A SEMI-AUTOMATED WORKFLOW PARADIGM FOR THE DISTRIBUTED CREATION AND CURATION OF EXPERT ANNOTATIONS

Johannes Hentschel[1], **Fabian C. Moss**[1], **Markus Neuwirth**[2], **Martin Rohrmeier**[1]
[1] École Polytechnique Fédérale de Lausanne, Switzerland
[2] Anton Bruckner University Linz, Austria
johannes.hentschel@epfl.ch

ABSTRACT

The creation and curation of labeled datasets can be an arduous, expensive, and time-consuming task. We introduce a workflow paradigm for remote consensus-building between expert annotators, while considerably reducing the associated administrative overhead through automation. Most music annotation tasks rely heavily on human interpretation and therefore defy the concept of an objective and indisputable ground truth. Thus, our paradigm invites and documents inter-annotator controversy based on a transparent set of analytical criteria, and aims at putting forth the consensual solutions emerging from such deliberations. The workflow that we suggest traces the entire genesis of annotation data, including the relevant discussions between annotators, reviewers, and curators. It adopts a well-proven pattern from collaborative software development, namely distributed version control, and allows for the automation of repetitive maintenance tasks, such as validity checks, message dispatch, or updates of meta- and paradata. To demonstrate the workflow's effectiveness, we introduce one possible implementation through GitHub Actions and showcase its success in creating cadence, phrase, and harmony annotations for a corpus of 36 trio sonatas by Arcangelo Corelli. Both code and annotated scores are freely available, and the implementation can be readily used in and adapted for other MIR projects.

1. INTRODUCTION

Labeled datasets are an essential prerequisite for many tasks in Music Information Retrieval (MIR) and Digital Musicology, and those tasks are directly dependent on the quality of the labels. Thus, great care needs to be taken during data creation and curation processes in order to provide reliable data for algorithmic evaluation and other purposes. However, the procedures underlying data creation processes as well as how data quality is assessed and assured are not always made explicit and well-documented, a fact which consequently impedes *post hoc* quality control and reproducibility.

The literature on annotation workflows is sparse and widely dispersed, and the description of data, their metadata, and creation processes is commonly tailored to specific datasets and research questions. Generic procedures that emphasize common aspects and steps in labeling pipelines are rare. Fortunately, recent years have seen an increasing number of publications directly addressing these issues for MIR contexts, and researchers are more and more actively describing and documenting the procedures that lead to the creation of datasets, along with discussions of the challenges faced and proposed solutions [1–6]. In particular, publications presenting dedicated datasets [7–13] or workflows [14, 15] discuss these aspects in more detail and present the solutions adopted for the specific research purposes. Standardized solutions for the wider community, however, do not yet exist, both due to the diverse and specific requirements for different data and to the relatively high workload and generally low recognition associated with documentation. The ability to rely on established workflows thus would liberate researchers from this arduous resource- and time-consuming responsibilities.

We propose a viable solution that addresses these issues by introducing a novel workflow paradigm for the distributed production of expert or crowd-sourced annotations that is easy to adopt and adapt. It has the overarching aim to streamline the annotation procedure by reducing the associated administrative overhead. Our workflow is designed to optimize the trade-off between working hours spent and data quality achieved, with the goal to maximize trustworthiness and usability of the resulting annotation labels. We demonstrate its effectiveness through an example implementation [1] by means of GitHub Actions which we use to create cadence, phrase, and harmony annotations for a corpus of scores of 36 trio sonatas by Arcangelo Corelli [2].

Our proposal may serve as a starting point for a wider discussion in the MIR community about how to optimize annotation tasks on a larger scale according to a set of agreed-upon criteria.

[1] github.com/DCMLab/dcml_annotation_workflow
[2] github.com/DCMLab/corelli

2. SETTING

This paper addresses distributed settings in which music experts generate symbolic research data by annotating given music representations such as scores, recordings, or timelines. In this section we describe (1) the different roles of individuals involved, (2) some requirements for the annotations that are to be produced, and (3) common steps in an annotation workflow. Finally, we (4) highlight some concrete problems pertinent to these aspects, motivating our novel workflow paradigm.

2.1 Roles

Individuals involved in the annotation of datasets for MIR tasks can assume one or several of the following roles:
Annotators: experts who provide labels after having familiarized themselves with a set of annotation guidelines.
Reviewers: experts who proof-read the provided annotations and provide feedback, criticism, or corrections.
Curators: individuals who are responsible for the initiation, planning, coordination, and/or financing of the project and whose tasks might also involve the creation and maintenance of annotation guidelines and standards (e.g. through the use of ontologies, vocabularies, or syntax specifications).
Annotators, reviewers, and curators all benefit from the workflow we propose, since it aims to streamline their interaction and to reduce their workload.

2.2 Labels

We use the words annotations and labels interchangeably and in their widest sense, meaning that they could, for instance, follow a pre-defined syntax, pertain to a closed vocabulary, or represent score reductions, graphs, trees etc. Our workflow paradigm is applicable in projects for which the following general assumptions hold: (a) annotation labels are encoded and stored as plain text in order to allow for transparent version control; (b) each label thus provided uniquely refers to a specific segment in the music, such as a range of bars, a set of events, or a time-span; (c) each label corresponds to a precisely formalized encoding scheme, structured vocabulary, or other annotation standard that can be algorithmically validated; (d) annotators and reviewers share the goal to bring forth a final version of labels that reconciles their particular musical expertise with the analytical guidelines underlying the project; (e) the creation and curation of annotated datasets is an inherently open-ended process and one must be able to potentially subject the data to future changes, in particular when the curators introduce changes in the annotation guidelines or syntactic specifications; (f) access to the full history of each label makes a dataset's provenance transparent and increases its trustworthiness.

2.3 Annotation process

Usually, it is the curators' responsibility to clearly define the annotation task(s), to assemble and organize the music representations to be annotated ('original data'), to set up the project infrastructure, and to engage a pool of experts in the endeavor. Depending on the setting, annotators and reviewers need to be familiarized with the tasks, processes, tools, and specific guidelines, often supported by tutorials, training videos, trial phases, or individual coaching. The original data needs to be made available and assigned to (or self-assigned by) annotators, and annotation data to reviewers. The latter assess the quality of the provided labels, e.g. on a case-by-case basis or by applying specific sampling criteria. Finally, curators may check further samples to ensure highest possible quality, and admit the proof-read and validated labels to the final stage, which might coincide with publication.

2.4 Problem statement

The problem we address with this publication is the optimization of the portrayed annotation process in terms of human resources and data quality. We consider this process complete as soon as all envisaged annotations have been created, validated, and verified at least once (for details, see Section 3). At any given moment, the latest validated version of the dataset should be retrievable and the full history of the data genesis, including the provenance of every label, must be stored for maximum transparency.

Most music annotation tasks rely heavily on human judgement and are thus to a large extent subject to interpretation [5, 16–21]. MIR as well as other domains in need of great amounts of subjective annotation data have long since turned to crowd-sourcing as a means of leveraging a massive inexpensive labor force, a paradigm that entails a whole range of well-known problems with respect to quality control, amongst others [22–25]. By valuating quality over quantity, however, our approach favors soliciting fewer and appropriately remunerated experts. This requires diligence and careful organization to ensure outcomes that are effective regarding the tasks to be accomplished as well as economically feasible.

Annotation tasks moreover require appropriate technological infrastructure that allows annotators and reviewers to debate their interpretations within the scope set forth by the annotation guidelines, and to subsequently incorporate the consensual labels in the dataset. This process of consensus-building between experts, proposed in [17], needs to be recorded for future reference.

Finally, curating such an endeavor 'manually', i.e., by exchanging commissions, files, and arguments (e.g. via e-mail), creates a strong desire for a workflow paradigm that easily automates repetitive maintenance tasks. These laborious tasks include the dispatch of notification messages, data validation, and updates of metadata, paradata, and data facets. Not only are these tasks time-consuming for annotators, reviewers, and curators alike, they also bear the danger of being oblivious of issues that require crucial attention. Thus, a system must be put in place that prevents important production steps from being forgotten.

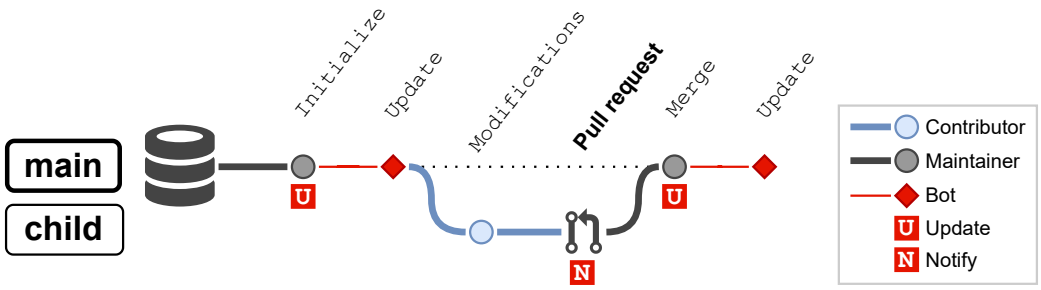

Figure 1. Basic branching scheme. The main branch of the repository exposes the latest version of the dataset that has been approved by a maintainer, whereas contributors can commit only to child branches. Circles represent commits by humans, red diamonds commits by bots, and red squares with a white letter represent triggered scripts.

3. A NOVEL ANNOTATION WORKFLOW PARADIGM

The problems and characterizations of the annotation process stated in the preceding section exhibit a large overlap with the domain of collaborative software engineering [26]. Both describe distributed settings in which a potentially great number of contributors collaborate on creating a possibly large collection of texts: a codebase in the case of software, a dataset of annotations in the present case. Both scenarios can be regarded as open-ended tasks since there is commonly no clearly defined final state; continuous development, possibly based on user feedback, is the norm rather than the exception [27]. Quality control plays a crucial role for the creation of both software and annotations, and is often aided by guidelines, mutual reviews, and automated tests. Moreover, keeping a version history is required for compatibility in software and for reproducibility in data evaluation. Consequently, adopting best practices from distributed version control presents itself as a naturally viable solution. Throughout this section we use terminology that has partly been coined and popularized through the version control system Git [28, 29], but our proposed workflow paradigm can equally well be implemented with similar systems such as Mercurial or Subversion.

3.1 Distributed version control

Our workflow paradigm builds on a well-proven pattern from distributed version control, namely parallel branching [26, 30]. The patterns that we describe in this section are generic in the sense that the involved concepts can be understood as abstract classes which can manifest and be implemented in many different ways.

The basic principle is shown in Figure 1 that displays the version history of a data 'repository' along a timeline. Each new version is created by a 'commit', that is, a set of changes applied to the previous version by an 'author' and summarized in a 'commit message'. In the sketch, commits by human authors are shown as circles and those made by bots as diamonds. Each of the two horizontal levels represents a 'branch', of which, in principle, there can be infinitely many in parallel. Every child branch branches off directly or indirectly of the repository's main branch, or 'trunk', and the commits it contains can be merged back into it anytime. We refer to individuals with the permission to merge, or 'pull', changes into the main branch as 'maintainers'.

All other human authors, or 'contributors', do not have permission to merge into `main` and therefore need to issue a 'pull request' that may or may not be accepted by a maintainer based on their review of the commits on the child branch. Maintainers therefore keep full control over the trunk which exposes at all times the latest version of the repository in which all commits have been made, or approved by them. This is a design choice for a scenario where one entity (person or institution) guarantees the integrity of the repository's main branch (e.g., with respect to a stipulated set of guidelines) while allowing for external contributions in a controlled manner.

3.2 Data maintenance

The red elements in Figure 1 express automation. Specifically, squares represent scripts that are triggered by certain events, and which may result in a bot pushing a commit. For instance, the script `Notify` simply dispatches automated messages to the relevant persons to inform them about a newly issued pull request, so that it does not go unnoticed. The `Update` script is triggered upon push to the main branch and uses a bot to commit its outputs. In the case of an annotation workflow such updates most typically comprise (a) **metadata**, e.g., amount, type, and format of annotation labels, information on the annotated pieces, free text descriptions; (b) **paradata**, e.g., annotator identities, name and version of the employed annotation software, review status, time stamp; or (c) **data facets**, e.g., extraction/separation of particular features for increased accessibility, or summary statistics for individual pieces or the entire dataset. For a concrete example, see Section 4.4. Including an `Update` routine immensely facilitates the project coordination—e.g. through an always up-to-date dashboard or README file—and allows one to use the dataset in its current state at any point in time.

3.3 Data validation

Figure 2 exemplifies the workflow paradigm for the completion of one (partial or comprehensive) set of annota-

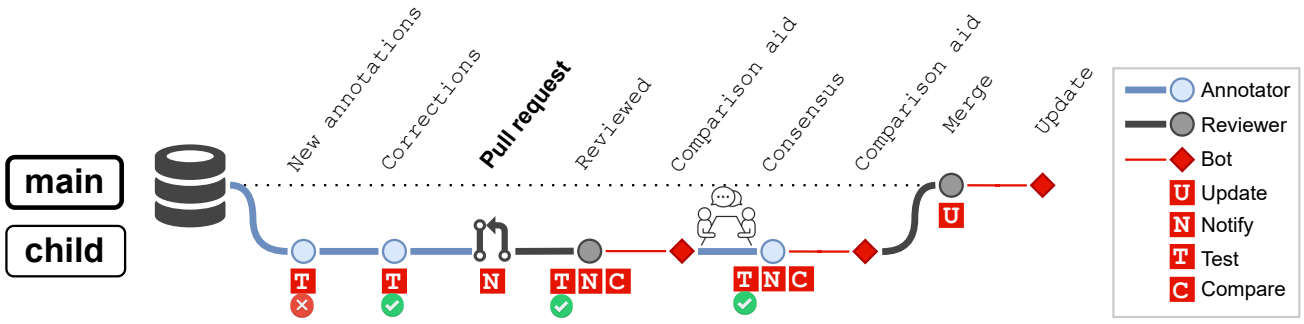

Figure 2. Example for a complete annotation cycle. In a parallel branch, an annotator pushes new annotations that don't pass the automatic validation (`Test`), corrects the errors, and issues a pull request. As soon as a reviewed version is pushed, `Compare` adds files to aid the annotator with the process of going through the changes and deliberating with the reviewer.

tions, including data verification (see the follow subsection) and validation. Since we established in Section 2.2 that the annotation labels may be automatically validated, we included the script `Test` which is triggered upon every commit to any child branch (except if performed by a bot). This way we ensure that syntactic and orthographic errors in the annotations can be corrected right away and we disallow merging into the main branch in cases where the validity test did not pass. This way we ensure that invalid data cannot make its way into the main branch.

3.4 Data verification through expert consensus

As a consequence of the subjective nature of annotations, data verification can not completely be delegated to algorithmic evaluation. Conceptually, we substitute for the concept of an objective and indisputable ground truth the idea of consensual solutions based on transparent deliberation. The data verification procedure begins with the annotator issuing a pull request (see Figure 2) for merging a set of new annotations into the main branch, thus setting off the `Notify` script (see Section 3.2). From here on, every additional commit is included in the open pull request and triggers (except if performed by a bot) an additional third script, here called `Compare`. In the example, the reviewer updates the proposed annotations, correcting obvious errors and substituting diverging musical interpretations. Pushing these changes causes (a) `Test` to validate the reviewer's changes, (b) `Notify` to inform the annotator about the completed review, and (c) `Compare` to create one or several files that may aid the annotator to retrace the reviewer's changes easily, for instance through a diff file, a revision report, or a compilation of reviewer comments. In case the annotator agrees with all changes, the new set of annotations is considered to represent a consensus between the two experts. Otherwise, this consensus is to be reached through transparent deliberation, i.e., a recorded exchange of arguments (symbolized in Fig. 2 by the discussion table), at the end of which the annotator pushes the consensual solution, whereupon the reviewer or anyone with the relevant permissions may merge the thus verified data, completing this subset. Consequently, every verified label represents a consensus between at least two experts.

4. GITHUB ACTIONS IN ACTION: THE GENESIS OF A NOVEL DATASET

To demonstrate our paradigm's effectiveness we implemented one possible instance [1] of the proposed workflow paradigm and showcase its success in creating cadence, phrase, and harmony annotations for a corpus of 36 trio sonatas (opp. 1, 3, and 4) by Arcangelo Corelli. [2] The implementation was created using the code hosting service `github.com` because (a) it is frequently used in music research projects for storing research data; (b) it is well–known and chances are that users and new annotators are already familiar with it; (c) it offers a free plan that includes server run time for automating tasks; and because (d) of its easy-to-use automation capacities which include predefined code patterns (called Actions) as well as custom scripts. Since the automation is defined in configuration files contained in the repository, reusing our proposed implementation can be easily achieved by simply starting from the corresponding template repository. It makes use of the Python library `ms3` [3] to perform tasks on the annotated scores, such as extracting and processing the contained labels, and storing them as tabular TSV files.

4.1 Score annotation: harmony, phrases, and cadences

To create the annotated dataset we commissioned trained music theorists to enter the corresponding labels directly into the provided digital scores, using the latest version of the DCML harmonic annotation standard. [4] It has a predefined syntax that can be automatically validated, and a set of annotation guidelines [5] on the basis of which our contractors were able to put forth consensual solutions. The annotation task was performed using the free and open-source notation software MuseScore because it is well-known to many musicians and theorists and offers one of the most convenient interfaces for digitally annotating scores. Thus, the annotation labels are stored within the original data and can be viewed by opening the MuseScore files or the corresponding annotation tables in TSV format.

[3] `pypi.org/project/ms3/`
[4] `github.com/DCMLab/standards/`
[5] `dcmlab.github.io/standards/tutorial`

4.2 Validation: Automated syntax checks

Putting into practice what was introduced in Section 3, annotators create new annotations by committing changes made in the MuseScore files to side branches of the central repository on GitHub, called 'origin'. Every time a set of such commits is pushed to the origin, a virtual machine is initialized on GitHub's server infrastructure in order to run the above-mentioned `Test` script. It is defined in the configuration file `check.yml` and uses `ms3` to parse the newly annotated files and validate the syntactic correctness of the contained labels. If the validation fails the annotators will be informed instantly and see a list of all syntactically wrong labels together with their exact positions in the score.

4.3 Verification: Automated comparison files

As soon as the new labels have been validated, the annotator issues a pull request, causing GitHub to dispatch notifications to reviewers and curators, and the data verification by a reviewer ensues, as described in Section 3.4. The main problem that our workflow solves at this stage is tracking changes to the proposed set of new annotations individually and in conjunction with the respective justifications. Reviewers are requested to combine modifications into individual commits such that the commit messages include a measure number and the reasoning behind the changes. That way, the original annotators are able to go through these commit messages one by one in order to decide whether they consent to the change or not. In the latter case, they object by engaging in a written exchange of arguments with the respective reviewer. The pull request functionality on code hosting sites such as GitHub ideally supports the subsequent consensus-building process by providing and storing practical and interactive summaries of the comments and commits added in the process as well as by notifying the involved parties about such events.

Although GitHub lets users conveniently visualize the changes made with every commit, a difficulty arises from the fact that, in most cases, it is not sufficient to inspect the source code excerpts from the modified MuseScore files to appraise a changed label. Therefore, our automation script `Compare` adds or updates an additional MuseScore file upon every commit into an open pull request, in which the modifications pertaining to the current verification phase are highlighted with different colors. These files immensely facilitate the discussions about the proposed solutions and may also serve at a later point for documentation purposes or evaluations (see Section 4.5).

As soon as consensus is reached and the new annotations have thus been verified, the corresponding branch is merged into `main` and the pull request is archived, storing and documenting the verification process for the future.

4.4 Maintaining metadata, paradata, and data facets

In our implementation, the `Update` script (see Section 3) is called `extract` because it automatically commits information extracted from the source code of the modified MuseScore files upon every push to the origin's main branch. Further, meta- and paradata are obtained for and copied from the changed MuseScore files and then included in a tabular metadata file summarizing the dataset as well as in the repository's README file. Moreover, annotation labels, notes, and other score elements are extracted and stored as individual tabular files. This mechanism ensures that those facets of the dataset that users will generally be most interested in stay updated and may be loaded, transformed, and evaluated with greater ease, thus maximizing the dataset's accessibility and reusability.

4.5 Workflow evaluation

# changes	count	%	# syntax errors	count
0	5333	62.6	**0**	5333
1	2511	29.5	**0**	2466
			1	45
2	574	6.7	**0**	534
			1	40
3	89	1.0	**0**	72
			1	15
			2	2
4	14	0.2	**0**	12
			1	1
			3	1
5	3	0.0	**0**	1
			1	1
			3	1

Table 1. For 8542 verified labels in the Corelli dataset, we report the label count for the number of changes that they underwent, and how many of these changes addressed syntactic errors rather than inter-expert disagreement. For example, 3 labels were changed 5 times and throughout 6 versions, they saw 0, 1, and 3 syntactic errors respectively.

Our workflow implementation allows for multiple ways of evaluating the annotation procedure, of which we will showcase two examples. First, for every annotated MuseScore file in the Corelli dataset, we extracted all versions from the Git history and tallied the labels and their positions from each. Tracking all occurring positions over the file's entire history allowed us to count the number of changes that the label at each position was subjected to and whether they were required due to syntactic errors or otherwise. The results in Table 1 show that roughly 63 % of labels were considered correct from the start and did not change over the course of a file's history, whereas only 1.2 % needed to be modified more than twice until a consensus was reached. Note that this approach does not reveal whether changes were effected by annotators or reviewers. However, since after a full workflow cycle all labels eventually represent consensual solutions between at least two experts, we can deduce that our workflow is highly efficient in putting forth validated and trustworthy annotation data. The overall rate of labels that has been syntactically wrong at one point is extremely low (1.2 %),

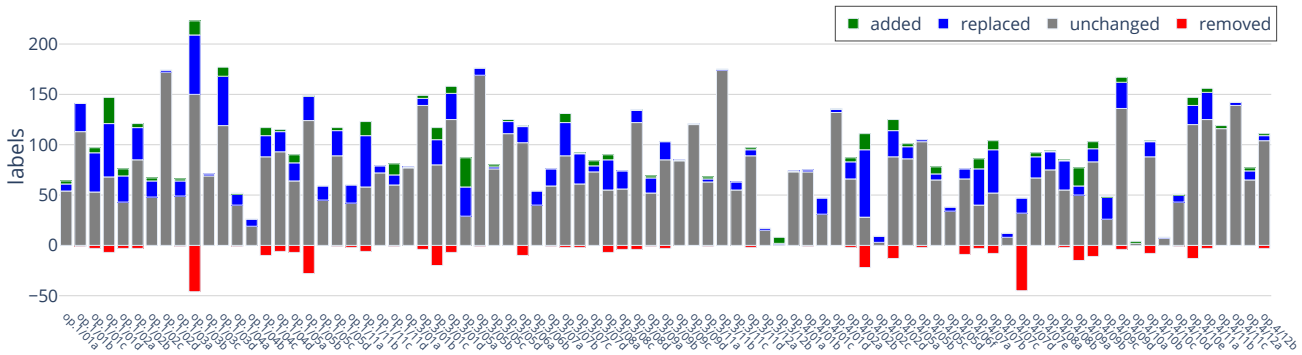

Figure 3. This plot shows for 85 movements how many of the labels have been added, replaced, or left unchanged during the most recent consensus-building phase, and how many have been removed that were previously included. Files showing very few modifications might originate from later minor revisions of verified labels.

suggesting that the guidelines the annotators received were comprehensible and easy to implement. Nonetheless, detecting these kinds of errors automatically constitutes an invaluable advantage, since syntactic validity is the minimal requirement for usable annotations.

For our second evaluation we exploited the annotated MuseScore files that were automatically generated during every verification procedure. They show the difference between exactly two versions, namely between the one that the reviewer started off with and the verified version that represents a consensus between reviewer and annotator. These files maintain the labels from the previous version and show the differences through a color coding, namely red for deleted labels and green for substituted or added labels. This color coding is indepentend of who committed the modifications, since the most recent version always represents the consensus.

Figure 3 shows an evaluation of the changes made to the 8524 labels of 85 movements from the trio sonatas. We interpreted co-occurences of a green and a red label at the same position as replacements, which we show in blue and subtract from the green and red bars. Labels that were removed during the verification are shown in the negative range so that the positive range reflects the *status quo*.

A closer analysis of the changed content of the labels by substitution or replacement shows that these often entail music-theoretically fine distinctions, such as whether a chord should be interpreted as `V7/IV` or `I7`. Both have the identical absolute surface realizations (e.g. a C7 triad in the key of C major) but their relative, hierarchical interpretations differ. Another example would be `V(6)` versus `iii6`, i.e. a dominant triad with a suspension of its fifth or a minor triad on the third scale degree in first inversion. Again, both chords are identical in terms of their pitch-class content but differ with respect to their harmonic function. A full and detailed analysis of these changes is beyond the current scope and left for future research.

While one could have assumed that annotations of harmony, phrases, and cadences is a relatively straightforward task for music theory experts, our evaluation reveals that in many cases considerable modifications are necessary to arrive at an agreed-upon solution. This result emphasizes the need for broader studies on inter-annotator (dis-)agreement [3, 13, 20] and moreover corroborates the weak status of 'ground truth data' for annotations with a high degree of interpretability [5, 10–12, 14, 16–19].

5. CONCLUSIONS

In this contribution we proposed a semi-automated workflow paradigm for streamlining the creation of annotated datasets by experts, and introduced, demonstrated, and evaluated one possible implementation. It bridges a gap between two by and large distinct skill sets: on the one hand researchers with expertise in computational methods and paradigms, and on the other hand expert music theorists and musicologists who contribute their considerable domain knowledge but may lack technical prowess.

Our proposal overcomes this 'communicative barrier' by providing a clearly defined workflow for the creation of annotated data that requires on behalf of the domain experts only the comprehension of the branching model outlined above and the usage of graphical user interfaces for label entry and revisions. Whereas our proposal greatly reduces the workload of annotators and reviewers, too, the automated notifications and validations, as well as the ease of communication and discussion, renders the curators its main beneficiaries, who usually bear the responsibility for a project's coordination and success. As a proof of concept we have provided with this publication a GitHub repository with a new annotated dataset for which our workflow implementation was used. Future discussions within the MIR community may illuminate the repercussions of and alternatives to using proprietary hosting services in terms of cost, functional range, and data longevity/security.

Although our case study (building, providing, and evaluating corpora of annotated scores) is somewhat specific, we believe that a wide range of research projects will benefit from adopting or adapting it. We welcome alterations or alternative proposals, trusting that an active and constructive discussion around the topics laid out in this paper is valuable for the consolidation of data creation and annotation practices in the MIR community.

6. ACKNOWLEDGEMENTS

This research was supported by the Swiss National Science Foundation within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (Grant no. 182811). This project is being conducted at the *Latour Chair in Digital and Cognitive Musicology*, generously funded by Mr. Claude Latour.

7. REFERENCES

[1] C. McKay, I. Fujinaga, M. Müller, and F. Wiering, "Building an Infrastructure for a 21st-Century Global Music Library," in *Proceedings of the 16th International Music Information Retrieval Conference*, Malaga, Spain, 2015, pp. 1–2.

[2] J. Yaolong, S. Howes, C. McKay, N. Condit-Schultz, J. Calvo-Zaragoza, and I. Fujinaga, "An Interactive Workflow for Generating Chord Labels for Homorhythmic Music in Symbolic Formats," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, Nov. 2019, pp. 862–869.

[3] J. Degroot-Maggetti, T. de Reuse, L. Feisthauer, S. Howles, Y. Ju, S. Kokubu, S. Margot, N. N. López, and F. Upham, "Data Quality Matters: Iterative Corrections on a Corpus of Mendelssohn String Quartets and Implications for MIR Analysis," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, Online, 2020, pp. 432–438.

[4] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, "Scores of scores: An openscore project to encode and share sheet music," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology - DLfM '18*, Paris, France, 2018, pp. 87–95.

[5] J. Devaney, "Using Note-Level Music Encodings to Facilitate Interdisciplinary Research on Human Engagement with Music," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 205–217, Oct. 2020.

[6] D. Lewis, D. Weigl, and K. Page, "Musicological Observations During Rehearsal and Performance: A Linked Data Digital Library for Annotations," in *6th International Conference on Digital Libraries for Musicology*, ser. DLfM '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–8.

[7] H. Schaffrath, *The Essen Folksong Collection*, D. Huron, Ed., Stanford, CA: Center for Computer Assisted Research in the Humanities.

[8] M. Kemal Karaosmanoğlu, "A Turkish makam music symbolic database for music information retrieval: SymbTr," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012, pp. 223–228.

[9] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "Mirdata: Software for Reproducible Usage of Datasets," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019.

[10] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets," *Frontiers in Digital Humanities*, vol. 5, no. July, pp. 1–5, 2018.

[11] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. G. Grohganz, "Schubert Winterreise Dataset: A Multimodal Scenario for Music Analysis," *Journal on Computing and Cultural Heritage*, vol. 14, no. 2, pp. 25:1–25:18, May 2021.

[12] J. Albrecht and D. Shanahan, "Can I Have the Keys?: Key Validation Using a MIDI Database," in *Proceedings of the 14th International Conference for Music Perception and Cognition*, San Fransico, California, 2016, pp. 752–755.

[13] J. B. L. Smith, J. Ashley Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and Creation of a Large-Scale Database of Structural Annotations," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 555–650.

[14] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, and D. Tidhar, "OMRAS2 metadata project 2009," in *Late-Breaking Session at the 10th International Conference on Music Information Retrieval*, Kobe, Japan, 2009.

[15] F. Simonetta, S. Ntalampiras, and F. Avanzini, "ASMD: An automatic framework for compiling multimodal datasets with audio and scores," in *Proceedings of the 17th Sound and Music Computing Conference*, Turin, Italy, Apr. 2020, pp. 40–46.

[16] D. Harasim, C. Finkensiep, P. Ericson, T. J. O'Donnell, and M. Rohrmeier, "The Jazz Harmony Treebank," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, Montreal, Canada, 2020, pp. 207–215.

[17] J. Hentschel, M. Neuwirth, and M. Rohrmeier, "The Annotated Mozart Sonatas: Score, Harmony, and Cadence," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 67—80, 2021.

[18] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, "Annotator subjectivity in harmony annotations of popular music," *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.

[19] C. Raffel and D. P. W. Ellis, "Extracting Ground Truth Information from MIDI Files: A Midifesto," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, NY, 2016, pp. 796–802.

[20] A. Selway, H. V. Koops, A. Volk, D. Bretherton, N. Gibbins, and R. Polfreman, "Explaining harmonic inter-annotator disagreement using Hugo Riemann's theory of 'harmonic function'," *Journal of New Music Research*, vol. 49, no. 2, pp. 136–150, 2020.

[21] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Miami, Florida, 2011, pp. 633–638.

[22] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, Jan. 2011.

[23] A. T. Nguyen, M. Halpern, B. C. Wallace, and M. Lease, "Probabilistic Modeling for Crowdsourcing Partially-Subjective Ratings," *4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pp. 149–158, 2016.

[24] A. Dumitrache, "Truth in Disagreement: Crowdsourcing Labeled Data for Natural Language Processing," Ph.D. dissertation, Amsterdam University, 2019.

[25] M. Kutlu, T. McDonnell, M. Lease, and T. Elsayed, "Annotator Rationales for Labeling Tasks in Crowdsourcing," *Journal of Artificial Intelligence Research*, vol. 69, pp. 143–189, Sep. 2020.

[26] A. Leon, *Software Configuration Management Handbook*, 3rd ed. Boston & London: Artech House, 2015.

[27] J. Humble, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, ser. The Addison-Wesley Signature Series. Upper Saddle River, NJ: Addison-Wesley, 2011.

[28] T. Swicegood, *Pragmatic Version Control Using Git*, ser. The Pragmatic Starter Kit, S. Davidson Pfalzer, Ed. Raleigh, North Carolina: The Pragmatic Bookshelf, 2008, no. 1.

[29] E. J. Hogbin Westby, *Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git*. Beijing: O'Reilly, 2015.

[30] B. Appleton, S. P. Berczuk, R. Cabrera, and R. Orenstein, "Streamed Lines: Branching Patterns for Parallel Software Development," in *Proceedings of the 1998 Pattern Languages of Programs Conference*, Monticello, Illinois, USA, 1998, pp. 1–67.