||voice=4||type=e
323: note=pitch=step=A||octave=2|||duration=8||voice=4||type=c
324: note=pitch=step=A||octave=3|||duration=2||voice=3||type=1
325: note=pitch=step=A||octave=3|||duration=2||voice=4||type=1
326: note=pitch=step=A||octave=3|||duration=2||voice=5||type=1
327: note=pitch=step=A||octave=3|||duration=4||tie=type=start||\
328: note=pitch=step=A||octave=3|||duration=4||tie=type=start||\
329: note=pitch=step=A||octave=3|||duration=4||tie=type=stop||v
330: note=pitch=step=A||octave=3|||duration=6||tie=type=stop||v
331: note=pitch=step=A||octave=3|||duration=8||voice=4||type=c
332: note=pitch=step=A||octave=4|||duration=1||voice=1||type=1
333: note=pitch=step=A||octave=4|||duration=2||voice=1||type=1
334: note=pitch=step=A||octave=4|||duration=2||voice=1||type=e
335: note=pitch=step=A||octave=4|||duration=4||voice=1||type=c
336: note=pitch=step=B||alter=-1||octave=3|||duration=2||voice=3
337: note=pitch=step=B||alter=-1||octave=3|||duration=8||voice=4
338: note=pitch=step=B||octave=2|||duration=2||voice=4||type=1
339: note=pitch=step=B||octave=2|||duration=2||voice=4||type=1
```

```
Baigta | n1=87 | N1=6652 | n2=743 | N2=4905+6651 | n = 830, N = 18208, V = 176562,38, D = 676,56, E = 119455503,26, I = 260,96
```

Santrauka

Muzikinio teksto ir kompiuterinės programos pradinio kodo panašumai: naujos muzikinės metrikos galimybės

Tiriant kompozitoriaus kūrybą, visada tikslinga remtis autentiška jos išraiška – muzikiniu tekstu. Muzikinis tekstas taip pat yra kūrinio analizės bazinis elementas.

Muzikinio teksto ir kompiuterinės programos pradinio kodo analogija yra akivaizdi. Tam tikru sutartu kodu (natomis, ženklais) kompozitoriaus minčiai yra suteikiamas statiškas pavidalas. Vėliau ši informacija grąžinama į dinaminį, erdvėje bei laike egzistuojantį būvį – perduodama atlikėjui-interpretatoriui. Kompiuterinio kodo programuotojas taip pat pagal nustatytas pasirinktos programavimo kalbos taisykles išreiškia tam tikrą algoritmą, kuris yra interpretuojamas konkrečioje operacinėje sistemoje ir paverčiamas realiai funkcionuojančia programa. Kompozitoriaus mąstymas irgi turi daug analogijų su algoritmizavimo principais.

Programavimas, algoritmo kodavimas tam tikroje programavimo kalboje (kompiuterio programų kodų užrašymas) ir muzikinė kūryba (jos užrašymas tam tikru formaliu raštu) yra panašios prigimties ir jiems galioja tie patys formalizavimo bei interpretavimo principai. Geriausias šio panašumo įrodymas – 2000 m. sukurta MusicXML sistema (www.recordare.com).

Analizuojant muzikinį tekstą, yra siūloma jį traktuoti kaip operatorių ir operandų (funkcijų ir kintamųjų) eilę. Tas pats muzikinis tekstas gali būti išreikštas keliais skirtingais funkcijų ir kintamųjų kompleksais, lygiai taip, kaip tas pats algoritmas gali būti išreikštas skirtingo lygio programavimo kalbomis. Žemesnio lygio programavimo funkcijos apima konkretesnę, su procesoriaus komandomis susijusią sintaksę, vidutinio lygio – jungia elementarias funkcijas į sudėtinius procesus bei tam tikrą formą, o objektinio programavimo kalbos, apimančios ir visas kito lygio kalbų funkcijas, leidžia tiesiogiai operuoti sudėtingomis funkcijomis ir esminėmis programos struktūros sistemomis (objektais).

Pagal XX a. 8-ąjį dešimtmetį amerikiečių fiziko ir informatiko Maurice H. Halsteado išplėtotą metodologiją (Halstead, Maurice H. Elements of software science. ELSEVIER, New York, Oxford, Amsterdam, 1977), tam tikros matematinės formulės leidžia nustatyti programos kodo sudėtingumą, žodyną, informacinį turinį bei kitus rodiklius, kurie gali būti gretinami tiek su kitomis to paties algoritmo išraiškomis skirtingose programavimo kalbose, tiek su kitų programų algoritmais. Šių matų pagrįstumas buvo įrodytas tiek matematiškai, tiek empiriškai.

Šios metodologijos pritaikymas muzikinio teksto analizei suteikia galimybę nustatyti tam tikrus objektyvius rodiklius, kuriant muzikinio teksto kiekybinės metrikos sistemą.