CriticalEd: A Tool for Assisting with the Creation of Critical Commentaries

Caspar Mølholt Kjellberg
Aalborg University
mail@caspark.com

David Meredith
Aalborg University
dave@create.aau.dk

ABSTRACT

The *best text method* is commonly applied among music scholars engaged in producing critical editions. In this method, a comment list is compiled, consisting of variant readings and editorial emendations. This list is maintained by inserting the comments into a document as the changes are made. Since the comments are not input sequentially, with regard to position, but in arbitrary order, this list must be sorted by copy/pasting the rows into place—an error-prone and time-consuming process. Scholars who produce critical editions typically use off-the-shelf music notation software such as Sibelius or Finale. It was hypothesized that it would be possible to develop a Sibelius plug-in, written in Manuscript 6, that would improve the critical editing work flow, but it was found that the capabilities of this scripting language were insufficient. Instead, a 3-part system was designed and built, consisting of a Sibelius plug-in, a cross-platform application, called CriticalEd, and a REST-based solution, which handles data storage/retrieval. A prototype has been tested at the Danish Centre for Music Publication, and the results suggest that the system could greatly improve the efficiency of the critical editing work flow.

1. INTRODUCTION

The *best text* method [1] is commonly applied in the creation of critical music editions by publishers such as the Danish Centre for Music Publication (DCM) [1]. In this method, the editor selects a primary source from the available historic material, based on a judgement of relative quality. Any necessary changes are implemented in the final document using that source as base. A list of comments is compiled by the editor, consisting of variant readings and editorial emendations (see Figure 1). Currently, this comment list is maintained manually by inserting the respective comments into a separate text document as the changes are made in the score. Since the comments are not input sequentially, with regard to position in the score, but in an arbitrary order, this comment list must be sorted by bar number and part by copy/pasting the rows into place—an error-prone and time-consuming process since a substantial score may require thousands of comments.

Bar	Part	Comment
114-115	fl.3 cl.2 cor.1,2	**B**: slur incomplete
114	fg.	notes 2-3: marc. added by analogy with cor.1,2
114	cor.3,4	*ff* added by analogy with the other parts; notes 2-3: marc. added by analogy with cor.1,2
114	vl.1	**C**: note 3: stacc.
114-115	va.	**C**: b.114 note 2 to b.115 note 1: slur
114	vc.	marc. added by analogy with va., cb. and in accordance with **C**
115	fg. cor.3,4	notes 2-3: marc. added by analogy with b.114 (cor.1,2)
115	cor.1,2	notes 2-3: marc. added by analogy with b.114
115	vl.1	added by analogy with vl.2; **B**: note 1: stacc.; **C**: note 3: stacc.

Figure 1. An excerpt from the comment list for DCM's edition of Carl Nielsen's Symphony No. 1 (Op. 7). The comments are sorted by bar and part. The part names can be looked up in DCM's official list of instrument name abbreviations. As an example, 'vl.1' corresponds to 1st violin. The letters in bold indicate the individual sources used during the editing process.

	fl.	**C**: flutes 1 and 2 written on separate staves
	tr.	**B**: trumpets in F instead of trumpets in D
1–124	cl.	**A**: originally written for clarinets in A, cancelled in ink and written for clarinets in B-flat below system
2–6	vc.	**C**: *col Basso*
2	vc. cb.	**A, B**: rest 3 to note 4: 8-rest 32-note 32-note 32-note 32-note changed to 8p-rest 64-note 64-note 64-note 64-note in pencil (**A**) and ink (**B**); **C**: rest 3 to note 4: 8-rest 32-note 32-note 32-note 32-note
3	ob. cl. fg.	*mfp* added by analogy with b.1 and va.
4	vc. cb.	**A, B**: rest 3 to note 5: 8-rest 32-note 32-note 32-note 32-note 32-note changed to 8p-rest 64-note 64-note 64-note 64-note 64-note in pencil (**A**) and ink (**B**); **C**: rest 3 to note 5: 8-rest 32-note 32-note 32-note 32-note 32-note

Figure 2. A example of the comment list format that is currently sent to the typesetting company.

These shortcomings in the current work flow motivated the development of a software solution that would improve the efficiency of the process of creating and editing a critical commentary. In particular, the editors desired a link between the notation software and the comment list, making it possible to look up the individual comments in the score, helping them determine whether the respective changes have actually been implemented. In the future, the architecture of the system will also be able to support embedding the variant readings and emendations in MEI format [2]. Since the development of MEI has been heavily motivated by musical scholars who work with critical

[1] http://www.kb.dk/en/nb/dcm

editing, MEI includes many features relevant to the topic. It is also an open format, and it is already used in DCM's MerMEId [3, 4], a system for storing metadata about musical works. By presenting this data accordingly, it would be possible to take a step towards fully multidimensional, digital critical editions [5], in which the user is able to select and even specify his or her own customized view on the source material. In this paper, we present the design, development and evaluation of our software solution, named CriticalEd.

2. SYSTEM DESIGN AND DESIGN PARADIGM

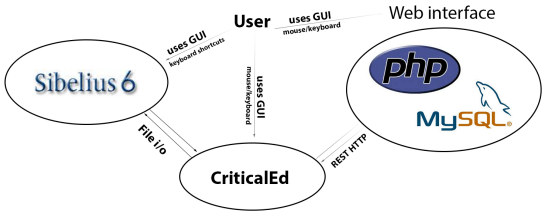


Figure 3. The structure of the CriticalEd system.

As our primary purpose is to produce a tool that can be used at DCM, our proposed solution to the problem of creating and maintaining critical commentaries is a 3-part system, consisting of a Sibelius [6] plug-in, written in Manuscript 6; a helper application, named CriticalEd; and a REST-based [7] solution for data storage/retrieval (see Figure 3). It would not have been possible to construct such a system using only Sibelius' own scripting system, Manuscript, since it has no support for event handling. Luckily, Manuscript has support for basic file input/output operations, making it possible to communicate with other applications through a command protocol. This led to the idea that by making the requirements for the notation software plug-in lightweight, it may be possible to support more than one notation editor (e.g. Finale [8]). This is a significant point, since different scholars and publishers may use different notation software. In order to be able to support other notation software, the respective software's scripting language must support gathering bar and staff name information, writing and reading files and inserting graphics in the score. It also needs to provide a way to associate values with the graphic objects, if it is to be possible to look up comments in the score. In order to be able to support as many platforms as possible (and, by extension, as many potential users as possible) it was decided that an open, cross-platform approach to the design problem had to be taken.

The helper application, CriticalEd, was written in JUCE [9], a C++ library that supports multiple platforms and has an extensive feature set. Because of its open nature, extremely large function set and large user base, it was chosen to develop the REST interface for storing/retrieving data from the database in PHP, using a MySQL database. The web interface is written in HTML5.

3. INTERFACE DESIGN

The interface design is divided into three significant parts, Sibelius, CriticalEd and a web-based interface for viewing and editing the comment list.

Sibelius allows for mapping plug-ins to user-defined keyboard shortcuts. This is utilized to achieve tight integration with Sibelius. When the defined keyboard shortcut is pressed, the command file is changed, causing the respective dialogue to pop up in front of Sibelius. The user selects a passage in the score, then clicks the keyboard shortcut. This causes the "Add Comment" dialogue to pop up, with bar and part information already filled in, as can be seen in Figure 4. The interface for editing individual comments in the score has a similar design. The web interface is designed to be as transparent as possible, and it is written in HTML5. In order to make the system easier for users to adapt to, the structural layout of the data is made to be a tabular layout that resembles that of the documents currently used by the staff at DCM (see Figure 5). In the future, a more sophisticated view on the data will be designed and implemented in HTML5.

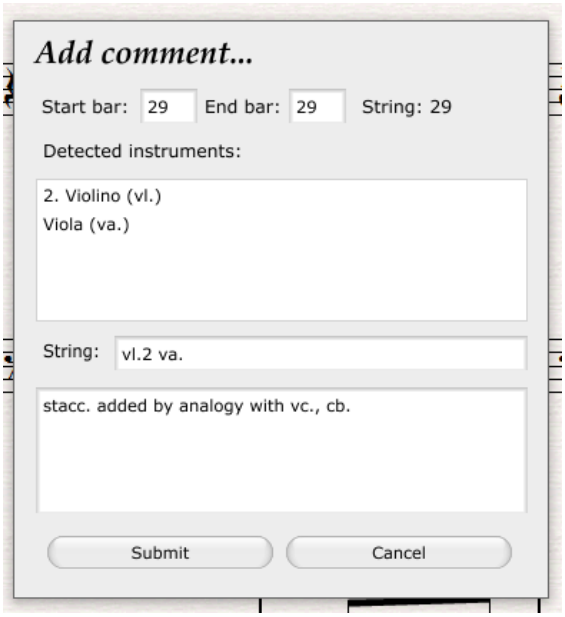
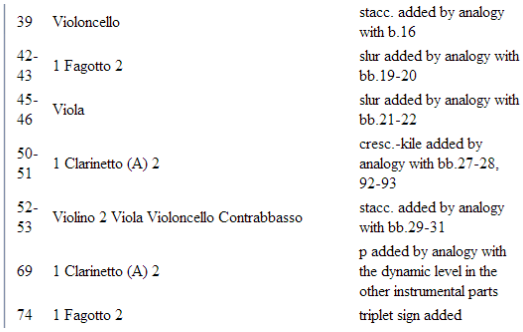


Figure 4. The dialogue presented when adding a comment. Information about the affected instruments and the position has been automatically filled in based on the current selection in Sibelius.

4. INTER-PROGRAM COMMUNICATION

Since the Manuscript scripting language has basic file input/output capabilities, we decided to design a lightweight communications protocol around this. A repository directory is created, in which a set of files are created, one for each type of command. Using the cross-platform, open-source FileWatcher library (`http://sourceforge.net/projects/fwutilities/`), a process running on another thread then watches the repository for file operations. When a file operation is detected (e.g. "addComment.cmd" was changed), a message is sent to CriticalEd's



39	Violoncello	stacc. added by analogy with b.16
42-43	1 Fagotto 2	slur added by analogy with bb.19-20
45-46	Viola	slur added by analogy with bb.21-22
50-51	1 Clarinetto (A) 2	cresc.-kile added by analogy with bb.27-28, 92-93
52-53	Violino 2 Viola Violoncello Contrabbasso	stacc. added by analogy with bb.29-31
69	1 Clarinetto (A) 2	p added by analogy with the dynamic level in the other instrumental parts
74	1 Fagotto 2	triplet sign added

Figure 5. A screenshot of the comment list currently generated by the PHP script. The first column shows the bar number, the second column the part name and the third column the actual comment text.

CommandInterpreter, which then takes the appropriate action (e.g. launch the "Add Comment" dialogue).

The file is then read, inserting each token-separated command parameter into a string array. Depending on the type of command, the parameters are passed on as needed. For an "Add Comment" operation, this includes bar and part data gathered by the plug-in from the current selection.

For some operations, it is necessary to confirm whether the comment was actually added to the database, in order to ensure consistency between the comment list and the score. This means that the software is locked by the plug-in while the operation is carried out in CriticalEd, checking for changes to the response.cmd file. Upon finishing the dialogue in CriticalEd, a corresponding message (e.g. SUBMIT_SUBMITTED, SUBMIT_CANCELED or SUBMIT_FAILED) is sent to Sibelius through response.cmd. Before initiating the transaction, a check is made that a response is not associated with the wrong command at any time. The unique integer identifier of the comment is also sent to Sibelius, which is added to the graphic objects used for indication as a user parameter, which can later be used to identify which comment indicators are associated with which comments in the score.

5. PARSING DCM INSTRUMENT STRINGS

It is not possible to simply use the output from the plug-in directly because DCM adhere to certain strict formatting guidelines in their publications. The part string has to be parsed and converted to an instrument abstraction in CriticalEd. An example of such a string could be "1 Clarinetto (A) 2", which is the data gathered from a staff containing both the 1st and 2nd A clarinets in DCM's edition of Peter Heise's *Drot og Marsk* [10]. More examples can be seen in Figure 6. In total, four major instrument string variants can be found in this particular work (see Table 1). Each of these variants can also be found with additional information added in parentheses, e.g. (Flauto piccolo) or (A), indicating the transposition of the instrument.

In order to parse these strings, it is first determined which of the four types the string belongs to by checking for the conditions found in the second column of Table 1. Then the string is checked for the "(" character, which

parts
Oboe 1
Flauto 1
Flauto 1
Violino 2 Viola
Violino 1
Violino 2 Viola
1 Clarinetto (A) 2 1 Corno (F) 2 Violoncello
1 Fagotto 2
Contrabbasso
1 Corno (F) 2
Violino 1 Violino 2
Violoncello
1 Fagotto 2

Figure 6. An example of some of the strings that were output during the first prototype test at DCM.

would indicate that extra detail is also present in the string. The length of the string between the parentheses is then checked. If its length is only a single character, it is assumed that it is transposition information. If it is longer, the string is identified as an arbitrary detail string, such as "Flauto piccolo". Finally, all identified instruments per staff are added. This is done for each staff.

Instrument(s)	Identification Method
One numbered	Last character is numeric.
Two numbered	Numeric character at both ends.
One unnumbered	No numeric characters present.
Two unnumbered	Two concurrent space characters, no numeric characters.

Table 1. The instrument string variants found in *Drot og Marsk* [10].

6. IN-SCORE COMMENT INDICATION

The DCM staff desired clear visual in-score indication of comments in Sibelius. It was discovered that it is possible to add graphics to individual bars using Manuscript. This is done by calling the AddGraphic method on the affected bar object. The graphics must be in .TIFF format, and it is possible to change the x- and y-offset and scaling factors.

In order to clearly indicate where comments have been inserted, sets of nine images are used, consisting of four corners and vertical and horizontal bar lines for the edges. An example of a 2-by-2 comment can be seen in Figure 7. An icon has also been designed for single bar comments. Several sets of images have been generated, each in a different, strong colour theme. If more than one comment begins at a specific bar and part, the x-offset is changed, and another image set is chosen. The indicators are inserted into the score using a dynamic script, written in Manuscript 6.

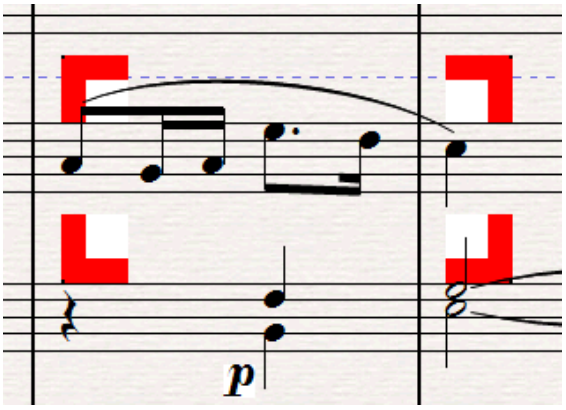


Figure 7. A screenshot from Sibelius of the in-score indicator graphics, here a 2-by-2 comment. Note that the box is extended by vertical and horizontal lines if a larger comment block is selected.

7. DATA STORAGE/RETRIEVAL

Data storage and retrieval is done using a REST-based approach, meaning that all communication is done with HTTP GET/POST/PUT/DELETE requests. The comment-list data is stored in a MySQL database, with storage and retrieval to and from this database being carried out through a RESTful [7] interface written in PHP. The MySQL database may very well be changed to an XML-native database in a future revision, such as eXist-db (`http://www.exist-db.org`). As both PHP and JUCE have excellent facilities for processing XML, the data is passed around in XML format to enable easy handling. This is also done because, in future versions of the software, we want to allow for the possibility of embedding musical snippets into comments in MEI format.

8. EVALUATION

An ethnographic approach was taken during the evaluation of the first prototype, which was used as a proof-of-concept. The prototype was tested on the DCM staff, three senior staff members and a student help. Everything was recorded on audio for later transcription. The participants were given the task of inserting 5 + 25 comments into the score, using the CriticalEd system. The first five comments were used for familiarizing the participant with the software, the remaining 25 for the actual test. Initially, the participant was given a short introduction to the control scheme. Then, the participant was asked to begin inputting the first five comments. Taking note of any annoyances and comments expressed by the staff, the facilitator guided the participant until the first five comments were input satisfactory and then moved on to the larger set of comments. The final stage was timed using a stopwatch.

The test resulted in a set of comments and observations that could be interpreted to define a new set of requirements for the next prototype. Most of the changes were fairly trivial, caused by the early stage of the software, but there were also some important observations and comments.

One participant noted that the instrument definitions were messy and could be improved, a problem that was solved by introducing parsing of the output strings from Sibelius. Another user suggested that icons should be more visible, in response to which the visual indication in Sibelius was made larger and more colour-intensive. Another suggestion was that the system could be used by music teachers to evaluate and annotate written assignments involving the writing of music. The potential of the system to be used for this will be explored in future work. There were also a number of comments from users that suggested that the system was, broadly speaking, fulfilling its intended purpose and could potentially improve work flow (e.g., "Marking scores works really well", "Not a realistic reproduction of our work flow, but it is very exciting, and has potential to improve our work flow.")

9. CONCLUSIONS

Even though the system is currently at a rather immature state, it was concluded, based on the test participants' comments from the first prototype test, that the taken approach is indeed a valid way to tackle the problem of assuring comment list consistency and improving the work flow within teams of scholars creating critical music editions. The information gathered at the next evaluation phase will be used to improve the system further, leading, hopefully, to deployment in the production process at the Danish Centre for Music Publication within the very near future. Having achieved that goal, our next steps will be to (1) add support for other notation software packages (e.g., Finale), (2) explore other possible applications of the software (e.g., as an educational tool) and (3) work towards integrating CriticalEd into a system for authoring fully multidimensional critical editions.

Acknowledgements

David Meredith is partially funded by the EU FET STREP collaborative project grant, "Learning to Create" (Lrn2Cre8). The project Lrn2Cre8 acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 610859.

10. REFERENCES

[1] J. Grier, *The Critical Editing of Music: History, Method and Practice*. Cambridge: Cambridge University Press, 1996.

[2] MEI Council, "Music encoding initiative guidelines (release 2013, revision 2.1.0)," 2013, available online at http://music-encoding.googlecode.com/files/MEI_Guidelines_2013_v2.1.0.pdf.

[3] S. Lundberg and A. T. Geertinger, "MerMEId: Metadata Editor and Repository for MEI Data," 2013, http://labs.kb.dk/editor/. Accessed 1 April 2014.

[4] A. T. Geertinger and L. Pugin, “MEI for bridging the gap between music cataloguing and digital critical edition.” *Die Tonkunst*, vol. 5, no. 3, pp. 289–294, 2011.

[5] F. Wiering, T. Crawford, and D. Lewis, “Digital critical editions of music: A multidimensional model,” in *Modern Methods for Musicology: Prospects, Proposals and Realities*, T. Crawford and L. Gibson, Eds. Farnham, UK: Ashgate, 2009, pp. 23–46.

[6] Avid, “Sibelius,” 2014, http://www.sibelius.com. Accessed 1 April 2014.

[7] R. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[8] MakeMusic Inc., “Finale,” 2014, http://www.finalemusic.com. Accessed 1 April 2014.

[9] J. Storer, “JUCE: Jules’ Utility Class Extensions,” 2014, http://www.juce.com. Accessed 1 April 2014.

[10] P. Heise, *Drot og Marsk*, N. B. Foltmann, P. Hauge, N. Krabbe, and A. T. Geertinger, Eds. Copenhagen: Edition·S/The Royal Library (Danish Centre for Music Publication), 1878/2013, libretto by C. Richardt.