Estimating Interaction Time in Music Notation Editors

Matthias Nowakowski[1] and Aristotelis Hadjakos[1] *

Center for Music and Film Informatics (CeMFI),
University of Music Detmold, Germany
matthias.nowakowski@hfm-detmold.de

Abstract. Modern music notation software is extensive and so can be a comparative analysis. Since they employ a lot of different interactions to write scores, due to the mass of different symbols and their combinations, we developed a method to estimate the time spent by the user in interacting with the software interface in order to perform fundamental operations. For this we applied and extended the Keystroke-Level Model by analyzing interaction percentages in MusicXML files. Our findings contribute to modeling interaction and usability/ user experience research about interaction in music notation editors. These findings can be then transferred to analyze other editors and we expect to use the method in formative analyses to reduce user studies and thus development time in the long run.

Keywords: Human Computer Interaction · Music Notation Editor · Keystroke Level Model

1 Introduction

Score editors allow users to create, edit and play musical scores. They are widely used by composers, musicians, teachers and students for various purposes, such as composing music, arranging songs, transcribing audio, or learning music theory. However, developing score editor interfaces with good usability and user experience is hard. User interface design is a complex task in general, as evidenced by the documents of the extensive ISO standard 9421 [1], which provides various guidelines for interface design. In the context of score editors, it is necessary to identify the needs and context of use for the score editors, to specify the design criteria as well as functional and non-functional requirements, to produce design solutions, create prototypes and test them with users or experts. Implement the design solutions and evaluate the use of the score editors in real or simulated situations.

* We would like to thank Claudia Cecchinato, Árpad Kovács and Juan Sebastian Mora Lopez for creating the KLM encodings and giving valuable input for task description and selection.

Another important challenge is media specificity by which we mean the purpose for which a software is created and by which means the (editable) medium can be accessed. For example, a word processor is created for the purpose of writing and editing text documents. In the case of text editing, consensual interactions and modalities with mouse and keyboard were established which are based on the metaphor of the type writer. These interactions are familiar and so feel intuitive to most users. Transferring the input modality to another medium—namely sheet music, music creation and music editing in general—may cause conceptual dissonance simply because of mismatch of input and output symbols and gestures. This means that users may find more difficult to match interactions that rely on a text-based keyboard to interact with musical symbols. In lack of consensual metaphors, it is symptomatic that widely used notation editors such as Sibelius, MuseScore, Dorico and Finale often employ vastly different interaction paradigms combining mouse and keyboard interaction in idiosyncratic ways. We could already show that standard questionnaires yield mostly low ratings for usability and user experience [2]. Moreover, we found in that study that most users used the score editors mainly for practical purposes and tend not to require features that are specifically supporting the creative process.

By visualizing a distribution profile of task-specific interaction times, we can gain insight into how long different tasks take and which ones are particularly time-consuming. With this approach we analyze six widely used score editors as well of our web-based score editor for Learning Management Systems, which uses the Verovio[1] engraving packet to render musical scores.

2 Related Work

To our knowledge there is no systematic and comparative work which deals with usability and user experience of notation editors. Nevertheless, Human Computer Interaction (HCI) is a prevalent topic in music research especially in the context of New Instruments for Musical Expression (NIME) [4] and Education [5]. Dealing with music production specifically Nash et al. [6][7][8] were interested in the creative involvement of the user with trackers and sequencers based on the concepts of Cognitive Dimensions [9] and Flow [10]. Although score editors are not explicitly analyzed they employ the same feedback loops as trackers, albeit by using different symbols. Based on this we also applied these metrics to analyze characteristics of usage of different score editors and isolate significant items by which they can be best described and most efficiently assessed [2]. Peterson et al. [11] approached the quality of creative outcome in digital media and on paper respectively by measuring interaction times.

A good starting point to research interaction times in general is to refer to the earliest of HCI research relating to text editors which can be adjusted and applied to music notation editors which we could view as text editors with special requirements. First wave HCI methodology in the 1980s was concentrated on operations which could be modeled as simple reactions in order to operate a system, ignoring such factors as emotions or the personality of the user. Card et al. [3] introduced the Keystroke-Level model which

[1] `https://www.verovio.org/index.xhtml`

was intended to model expert user interactions with text editing software on a low level, consisting of operations such as “Keystroke”, “Button Press” and “Pointing”. However, mental operations were also introduced and with higher complexity of software core tasks had to be defined [12] to focus on the most relevant interactions, i.e. tasks which every software with the same purpose should have it implemented. We will discuss how we defined these operators and core tasks for our research in Section 3.

In contrast to this, the GOMS (Goals, Operators, Methods, Selection rules) Model tries to explain a users behavior from a top-down perspective. It takes the actual goal of the action into account and fragments it into the actions that have to be taken to accomplish it [13]. This is useful in analyzing the procedural knowledge users and why they might use a certain interaction path. It is also useful to model new tasks around the given goals, since it is not based on an existing system [14].

Today the Keystroke-Level model remains a viable tool for fundamental research with new input modalities and situations, e.g with touch screens [15][16], in virtual reality [17], device interaction while driving [18] or exploring interactions with non-western writing systems [19]. Of course, the list of operators was adapted, where necessary to accommodate for new input devices and gestures [20].

3 Method

3.1 Program Selection

To decide for which music notation editors to compare we referenced to a previously conducted study in which we analyzed the most used ones [2]. From the 29 mentioned programs six were viable for statistical analysis, being Capella, MuseScore, Dorico, Finale, Sibelius and Lilypond. For the paper at hand we were not able to make a KLM analysis for Lilypond, since it is entirely text based and cannot be adequately compared to graphical user interfaces (GUI) we implicitly had in mind for the study. Since MuseScore had a major update during the preparation of the data, we also decided to integrate the KLM of MuseScore Versions 3 and 4, giving us the opportunity to discuss recent changes in their interaction design.

3.2 Data Collection & Evaluation

First we agreed on a set of unit tasks, meaning any atomic tasks that can be accomplished with a music notation editor. Encoding these tasks then was then performed by three people according to Card et al. [3]. We did not expect the encoders to know every program, but they must have worked with music score editors in the past. We are aware that each encoder might have more experience with a certain program and to ensure an average view on the multiple interaction paths we have taken the following measures:

– Multiple people explored each software for the same unit task. This accounts for different ways to solve a task in the case the software has different paths.
– Encodings were taken for different modalities, i.e. major keyboard and major mouse use respectively. This represents people with different ways of working. Although many people might prefer a mix of both, we have a potential range and a basis to interpolate between those values.

– The resulting times of all encoders were averaged for each software and modality.

The KLM provides encodings and already fixed times for series of actions (operators) as "methods" that are necessary to perform a certain task. The operators can be mental preparation (M), keystroke (K), button click (B), homing (H), pointing (P) and selecting from a pull down menu (pd) in different combinations which are empirically determined and applied in our study. The encoders worked on their own computers. We regard the variability of screen sizes as negligible, since the KLM already provides times which are insensitive to this factor.

We also aim to incorporate actions involving multiple inputs that lead to valid changes in the score, such as composing compound elements like tempi or chords. To achieve this, in the following section we calculated average sequence lengths, which we utilized as factors for encoding interaction methods.

As we cannot anticipate every potential context in which a task might arise, the encoding process may result in tasks being coded with slightly slower execution times than they would exhibit in actual scenarios. For instance, consider the scenario where a specific palette must be accessed before adding an articulation, and typically, the palette remains open when multiple articulations are added in sequence. However, in our encoding approach, the act of opening the palette is always included. Consequently, it's important to acknowledge a margin of error, which could extend up to 20% according to previous research [3].

3.3 Task Selection

KLM describes existing systems on a single level by taking inventory of interaction durations and so making tasks comparable between systems. Since not all tasks are used with equal frequency, we decided to perform four steps that helped us to access interaction times for relevant tasks:

1. Define all unit tasks that are found in at least one music notation editor.
2. Analyze MusicXML data to find frequencies of all elements that result from interactions.
3. Apply the frequencies from step 2 as weights to compute distributions for all unit tasks.
4. Filter unit tasks with the help of step 2 that account for 95% of interactions. Include interactions that are necessary to write a valid music score to get a more manageable number of tasks to discuss. All these tasks we will be denoted as "core tasks".

In total we defined 234 unit tasks first. These are actions that lead to a visual and/ or sound change in the GUI (including score and menus). This effectively filters out all subordinate system interactions which only indirectly contribute in visual outcome. Unit tasks do not have to be solely tasks that change sound events such as notes, chords or articulations. This can be annotations of every kind, as well as lyrics, but also the act of selecting elements, since they add highlighting to the score, and playing the music which adds automatic highlighting to the currently sounding events.

According to Roberts et al. [12] core tasks consist of a cross product of the following operations and objects as seen in Table 1. We had to do some accommodations for musical syntax, since some operations like "transpose" have different meanings in music. We also omitted "swapping", "splitting" and "merging" for which we found no scenarios in the described GUIs. Also the number of objects is much larger than in linguistic text, so that we had to group symbols in a similar hierarchical manner (Table 2).

Table 1. Operations and objects according to [12].

Operations	*Objects*
insert	
delete	character
replace	word
move	line
copy	sentence
transpose (≈ swap)	paragraph
split	section
merge	

Table 2. Adjusted operations and objects for music notation editors.

Operations	*Objects*
add	
delete	primitives (notes, rests, lines, clefs, marks, etc.)
replace	diacritic signs (beams, articulations, ornaments, etc.)
move/displace	compounds (chord, measures, key signatures, tempo, etc.)
rebind	semantic structures (parts, voices, lyrics, annotations, etc.)
copy	
paste	

Some operations are only applicable to some objects such as transposing can only be applied to chords and notes while rebinding (bind an anchor to a new event and so making also a change in the synthesized sound) is mostly associated with elements which modify the sound on larger time scales such as crescendo/ decrescendo, tempi, slurs, dynamics, etc.

To get a more concise view of the frequencies of occurrence of all elements we analyzed freely available MusicXML files by simply counting the elements and mapping them to their corresponding tasks. We also counted sequences of inputs to account for unit tasks that require multiple consecutive inputs, like writing a sequence of notes with the same duration, writing a chord, writing chord symbols or textual tempo instructions. For durations this value lies at 1.4, word length is 5.7 on average. The mean of all found sequences in the analyzed pieces is 3.6 which we will use as a multiplier to compute individual task related times.

As a base for our model we took four pieces from different time periods, with different instrumentation to cover a wide range of quantities of used symbols:

- Johann Sebastian Bach: Orchestral Suite in D Major (BWV 1068)
  20897 elements
- Wolfgang Amadeus Mozart: Clarinet Concerto in A Major (KV 622)
  81844 elements
- Frédéric Chopin: Three Waltzes (Op. 64)
  13209 elements

- Frederik Pfohl: Symphonic Phantasy for great Orchestra *The Sea*, Movement 5 *Frisian Rhapsody* (PWV 24)
  92519 elements
- Gabriel Fauré: Piano Quintet No. 2 (Op. 115)
  75591 elements

Table 3 shows the most used elements in the MusicXML that account for ≈ 95% of interaction according to the weighted means. These percentages are representing the weights which we will apply to the tasks resulting in the distribution in Figure 1.

In the table "type" is referring to the symbolic duration (quarter, 16th, etc.), which by itself accounts for 35.72% of interactions in a notation program, followed by pitch with 29.25%. Slurs and articulations are child elements of "notations" element and can include further symbols that modify the note such as ornaments, arepeggios etc. "Dot" represents the prolongation of a note.

We decided to base our evaluation on the weighted mean, since we have wide differences in element numbers per piece. By this we assume that the selected pieces are somewhat representative for scores produced for music of this period.

The ranks of the elements for each piece follow the ranks of th arithmetic and the weighted mean in general. The arthmetic mean has some shifted numbers, only the ranks for "slur" and "accidentals" are swapped. Higher shares of accidentals are found for Chopin, Fauré and Pfohl, whose pieces may include extended harmonic development which can likely occur in 19th century pieces. Rests have higher percentages in the large orchestra pieces (Mozart and Pfohl), where entire instruments could stop playing for long times which results in a relatively high standard deviation of 4.8%.

Table 3. Percentages of MusicXML elements that are found in all of the analyzed pieces and account for ≈ 95 % of the interacton with the score. sd = standard deviation, mad = mean absolute deviation, wt ... = weighted ...

element name	*BWV 1068*	*Chopin Op.64*	*Fauré Op.115*	*PWV 24 (Mvt 5)*	*KV 622*	*mean*	*sd*	*mad*	***wt mean***	*wt sd*	*wt mad*
type (= symbolic duration)	38.98	41.24	34.01	34.54	36.92	37.14	3.03	3.52	**35.72**	1.96	1.77
pitch	33.88	39.44	29.76	27.49	27.94	31.70	5.01	3.37	**29.25**	2.83	1.20
rest	5.10	2.03	5.68	13.20	11.98	7.60	4.78	5.42	**9.73**	3.75	3.74
beam	14.16	2.99	7.06	1.63	7.32	6.63	4.89	6.03	**5.70**	3.50	3.41
slur	0.58	2.94	5.16	4.72	5.48	3.78	2.04	1.12	**4.67**	1.27	0.51
accidental	1.38	5.65	7.93	2.82	2.39	4.04	2.69	2.13	**4.08**	2.44	0.83
articulations	0.22	0.11	0.99	8.06	3.73	2.62	3.37	1.30	**3.98**	3.09	4.15
dot (= prolongation)	2.94	0.91	3.22	2.86	1.92	2.37	0.95	0.52	**2.60**	0.63	0.44

However, these elements do not encompass the entirety of essential functionalities found in music notation editors. The editor must include specific features for initializing and managing information necessary for reading and playing from a score, as these aspects are imperative for its validity. We have meticulously selected these features and refer to them as "essential tasks". Table 4 summarizes the number of relevant tasks over the 12 most central unit task areas according to the combination scheme of operations and objects mentioned in Table 2 and Table 3.

Table 4. Number of core tasks accounting for most relevant interactions in music notation editors.

unit task area	*correspondences in XML elements*	*number of core tasks*
duration	type, dot, rest, chord	16
pitch	pitch	6
accidental	accidental	5
beam	beam	3
notations	slur	5
	articulations	6
initial score configuration	essential tasks comprising various compounds of XML elements	2
time signatures		5
key signatures		6
tempo		7
clefs		6
playback		1
staff/ measure		11
	tasks total	101

4 Results

4.1 General

In Figure 1 we show all accumulated KLM values that we encoded for all accessible functionalities, separated by using (if possible) only keyboard or only mouse. It is not surprising that mouse interaction is much slower than pure keyboard interaction in general. The distributions are already weighted according to Table 4. Mouse modality has less outliers in general which points to more equally distributed data. The violin plots now mostly remind hi-hats, meaning that interaction times of core tasks cluster around different regions with few tasks in between. Despite some variations between the graphs one can clearly identify peaks in the lower portions which mostly represent core tasks. Tasks that could be subsumed under "notations" as well as pitch and duration related tasks have usually similar speeds within the software and modality and so forming distinguishable peaks.

The descriptive statistics in Table 5 show that most of the tasks are performed in very similar speeds. Overall Sibelius is slowest for mouse interaction with 6.89 seconds. MuseScore4 is the fastest in key interaction with 3.22 seconds. Dorico, Finale and MuseScore 3 are significantly faster in mouse interaction than the rest. In general most tasks are performed in between 3 to 9 seconds.

Comparing MuseScore 3 and 4 we can see, that the later Version tends to make some interactions slower especially with mouse interaction. Pitch and and duration related tasks are clearly visible in the peaks. For mouse interaction the times in both editors are similar, but MuseScore4 having a higher median despite having a similar interquarile range. This indicates that non-essential tasks have become faster, which are not heavily weighted. Comparing the peaks around 7 seconds with Sibelius we can find mostly tasks for "notations" like in MuseScore 3 and 4 but Sibelius also includes many tasks about various changes about staves and element displacement which is usually faster in other editors. Many similar peaks over a wide range resulting in a mostly symmetric

Table 5. Descriptive statistics for Figure 1. q1 = first quartile, q3 = third quartile, iqr = interquartile range, mad = median absolute deviation, sd = standard deviation, se = standard error, ci = 95% confidence interval.

software	*modality*	*min*	*max*	*median*	*q1*	*q3*	*iqr*	*mad*	*mean*	*sd*	*se*	*ci*
Capella	key	2.08	13.58	3.56	2.96	4.46	1.51	1.3	4.14	1.66	0.01	0.01
	mouse	2.55	14.89	6.86	5.34	8.1	2.75	1.84	6.6	2.17	0.01	0.02
Dorico	key	1.95	15.17	3.58	2.95	4.01	1.06	0.74	3.81	1.33	0	0.01
	mouse	1.95	15.17	4.46	3.75	8.01	4.26	1.48	5.88	2.56	0.01	0.02
Finale	key	1.75	16.26	3.91	3.7	4.66	0.96	0.78	4.24	1.08	0	0.01
	mouse	2.52	19.18	4.94	3.76	8.32	4.56	1.76	5.96	2.85	0.01	0.02
MuseScore 3	key	1.68	15.62	3.47	2.86	4.04	1.19	0.85	3.64	1.15	0	0.01
	mouse	2.42	16.36	4.93	3.75	7.86	4.11	2.79	5.96	2.66	0.01	0.02
MuseScore 4	key	1.75	15.15	3.22	2.72	4.15	1.43	0.94	3.54	1.33	0	0.01
	mouse	2.15	14.25	6.58	3.75	7.79	4.04	3.8	5.99	2.51	0.01	0.02
Sibelius	key	1.68	13.67	3.7	3.2	4.2	1	0.75	4.05	1.42	0	0.01
	mouse	2.35	15.92	6.89	4.56	8.67	4.12	3.46	7.17	3.07	0.01	0.02

shape can be an indicator that some core tasks may be inconsistently modeled, also more coherent plots over a wide range can show special treatment of some methods that are less consistent with similar tasks and should be examined in more detail.

4.2 Outliers

As a rule it is not problematic having many outliers in the set of interactions. Since most of the editors have low third quartile boundaries in the key modality, interactions with lengths of 5 to 7 seconds can already count as outliers in these cases.

From the perspective of the software designer this might point to concentration on faster speeds in interaction design and addressing specific problems that have to be solved in order to appeal to a certain user group. Also, this does not mean, that one program is more preferable over the other due to interaction speed differences. As mentioned our previous study [2] Capella has the best usability and user experience ratings in our experiments, while from a KLM perspective there are more peaks in higher regions in both modalities. With this method it is more important to analyze different peculiarities, like for example having mostly core tasks hidden behind slow or dissimilar interactions.

In Dorico we can see, that the data is much wider distributed using a mouse than using solely keys. Only changing the instrument for a specific staff, transfer notes between voices and creating multiple bars at the end were considered to be very slow. In contrast we have 38 outliers for key interaction, most of them including tasks that immediately result in a different layout, especially adding and deleting measures. But we can also find frequently used objects as described in Table 3, like beams and staves, while 101 Tasks can be completed between 3 to 4 seconds.

Finale has a very characteristic peak at 12.5 seconds for the mouse modality, which consist of some layout and MIDI operations. Also there seems to be no simple way to

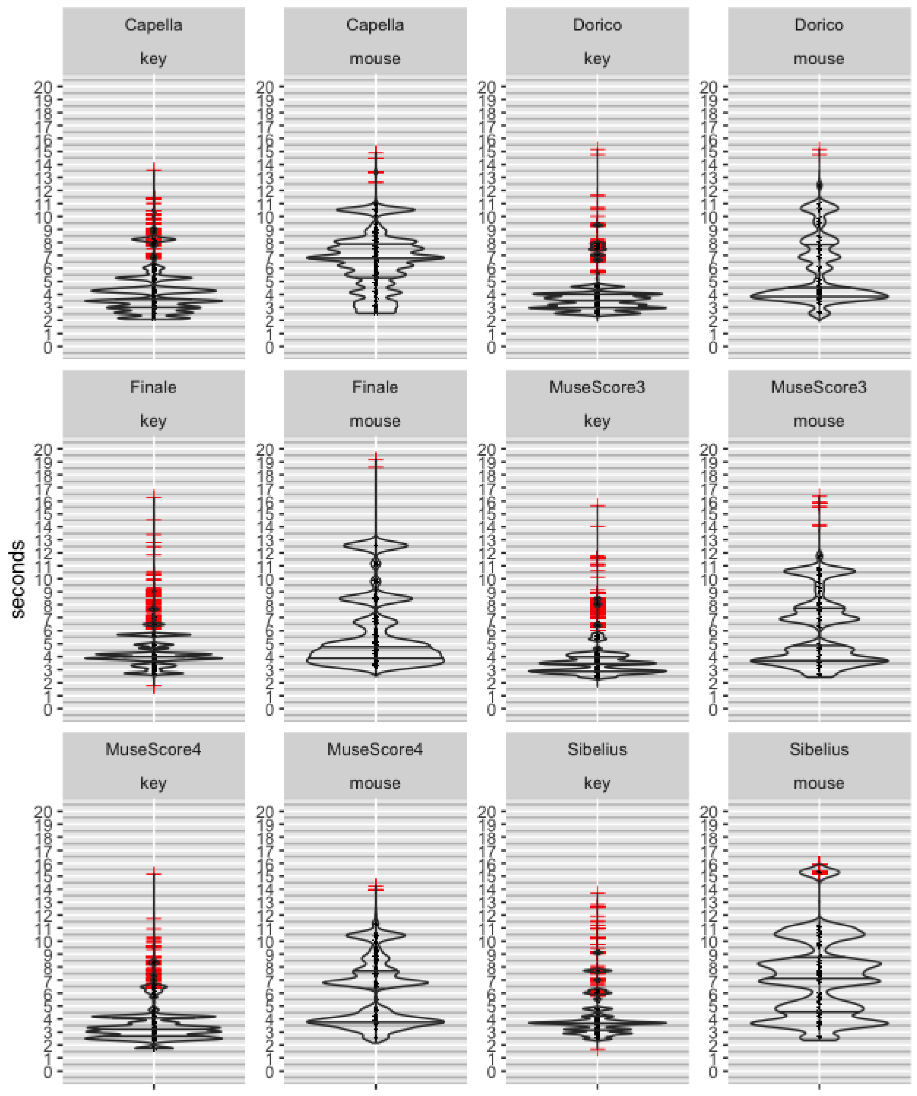

Fig. 1. Violin plots with quartiles of the weighted Keystroke-Level Model of the six music notation score editors. The red '+'-Symbols mark outliers. The width on the x-axis is not

paste notes, chords or rests—objects which are highly weighted—by mouse which in turn is better handled by keyboard interaction.

In general keyboard interactions across all editors are especially slow making major changes to the layouts or creating scores. Most outliers consist of these since there are seldom adequate methods so that here are no options despite using the mouse. These are actions that one might access rarely are not among outliers in any mouse interaction.

This also applies for more fine grained interactions considering changes around staffs and notes, like beams, meter changes or barline related tasks like creating repetitions. It is debatable, if fast keyboard access is necessary if such actions account for less than 2% of the total. Outliers in mouse interactions are mostly idiosyncratic and revolve around elements outside the staff like tempi and charts. Here especially Sibelius, MuseScore3 and Finale seem to have deficits.

4.3 Application

The results shown above provide guidelines for monitoring the ongoing development of our music notation interface called **VIBE** (Verovio Interface for Browser-based Editing) [2], as well as for assessing its performance in comparison to other solutions. While the method presented above entails a summative analysis, we are confident that it can also be adapted to formative scenarios. These scenarios can then be employed at different stages throughout the development process.

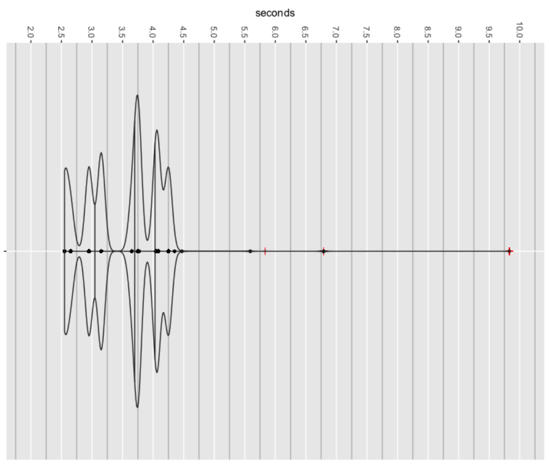

Fig. 2. Violin plot of VIBE. The red '+'-Symbols mark outliers.

VIBE currently implements 41 of the 101 listed core tasks (see Table 4), most of them use mouse interactions which are better explorable visually when using the program for the first time. Mostly "notations" and "dynamics" have to be implemented yet, as well as several actions of copying, pasting and rebinding. Main development dealt with actions around adjusting durations, interacting with the identity of a note directly and creating a valid score. Additionally we were also interested in handling annotations and chord symbols since these are important features for analyzing a score and make information accessible for other persons in a teaching environment, as required by the

[2] Source Code: `https://github.com/mnowakow/VerovioScoreEditor`
Demo: `https://mnowakow.github.io/`

underlying project for which it is developed. This added possible 14 unit tasks of which 10 are implemented, combining to 51 implemented unit tasks in total.

In Figure 2 we can see, that most of the interactions are around 3.75 seconds (with the median at this point and a very narrow third quartile) which belong to the relevant core tasks handling durations and pitch. Although we concentrated on mouse interaction first, the results can keep up with other editors sometimes even with fast keyboard inputs. In our case especially creating time signatures is slow, since it is currently required to choose always from two drop down menus to create a combination of count and unit which then has to be dragged to the intended position. Generally actions that include dragging and dropping items (clef, key, time) are found among the outliers, as well as actions which have to be performed multiple times to accomplish the intended result, like deleting or adding multiple measures at the end of the score.

5 Discussion

In this paper we presented an approach to evaluate music notation editors objectively by simulating and comparing their interaction times. We oriented our research on the original publications about the KLM and applied them to editors by several annotators. By defining unit tasks and model their weights after element occurrence in MusicXML we can find slow and fast interactions and especially locate relevant ones in the resulting distributions. Same speeds usually point to similar sequences of operators. Horizontally symmetric plots over a wide range might point to inconsistencies in the interaction modeling. This helps us to evaluate different music notation editors and model interactions in new interfaces according to access times. These times do not represent rigid metrics but a way do identify potential shortcomings fast.

By listing and ranking core tasks, this paper also contributes to monitor the process of development and functional completeness of notation editors as shown in section 4.3. New editors will at least have to implement the core tasks presented here, but might have different requirements for working more creatively or making elaborate editions. In these cases the list of unit tasks can be extended as presented in section 3.3. KLM is flawed when it comes to evaluating user responses. In our case we approached the topic by modeling methods which are not informed by the user manual, but by exploration and restriction to input modalities which might represent a user with average skills. We did not expect a perfect and efficient user, but did assume a perfectly set score. So still questions remain about the system and user behavior in case of errors: How fast can users correct their errors? What methods does the system provide to make corrections? That is why most research using KLM is concerned with user tests to verify interaction times with new interaction modalities such as touch, pen or VR. In our case, we adopted an established model, as we focused on mouse and keyboard interactions. We then extended its application to a domain within HCI that has received limited scientific attention until now. However, since we base our method on informed assumptions, we still would like to verify these results with actual user tests. This will also help us to bring the results from this paper closer to the field of user experience. When conducting user tests it will be also fruitful to combine it with discussions and questionnaires to evaluate specific usability issues, which could not be represented by the KLM directly.

References

1. Bevana, N., Kirakowski, J., Maissela, J.: What is Usability? Proceedings of the 4th International Conference on HCI, pp. 1–6 (1991)
2. Nowakowski, M., Hadjakos, A.: Online Survey on Usability and User Experience of Music Notation Editors. Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR'23) (2023), in press
3. Card, S. K., Moran, T. P., Newell, A.: The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of the ACM, 23(7), 396–410 (1980)
4. Fasciani, S., Goode, J.: 20 NIMEs: Twenty Years of New Interfaces for Musical Expression. Proceedings of the International Conference on New Interfaces for Musical Expression, (2021), http://doi.org/10.21428/92fbeb44.b368bcd5
5. Repenning, A., Basawapatna, A.R., Escherle, N.A.: Principles of Computational Thinking Tools. Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations. Springer, pp. 291–305 (2017)
6. Nash, C., Blackwell, A,: Tracking Virtuosity and Flow in Computer Music. Proceedings of the Interational Computer Music Conference (ICMC) (2011)
7. Nash, C., Blackwell, A,: Liveness and Flow in Notation Use. Proceedings of the International Conference on New Interfaces for Musical Expression (NIME) (2021), http://www.nime.org/proceedings/2012/nime2012 217.pdf
8. Nash, C., Blackwell, A,: Flow of Creative Interaction with Digital Music Notations. Oxford University Press (2014), https://doi.org/10.1093/oxfordhb/9780199797226.013.023
9. Green, T., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages Computing, vol. 7, no. 2, pp. 131–174 (1996)
10. Csikszentmihalyi, M.: Flow and the Psychology of Discovery and Invention. Harper Perennial, vol. 39 (1997)
11. Peterson, J. Schubert, E.:"Music Notation Software: Some Observations on its Effects on Composer Creativity, Proceedings of Intercational Conference on Music Communication Science (ICoMCS), vol. 127-130 (2007)
12. Roberts, T. L., Moran, T. P: The Evaluation of Text Editors: Methodology and Empirical Results. Communications of the ACM, 26(4), pp. 265–283 (1983)
13. Card, S., Moran, T., Newell A.: Computer text-editing: An information-processing analysis of a routine cognitive skill. Cognitive Psychology, Volume 12, Issue 1, p. 32–74 (1980)
14. Kieras, D., Butler, K.: Task Analysis and the Design of Functionality. The computer science and engineering handbook, 23, pp. 1401–1423 (2014)
15. Holleis, P., Otto, F., Hussmann, H., Schmidt, A.: Keystroke-Level Model for Advanced Mobile Phone Interaction. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07). pp. 1505–1514 (2007), https://doi.org/10.1145/1240624.1240851
16. Abdulin, E.: Using the Keystroke-Level Model for Designing User Interface on Middle-sized Touch Screens. CHI'11 Extended Abstracts on Human Factors in Computing Systems, pp. 673–686 (2011)
17. Guerra, E., Kurz, B., Bräucker, J.: An Extension to the Keystroke-Level Model for Extended Reality Interactions. Mensch und Computer, (2022).
18. Pettitt, M., Burnett G., Karbassioun D.: Applying the Keystroke Level Model in a Driving Context. Contemporary Ergonomics 2006, Taylor Francis (2006)
19. Myung, R.: Keystroke-Level Analysis of Korean Text Entry Methods on Mobile Phones. International Journal of Human-Computer Studies, 60(5-6), pp. 545–563 (2004)
20. Al-Megren, S., Khabti, J., Al-Khalifa, H. S.: A Systematic Review of Modifications and Validation Methods for the Extension of the Keystroke-Level Model. Advances in Human-Computer Interaction, 1–26 (2018)