Towards Feature-based Versioning for Musicological Research

Paul Grünbacher
Institute of Software Systems Engineering
Johannes Kepler University Linz
4040 Linz, Austria
paul.gruenbacher@jku.at

Markus Neuwirth
Institute for Theory and History
Anton Bruckner Privatuniversität Linz, Austria
4040 Linz, Austria
markus.neuwirth@bruckneruni.at

ABSTRACT

This paper discusses the management of revisions and variants of musical works for the context of musicological research. Domain-specific languages (DSLs) are a fundamental tool in music notation and analysis, as they enable the notation of music and also support music analysis when investigating particular structural properties of melody, harmony, rhythm, or form. However, the fields of music philology and music analysis still lack a systematic approach for uniformly managing revisions and variants of musical compositions. This research-in-progress paper proposes the use of feature-based versioning to streamline the management of revisions and variants in music artifacts. We introduce an illustrative example and present research challenges regarding variability and feature-based versioning for musicological research. We present a preliminary approach which involves mapping features to specific parts of musical works and musical analyses, thereby facilitating the composition of new variants based on selected features, a prerequisite for enhanced research in music philology and music analysis.

CCS CONCEPTS

• **Software and its engineering** → **Software configuration management and version control systems**; **Software product lines**; **Domain specific languages**; • **Applied computing** → *Performing arts.*

KEYWORDS

music and variability, feature-based version control, domain-specific languages, musicology

ACM Reference Format:
Paul Grünbacher and Markus Neuwirth. 2024. Towards Feature-based Versioning for Musicological Research. In *18th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2024), February 07–09, 2024, Bern, Switzerland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3634713.3634723

1 INTRODUCTION

As is common in an artistic field and typical of any creative process, musical works frequently exist in numerous versions, which often reflects their history and genesis during composition, publication, and performance. Composers, copyists, and editors often markup different revisions of manuscripts, e.g., to correct apparent errors; to indicate the regularization of irregular, non-standard, or eccentric features; or to handle editorial additions, suppressions, and omissions. In particular, versions are either *revisions* caused by changes or editorial markups made over time, or *variants*, e.g., different editions of the same musical work. While revisions usually concern minor differences (e.g., stem directions), variations may involve more significant differences (e.g., extra measures or omissions of entire sections). A well-known example is the excessive number of revisions and variants of Anton Bruckner's symphonies, which, in the case of the Third Symphony, primarily served to reduce the extraordinary length of the work's original version.

As a result, revisions and variants of musical works are a major topic in musicological research: Music philology describes and edits the revisions that exist regarding a particular work of music [14]. Music analysis addresses the variants and patterns, e.g., of motives, formal designs, and voice-leading idioms, that appear within a particular work or across a corpus of works [6]. This may also involve tracing the creation of a musical work in all its recorded details—from the first sketches to the complete text—to investigate processes of compositional thought [30].

Current musicological research adopts domain-specific languages (DSLs) [21, 26, 34] for representing both musical scores and supporting music analyses:

DSLs for musical scores are a prerequisite for the notation of any music, i.e., "visual analogues of musical sound, either as a record of sound heard or imagined, or as a set of visual instructions for performers" [3]. Examples are the Music Encoding Initiative (MEI), MusicXML, LilyPond, or Humdrum. The DSL LilyPond [23] allows engravers to focus on defining the actual music instead of the score's layout and the detailed rules of manual engraving [13]. The MEI is an open-source community effort for encoding musical documents in a machine-readable format [32] and also defines best practices for representing a broad range of musical documents and structures. The MEI guidelines cover the creation of digital scholarly editions of music.

DSLs for music analysis are used by musicologists to encode annotations of musical features such as harmony, meter, or form. Examples of annotations of harmony are the DCML standard [18, 29] and the Romantext Format [12, 33]. The convenience and expressiveness of such DSLs reduce the gap between high-level concepts used by the music analysts and low-level concepts needed for engraving music and implementing automated analyses.

As pointed out, the DSL code of musical artifacts evolves in time (leading to *revisions*) and in space (leading to *variants*), thus resulting in challenges for version control and variability management. For instance, analyzing the genesis and history of compositions may require creating multiple variants and score layouts of the

VaMoS 2024, February 07–09, 2024, Bern, Switzerland

ACM ISBN 979-8-4007-0877-0/24/02.
https://doi.org/10.1145/3634713.3634723

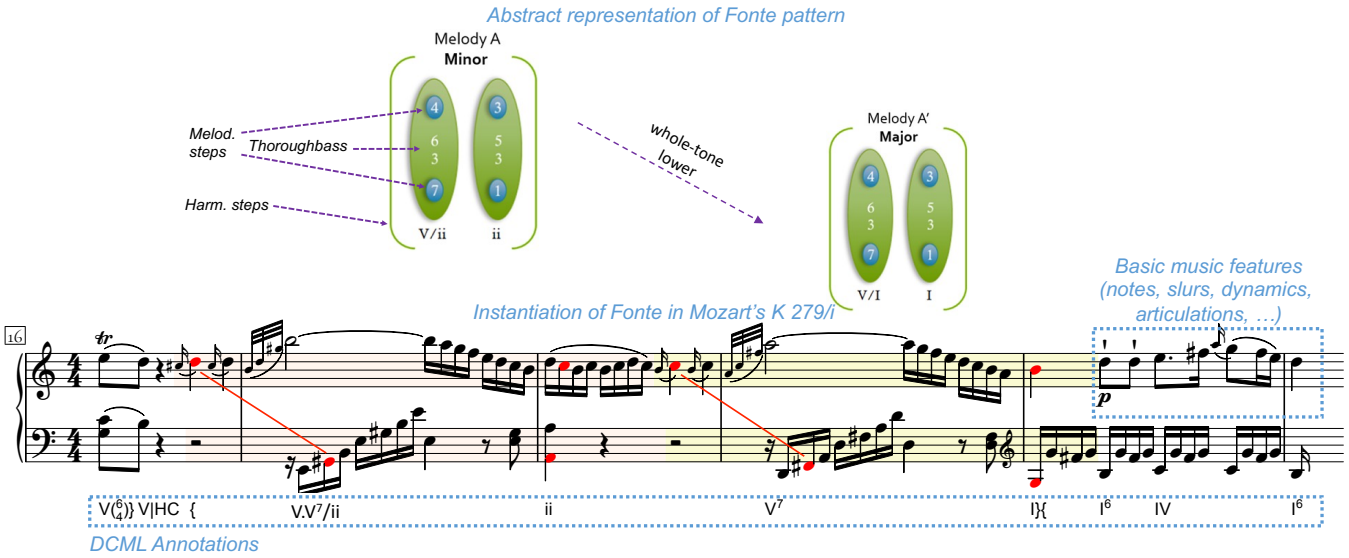

Figure 1: The excerpt from Mozart's K. 279/i covers basic music features like notes, slurs, and dynamics; features for music analysis including DCML annotations of tonal harmony; and a possible instantiation of the Fonte voice-leading pattern represented in abstract form above the score.

same musical work, and the variants and revisions may exist at different levels of granularity. At the same time, musicologists need to manage variants and revisions of annotations as well as editorial markups and interventions [32].

In the field of software engineering, *features* are used to manage both revisions and variants of digital artifacts [1, 10]. In particular, feature-oriented techniques map features to the parts of the artifacts realizing them. These mappings can then be used to compose new variants of the artifacts based on a selection of the desired features. However, this remains challenging, as the mappings cannot be defined fully automatically, and feature characteristics such as their granularity, scattering, and degree of interaction have a major impact on the process of composing them [19, 20, 35].

Considerable advances have been made in modeling and analyzing the genesis and versions of musical works (e.g., Beethoven's Werkstatt [8, 22] or Bruckner's compositional studies [31]). Also, standards for encoding music explicitly state requirements for managing versions [32]. However, a systematic, feature-based approach to uniformly managing both revisions and variants has so far not been investigated in the area of musicological research. Thus many opportunities arise for improving the state-of-the-art of managing revisions and variants in the fields of music philology and analysis.

This research-in-progress paper motivates our ongoing research based on an illustrative example (Section 2). We then outline research challenges on variability and feature-based versioning for musicological research (Section 3). We describe our preliminary approach, illustrated with examples. Finally, we provide an outlook on our future work.

2 ILLUSTRATIVE EXAMPLE: MUSIC FEATURES AND VERSIONING

The question of what constitutes a feature generally depends on the application context and the domain of interest [4]. In software engineering, a feature has been described as "a distinguishable characteristic of a concept (system, component, etc.) that is relevant to some stakeholder of the concept" [9]. However, what are the distinguishable characteristics of music relevant to a musicologist?

Musicological researchers and music engravers use DSLs and their accompanying software tools to define scores and score annotations. We illustrate the idea of music features with an excerpt from Wolfgang Amadé Mozart's first piano sonata, K. 279 (first movement, mm. 16–21; see Fig. 1):

After a short break, the second theme opens up with a non-tonic harmony (V^7/ii; m. 17), from which the music leads back to the tonic of the local key of G major (the dominant key) via a descending-fifth sequence (V^7/ii ii V^7 I). These harmonic events are expressed here using the DCML standard for harmonic annotation. A *within-movement variant* of this segment appears in the recapitulation (mm. 69–75; not shown in Fig. 1), which features a number of interesting modifications (e.g., mixed transposition by fourth/fifth, metrical shift by half-bar, and several melodic variations). Furthermore, the excerpts can be seen to represent variation on a different scale, one transcending the piece level: it instantiates a voice-leading *pattern* known as the Fonte, which has been in use since about the last quarter of the 17th century and is still in use today. The upper part of Fig. 1 shows the characteristic harmonic, melodic, and formal features of the Fonte sequence: *Harmonically*, it can be described as a subtype of a descending-fifth sequence (V^7/ii ii V^7 I); *melodically*, it consists of two local $\hat{4}\,\hat{3}$ progressions in the soprano, each contrapuntally combined with local $\hat{7}\,\hat{1}$ motions in the bass (see the notes highlighted red); and *formally*, the Fonte consists of

two halves, the latter of which (in minor) appears one whole-tone lower (in major) than the former [e.g., 11, 28]. The example shows that music features address different aspects:

Basic music features concern properties like pitches, durations, dynamics, or articulations. Variability management for basic music features has been investigated in our earlier research. For instance, we experimented with a feature-based version control system [25] to support the *composition* of music scores encoded in the DSL LilyPond based on music features [16], leaving the annotations required for musicological research out of account. In addition, we studied the variability-rich process of automatically rendering scores to different end-user devices [15]. We benefit from these findings to manage variability and rendering at different levels of abstraction.

High-level music features, on the other hand, concern harmonies, keys, voice-leading schemata, and formal designs, all of which are addressed in the related fields of music theory/analysis, computational musicology, and music corpus research: Harmony annotations, e.g., based on the DCML encoding standard, offer multiple harmonic features such as global and local keys, chord type, root, inversion, missing and added chord notes, and harmonic motion over a pedal note. Voice-leading schemata as a characteristic component of Western musical styles have been explored in the research project "From Bach to the Beatles" [28]. Annotations of musical form could be based on an extended version of an encoding standard introduced by Gotham and Ireland [12]. Based on such expert annotations, large datasets can then be analyzed in order to uncover characteristic stylistic patterns as, e.g., shown in [27].

3 RESEARCH CHALLENGES

Earlier research shows that musical features need to be encoded at different levels of granularity, are scattered across artifacts, and are highly interacting [16]. This leads to three main research challenges when considering the requirements of musicologists:

Understanding features for variability management in musicological research. The ability to trace features in musical artifacts and structures depends on the DSL used to encode the music, the notion of music features, and the workflow of musicologists. While initial studies suggest that feature-oriented techniques can work for basic features [15], it remains unclear to what extent they might be useful for high-level and more complex features. For instance, it is important to model music annotations (e.g., of harmony or large-scale form) as features such that musicologists can then create and analyze different variants of music works. Existing feature taxonomies from musicology [e.g., 5] may serve as a starting point to identify such feature candidates. An important area of our research is to investigate the usefulness of feature-based techniques for musicians, i.e., to find out if the automated integration of variants, as accomplished by feature-based version control systems, allows to derive semantically reasonable features for the discussed use cases. It is also interesting to examine the extent to which the process will be in need to rely on manual interventions.

Representing features with different properties in music artifacts. Manually or automatically relating features to the artifacts realizing them [1] is difficult, as features are often fine-grained and scattered, and may exist at different levels [4]: From an encoding perspective, feature *granularity* concerns the size of individual elements mapped to a particular feature. For instance, in Fig. 1 the notes of the first voice of the left-hand part notated in the bass clef represent a coarse-grained feature, while the articulation on a single note represents a fine-grained feature. Feature *modularity* means the number of different locations a feature is implemented in. For instance, a header with global settings of Mozart's K. 279/i is highly modular and can be defined in a single place only, while the slurs require scattered definitions in non-contiguous places in the code. Further, features need to exist at different *levels* to support versioning of high-level characteristics defined by musicologists via manual or automated annotations (cf. the harmonic annotations and the Fonte pattern in Fig. 1), as well as basic features expressed in a music DSL.

Modeling and analyzing dependencies and interactions between music features. Feature models use a tree-like structure to define the hierarchical relationships between features, with more general features at the top and more specific features at the bottom [10]. Feature models also capture constraints on the features, such as mandatory, optional, or mutually exclusive features. These constraints help to validate the variants generated from an integrated representation of music and annotations. However, to what extent do the assumptions made about feature dependencies also hold in the domain of musicology? For instance, how can the sometimes complex relations between formal sections within a given piece of music be modeled and managed? How are different kinds of features related with each other? What are the core features in music and what are its modifications? An example is the (generally highly flexible) instantiation of the abstract representation of the Fonte pattern shown in Fig. 1. Furthermore, one must consider feature interactions, i.e., the case of one feature modifying or influencing other features in defining overall behavior [35]. Such structural interactions manifest themselves at the code level whenever code needs to be included in a variant because of a combination of selected (or unselected) features [2], i.e., DSL code needs to ensure the correct joint working of the interacting features.

4 APPROACH

In our ongoing research, we aim at developing an approach that comprises the following steps to address the above challenges:

Music Encoding. This step involves the selection or creation of digital representations of scores. For instance, Fig. 2 shows a code snippet of Mozart's K. 279 (first movement, measure 16, right hand) encoded in the music DSL Lilypond. This step is carried out by incrementally committing the basic music features like notes of instruments, dynamics, articulations, etc. to a feature-based version control system.

Music Analysis and Annotation of Scores. Building on the previous step, we use the DCML standard [18, 29], a DSL to add music analytical annotations to the scores (e.g., harmony, voice-leading schemata, and form). This step involves feature-based commits of the DCML annotations, i.e., the high-level features. Fig. 2 shows examples of such annotations for Mozart's K. 279 (first movement, measure 16). The DCML harmony annotation standard allows us to account for a high degree of structural detail in the music. This

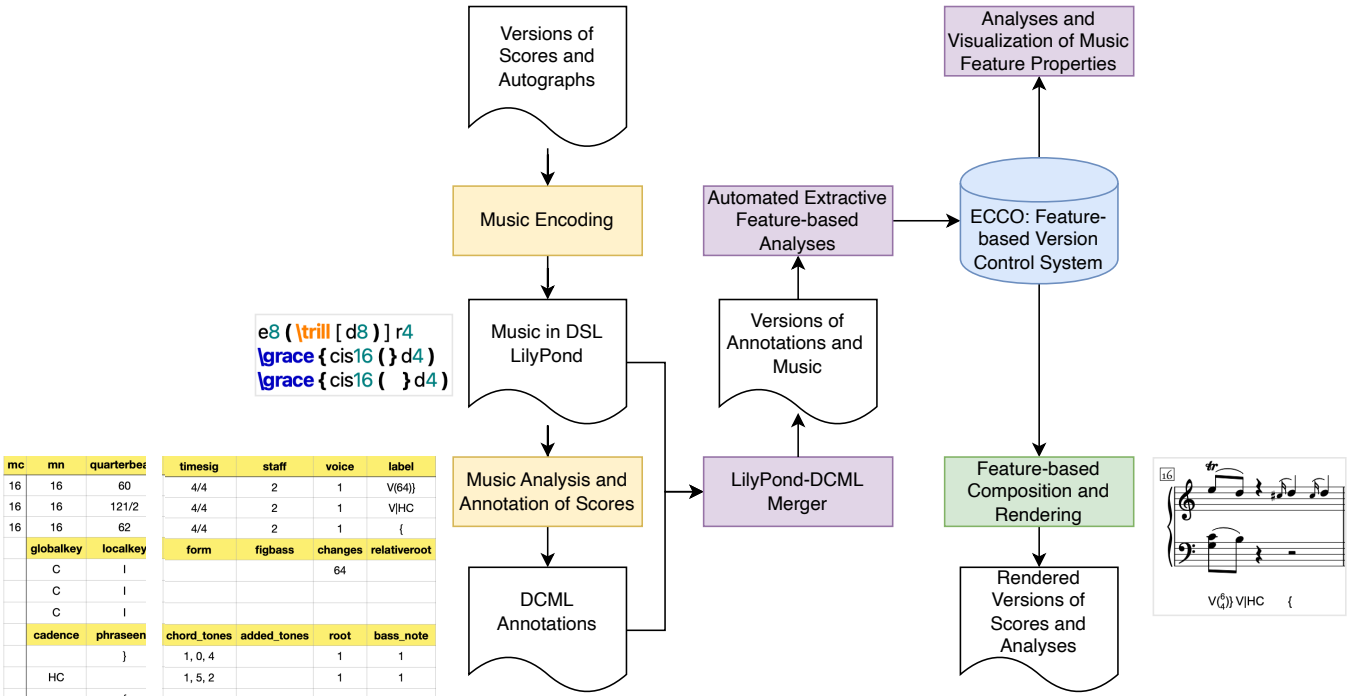

Figure 2: Feature-oriented approach for managing revisions and variants of DSLs for music notation and music analysis.

concerns both the level of key and the level of the individual chords. Keys occurring within a movement ("local keys") are referenced by means of Roman numerals in relation to the main key (e.g., I, ii, iii, IV, etc. as the chords used within the diatonic major mode). Temporary references to a key are expressed by "applied chords" (e.g., V/V as the dominant of the dominant). With regard to the level of chords, the standard allows one to encode the features of "chord type" (major, minor, diminished, augmented, etc.), "chord form" (such as a root-position chord or an inversion thereof), "suspensions" (such as 6/4 temporarily replacing the third and the fifth of a chord, as shown in Figure 2) and "added notes", as well as harmonic motion over a pedal. In addition, phrase endings can be indicated by using curly brackets, and cadence types (such as full or half cadences) appearing at phrase endings can be expressed by the pipe symbol and the appropriate abbreviation (such as PAC, IAC, HC, or DEC) after the phrase-final chord (for more details, see [18]). Our current prototype involves a Python script integrating music encoded in LilyPond with such DCML annotations. For instance, the annotated score in Fig. 1 was automatically rendered based on the output of this program.

Automated Feature-based Analyses. We analyze the commonalities and differences of the various versions of annotated scores. Specifically, we use the intensional variation control system ECCO providing feature-based mechanisms for automatically tracing the location of features in artifacts and storing them in a feature-based repository [25]. Specifically, ECCO determines the code snippets shared between multiple versions of the DSL code and creates mappings to features common in those versions. ECCO also determines code snippets of the DSL code only existing in some versions and creates mappings to features accordingly [24]. The resulting feature-to-code mappings allow managing fine-grained variants and thereby avoid feature branches or variant branches known from extensional versioning systems like Git. Furthermore, ECCO allows creating versions that have not been explicitly committed as such before. In our ongoing work we enhance ECCO to the domain of music analysis, based our experiences of extending ECCO to the DSL Lilypond [16].

Statistics and Visualization of Music Feature Properties. We will study feature properties by inspecting the feature-to-code mappings as well as highlighting differences in the versions with regard to different feature properties, such as different degrees of granularity and theoretical conceptions.

Feature-based Composition and Rendering. Variation control systems use features and composition rules to create arbitrary versions. This means that with ECCO a user can at any time check-out (combinations of) features stored in the repository to compose arbitrary variants of a music artifact, even if they were not submitted as such to the repository before. Depending on the history of the earlier committed features and their interactions, this can then be used to fully automatically compose new variants. Such sampling based on both intensional and extensional versioning [7] allows to create certain parts or excerpts of annotated scores as a pre-requisite for more sophisticated use cases for music analysis (e.g., performing different analyses of the same passages). However, manual effort may be needed to complete code for missing feature interactions. We will use ECCO to compose and visualize different versions of scores. This also includes rendering scores with attached music annotations. For instance, Fig. 2 shows the measure 16 of

Mozart's K. 279 (first movement) with some DCML annotations as rendered by our prototype.

We have identified Anton Bruckner's Third Symphony as a case study for the evaluation of our approach. This choice is justified by the fact that this symphony exists in no fewer than three major *versions* and features a complex harmonic, voice-leading, and formal structure in terms of intra-work relationships, which we shall describe in terms of *variants* and *patterns*. The first version from 1872-73 happens to be the longest symphonic work in Bruckner's entire œuvre), and changes in the second and third versions (from 1878 and 1889, respectively) primarily serve to reduce the work's excessive length. In addition, we shall consider the D-minor Symphony known as the "Annullierte," which in several ways prefigures the shape of the Third Symphony.

Overall, our goal to create a uniform approach for managing revisions and variants in music analysis research is challenged by the high number of musical revisions and variants results in many features, which increases computational complexity; the need to reconcile different views and perspectives when modeling variability both at the level of scores and the level of annotations as required by musicologists; the modularity and granularity of music features, which may render their analysis difficult; and the possibly still important role of the human in the loop, caused by the limitations in creating a fully automated approach for tracing music features in the DSL code.

5 CONCLUSIONS AND FUTURE WORK

Our research is interdisciplinary in nature, and depends on expertise from and interactive collaboration between the fields of musicology and software engineering. We expect that musicological research will benefit from feature-based versioning of music artifacts (both scores and analyses), as it allows for enhanced analyses by the advanced management of revisions and variants. Also, it will combine philology and analysis, which are usually carried out as separate activities. We expect that our results will be useful to the field of Digital Humanities, which is located at the intersection of computing and and different disciplines of the humanities. Software engineering research, on the other hand, can benefit from improving existing methods for version control by applying existing methods and techniques in a novel domain dealing with unusual feature characteristics. The research community in software engineering will benefit from results on variability engineering in the context of DSLs. Despite existing work (e.g., [17]), this field deserves attention given the growing importance of DSLs in many areas. Further, a systematic literature study on DSLs further pointed out a lack of evaluation research [21], which we intend to pursue in future work.

REFERENCES

[1] Sven Apel, Don Batory, Christian Kstner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines: Concepts and Implementation.* Springer-Verlag, Berlin, Heidelberg.

[2] Sven Apel, Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, and Brady Garvin. 2013. Exploring Feature Interactions in the Wild: The New Feature-Interaction Challenge. In *Proceedings 5th International Workshop on Feature-Oriented Software Development* (Indianapolis, Indiana, USA) *((FOSD '13))*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2528265.2528267

[3] Ian D. Bent, David W. Hughes, Robert C. Provine, Richard Rastall, Anne Kilmer, David Hiley, Janka Szendrei, Thomas B. Payne, Margaret Bent, and Geoffrey Chew. 2001. Notation. https://doi.org/10.1093/gmo/9781561592630.article.20114

[4] Thorsten Berger, Daniela Lettner, Julia Rubin, Paul Grünbacher, Adeline Silva, Martin Becker, Marsha Chechik, and Krzysztof Czarnecki. 2015. What is a Feature? A Qualitative Study of Features in Industrial Software Product Lines. In *Proc. 19th International Software Product Line Conference* (Nashville, USA). ACM, New York, NY, USA, 16–25. https://doi.org/10.1145/2791060.2791108

[5] Andrew Brinkman, Daniel Shanahan, and Craig Sapp. 2016. Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of Josquin. In *Proceedings of the Biennial International Conference on Music Perception and Cognition*. San Francisco, USA, 91–97.

[6] Thomas Christensen (Ed.). 2006. *The Cambridge History of Western Music Theory*. Cambridge University Press, Cambridge.

[7] Reidar Conradi and Bernhard Westfechtel. 1998. Version Models for Software Configuration Management. *ACM Computing Surveys* 30, 2 (1998), 232–282. https://doi.org/10.1145/280277.280280

[8] Susanne Cox and Johannes Kepper. 2021. Encoding Genetic Processes II. In *Music Encoding Conference 2021*. Humanities Commons, Alicante, Spain, 85–95. https://doi.org/10.17613/q6y4-9139

[9] Krysztof Czarnecki and Ulrich Eisenecker. 2000. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Boston, MA.

[10] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool features and tough decisions: a comparison of variability modeling approaches. In *Proceedings 6th International Workshop on Variability Modelling of Software-Intensive Systems*. ACM, New York, NY, USA, 173–182.

[11] Robert O. Gjerdingen. 2007. *Music in the Galant Style*. Oxford University Press, New York.

[12] Mark Gotham and Matthew Ireland. 2019. Taking form: A representation standard, conversion code, and example corpus for recording, visualizing, and studying analyses of musical form. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. International Society for Music Information Retrieval, Delft, The Netherlands, 693–699.

[13] Elaine Gould. 2011. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, London, United Kingdom.

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

[15] Paul Grünbacher. 2022. A Study on Variability for Multi-Device Rendering in Digital Music Publishing. In *Proceedings 16th International Working Conference on Variability Modelling of Software-Intensive Systems* (Florence, Italy). Association for Computing Machinery, New York, NY, USA, 6:1–6:9. https://doi.org/10.1145/3510466.3510482

[16] Paul Grünbacher, Rudolf Hanl, and Lukas Linsbauer. 2021. Using Music Features for Managing Revisions and Variants in Music Notation Software. In *Proceedings International Conference on Technologies for Music Notation and Representation* (Hamburg, Germany), Rama Gottfried, Georg Hajdu, Jacob Sello, Alessandro Anatrini, and John MacCallum (Eds.). Hamburg University for Music and Theater, Hamburg, Germany, 212–220.

[17] Øystein Haugen, Birger Møller-Pedersen, Jon Oldevik, Gøran K. Olsen, and Andreas Svendsen. 2008. Adding Standardized Variability to Domain Specific Languages. In *Proceedings 12th International Software Product Line Conference*. IEEE Computer Society, Washington, DC, USA, 139–148. https://doi.org/10.1109/SPLC.2008.25

[18] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. 2021. The Annotated Mozart Sonatas: Score, Harmony, and Cadence. *Transactions of the International Society for Music Information Retrieval* 4, 1 (2021), 67–80. https://doi.org/10.5334/tismir.63

[19] Daniel Hinterreiter, Lukas Linsbauer, Kevin Feichtinger, Herbert Prähofer, and Paul Grünbacher. 2020. Supporting feature-oriented evolution in industrial automation product lines. *Concurrent Engineering* 28, 4 (2020), 265–279. https://doi.org/10.1177/1063293X20958930

[20] Christian Kästner, Sven Apel, and Martin Kuhlemann. 2008. Granularity in software product lines. In *30th International Conference on Software Engineering, Leipzig, Germany, May 10-18, 2008 (ICSE)*, Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn (Eds.). Association for Computing Machinery, New York, NY, USA, 311–320. https://doi.org/10.1145/1368088.1368131

[21] Tomaž Kosar, Sudev Bohra, and Marjan Mernik. 2016. Domain-Specific Languages: A Systematic Mapping Study. *Information and Software Technology* 71 (2016), 77–91. https://doi.org/10.1016/j.infsof.2015.11.001

[22] David Lewis, Elisabete Shibata, Mark Saccomano, Lisa Rosendahl, Johannes Kepper, Andrew Hankinson, Christine Siegert, and Kevin Page. 2022. A Model for Annotating Musical Versions and Arrangements across Multiple Documents and Media. In *Proceedings of the 9th International Conference on Digital Libraries for Musicology* (Prague, Czech Republic) *(DLfM '22)*. Association for Computing Machinery, New York, NY, USA, 10–18. https://doi.org/10.1145/3543882.3543891

[23] LilyPond. 2022. LilyPond – Notation Reference. https://lilypond.org/doc/v2.22/Documentation/notation.pdf

[24] Lukas Linsbauer, Stefan Fischer, Gabriela Karoline Michelon, Wesley K. G. Assunção, Paul Grünbacher, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2022. Systematic Software Reuse with Automated Extraction and Composition for

Clone-and-Own. In *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*, Roberto Erick Lopez-Herrejon, Jabier Martinez, Tewfik Ziadi, Mathieu Acher, Wesley K. G. Assunção, and Silvia Regina Vergilio (Eds.). Springer, Cham, 379–404.

[25] Lukas Linsbauer, Felix Schwägerl, Thorsten Berger, and Paul Grünbacher. 2021. Concepts of Variation Control Systems. *Journal of Systems and Software* 171 (2021), 110796. https://doi.org/10.1016/j.jss.2020.110796

[26] Marjan Mernik, Jan Heering, and Anthony M. Sloane. 2005. When and How to Develop Domain-Specific Languages. *Comput. Surveys* 37, 4 (2005), 316—344. https://doi.org/10.1145/1118890.1118892

[27] Fabian C. Moss, Markus Neuwirth, Daniel Harasim, and Martin Rohrmeier. 2019. Statistical Characteristics of Tonal Harmony: A Corpus Study of Beethoven's String Quartets. *PLoS One* 14, 6 (2019), e0217242.

[28] Markus Neuwirth, Christoph Finkensiep, and Martin Rohrmeier. 2023. Musical Schemata: Modelling Challenges and Pattern Finding (BachBeatles). In *Mixing Methods. Practical Insights from the Humanities in the Digital Age*. transcript, Bielefeld, 147–164.

[29] Markus Neuwirth, Daniel Harasim, Fabian C. Moss, and Martin Rohrmeier. 2018. The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets. *Frontiers in Digital Humanities* 5, 16 (2018), 5 pages. https://doi.org/10.3389/fdigh.2018.00016

[30] Friedemann Sallis. 2015. *Music Sketches*. Cambridge University Press, Cambridge.

[31] Agnes Seipelt, Paul Gulewycz, and Robert Klugseder. 2018. Digitale Musikanalyse mit den Techniken der Music Encoding Initiative (MEI) am Beispiel von Kompositionsstudien Anton Bruckners. *Die Musikforschung* 71, 4 (2018), 366–378.

[32] Axel Teich Geertinger. 2021. Digital Encoding of Music Notation with MEI. In *Notated Music in the Digital Sphere. Possibilities and Limitations*, Margrethe Støkken Bue and Annika Rockenberger (Eds.). Nota bene – Studies from the National Library of Norway, Vol. 15. National Library of Norway, Oslo, 35–56.

[33] Dmitri Tymoczko, Mark Gotham, Michael Scott Cuthbert, and Christopher Ariza. 2019. The Romantext Format: A flexible and standard method for representing Roman numeral analyses. In *International Society for Music Information Retrieval Conference*. ISMIR, Delft, The Netherlands, 123–129.

[34] Arie van Deursen et al. 2000. Domain-Specific Languages: An Annotated Bibliography. *ACM SIGPLAN Notices* 35, 6 (2000), 26–36. https://doi.org/10.1145/352029.352035

[35] Pamela Zave. 1993. Feature interactions and formal specifications in telecommunications. *Computer* 26, 8 (Aug 1993), 20–28. https://doi.org/10.1109/2.223539