ODD Structures and Where to Find Them

Benjamin W. Bohl
Johann Wolfgang Goethe-Universität Frankfurt, Germany
bohl@em.uni-frankfurt.de

Johannes Kepper
Paderborn University, Germany
kepper@edirom.de

Abstract

In the past twenty years, the technical setup of the Music Encoding Initiative (MEI) data framework has been adjusted several times. Each of those transitions was motivated by the wish to improve the ways in which MEI could be integrated with other formats, to simplify the maintenance of MEI, and to encourage more people to actively contribute to the development of MEI. Some of those objectives are contradictory, and accordingly, there is no single right answer for all times about the best possible technical setup for MEI.

The main purpose of this poster is to give a historical overview of the technical setups that MEI has gone through in the 20 or so years of its existence, and to illustrate the current workflows. Ideally, this empowers wider parts of the community to contribute to the continued development of both the MEI *specification* and *documentation*. Eventually, it will explain the steps necessary to set up a local working environment to participate in these developments.[1]

Historical Background

The first versions of MEI were specified using Document Type Definition (.dtd) files, starting around 2001. While not being officially published back then, the first version of MEI to be made available as a set of XSD (XML Schema Definition) files was version 1.7b, released in September 2006.[2] This version introduced the possibility to use an XML namespace for MEI, turning it into a much more serious option for storing scholarly research data. While many structures available in current MEI sources were already introduced back then,[3] the lack of easily accessible documentation was surely a major obstacle for the wider dissemination of the format at that time. Actually, there was some very good documentation, but it was embedded into the MEI sources as XML comments, and it mixed user documentation with explanations of the underlying technical structures.

As a wider community started to adopt MEI, that community published a first official release of the format in May 2010, which is still available on GitHub.[4] This was the first version of MEI expressed as a set of RelaxNG (RNG) files. More importantly, the release included a PDF file called the “MEI Tag Library”, which provided proper end-user documentation about the elements available.

The next important change to the technical organization of MEI was introduced with MEI v2.0.0, released in August 2012. This was the first version to adopt TEI’s ODD language as the underlying technical framework.[5] The eponymous one document does it all (ODD) system is based on the idea of writing the technical format specifications and the corresponding documentation in the same place, and generating both schema and docs from there. The intention is to help avoid discrepancies between the two, and to ensure full coverage of the documentation. For MEI, this transition allowed the use of tools and workflows developed by the TEI community and simplified the combination of both TEI and MEI data in a mixed environment. Another benefit is that ODD has a built-in mechanism for customizations and extensions of the standard, which are again

1 This poster will assume a certain level of familiarity with technical concepts both from the XML world and around Git(Hub). We decided to not bloat this poster with (references to) explanations of these terms, and not give links to authoritative introductions. In our experience, there is no single ‘best’ explanation, but depending on the individual learning curve, different materials will be considered as appropriate starting points. However, all these terms and technologies are very common, so that Wikipedia or a simple search on Google will bring up tutorials and introductions easily. In order to follow this poster, it is not required to have expert knowledge in those technologies.

2 The DTD files for MEI 1.7b can be retrieved from https://web.archive.org/web/20110131100045/http://www2.lib.virginia.edu/innovation/mei/DTD/ (accessed January 12, 2022).

3 For instance, the organization of attributes in classes organized by musical domains still in use today was already implemented in those versions of MEI, utilizing quite different technologies than today.

4 https://github.com/music-encoding/music-encoding/releases/tag/MEI_release_2010-05 (accessed January 12, 2022).

5 ODD was introduced as a meta-schema-language by the Text Encoding Initiative during the 1990’s. See [1] .

self-documenting. The MEI documentation, now for the first time called the "MEI Guidelines", was significantly extended by more than 250 pages of prose description of how to use MEI, and also for the first time included documentation for attribute classes, model classes, and other technical structures used to specify MEI.

However, the perceived complexity of ODD led to very few contributions to the MEI sources from other than a small group of community members that were actively involved in the transition to ODD in the first place. Hoping to lower technical barriers and thus increasing community involvement in the maintenance of the MEI Guidelines, it was decided in 2017 to once again separate the documentation from the MEI specification. While the *specs* continued to be expressed in MEI, the Guidelines were converted to Markdown and published through GitHub Pages, making it relatively easy to contribute directly online with automatic pull requests.

In the years since that last transition, however, it became clear that this Markdown approach did not increase participation to the extent expected. At the same time, it became evident that the separation of *specs* and *docs* also led to significant friction loss when trying to keep both aligned. As both parts were stored in different repositories, pull requests had to be made independently against two separate repositories, which led to inconsistencies more often than necessary and also facilitated the development of the *specs* without keeping the *docs* in line. Accordingly, in a great community effort over the past year, the *specs* and *docs* of MEI have been reintegrated into a single codebase, which is now again compliant with the concepts of ODD. However, this return to ODD will also inevitably bring back the former challenges. Accordingly, the community is currently seeking to address this situation in multiple ways. First, better documentation for the ODD setup itself is provided. This includes online tutorials on how to use ODD in the context of MEI,[6] but also workshop events that offer in-person training on ODD.[7] There are also monthly so-called *ODD meetings*.[8] Parallel to this, the community offers tools like the MEI Profile Drafter,[9] which facilitates the generation of project-specific ODD customizations, or the ODD-API Web Service,[10] which allows software developers to query ODD structures programmatically. We anticipate that over time these efforts will increase ODD literacy within the MEI community, lead to a better use of the available concepts, and ultimately help to improve project-specific documentation.

Repository Structure

Currently, the MEI source code is maintained in a single repository on GitHub.[11] The main assets are located in the two folders `/customizations` and `/source` within that repository. The first folder holds ODD customizations for the official MEI profiles *mei-all*, *mei-CMN*, *mei-Mensural*, *mei-Neumes*, *mei-all_anyStart*, and *mei-basic*. Those profiles are pre-built versions of MEI, tailored to specific common use cases. The second folder holds the MEI source code. The main entry point there is the `mei-source.xml` file, which serves as an overarching container for all other files containing the MEI *docs* and *specs*. It uses XInclude[12] pointers to pull in the contents of these additional files. For an XML parser, this linking is fully transparent – it looks as if the content of those other files would occupy the position of the XInclude statements. This allows the *specs* to be split up into separate files according to MEI modules, while the *docs* are split up to match the first-level chapters. The main motivation for this separation is an improved maintainability of MEI. The resulting files for each component seem to offer a good compromise between having sufficient context and a manageable size.

Formerly, the RelaxNG schemata compiled from the MEI sources were also stored in the `/music-encoding/music-encoding` source repository. Today, this is organized in a different way. On a commit to the source repository, both *docs* and *specs* are compiled automatically through a so-called continuous integration (CI) workflow running on GitHub Actions. The resulting Guidelines files are then automatically pushed to the separate `/music-encoding/guidelines` repository[13] and served to the MEI website from there. The gen-

6 See https://music-encoding.org/resources/tutorials.html (accessed January 12, 2022).

7 There was a workshop on ODD during Music Encoding Conference 2021. Other opportunities include the annual MEI developer workshop in fall or the annual Edirom Summer School in Paderborn (https://ess.upb.de; accessed January 12, 2022).

8 Happening every last Friday of odd months and every last Thursday of even months via Zoom.

9 https://meigarage.edirom.de/profiler (accessed January 12, 2022).

10 See https://odd-api.edirom.de/ and https://github.com/Edirom/odd-api/ (both accessed January 12, 2022).

11 https://github.com/music-encoding/music-encoding (accessed January 12, 2022).

12 See https://www.w3.org/TR/xinclude/ (accessed February 14, 2022)

13 https://github.com/music-encoding/guidelines (accessed January 12, 2022).

erated schemata are automatically transferred to the `/music-encoding/schema` repository.[14] With this new workflow, both repositories are primarily considered read-only access relays of the compiled build artifacts. Although technically edits are, of course, possible, they are highly discouraged and reserved for bug fixing purposes, e.g., of older MEI versions.

Contributing

With this new source code structure, the 'official' procedure to contribute to MEI is roughly as follows:

- An MEI Interest Group, an individual member of the MEI community,[15] or just anyone willing to contribute to MEI prepares an idea on how to improve MEI.
- After discussing it, this idea is fleshed out as proper ODD customization. If necessary, members of the Technical Team of MEI will assist with this task.
- The customization is expected to not only change the MEI specifications, but also to cover the changes in the Guidelines.
- A Pull Request (PR) against the `/music-encoding/music-encoding` repository with the proposed changes is made.
- Technical aspects and interdependencies with other parts of MEI are discussed openly on GitHub.
- As soon as the discussion seems to be resolved, the PR will be added to the agenda of the next upcoming ODD meeting. Ideally, someone involved in the preparation of the PR should join that meeting in order to accomodate potential questions.
- During the ODD meeting, the community members present will have a final check on the PR and see if there are no major objections concerning the proposed changes. If approved and all questions could be resolved, the PR will be accepted and merged into the development version of MEI. If there are still questions, the topic is tabled to the next ODD meeting.

For all of these steps, support is always available, and feedback and questions are strongly encouraged at any time.

Besides this official workflow, it is of course possible to compile customizations locally for testing purposes, etc. This can be done using the MEI Customization Service,[16] but also in a local working copy of the `/music-encoding/music-encoding` repository. The latter requires the following steps on the command line:

1. If you do not have a local copy (clone) on your local machine yet, run the following from your command line:
   ```
   git clone https://github.com/music-encoding/music-encoding.git --recursive
   ```

2. If you already have a clone on your system, you still might have to initialize the submodules by running the following commands from the command line:
   a. Switch to your clone's directory:
      ```
      cd [YOUR-CLONE-LOCATION]
      ```

   b. Initialize the submodules:
      ```
      git submodules init
      ```

   c. Updatre the submodules:
      ```
      git submodules update
      ```

14 https://github.com/music-encoding/schema (accessed January 12, 2022).

15 According to the MEI By-laws, membership in the community is bound to the subscription of the MEI Mailing list, which is open to anyone interested in MEI (see https://music-encoding.org/community/mei-by-laws.html#3-community-membership; accessed January 12, 2022).

16 See https://meigarage.edirom.de/customization (accessed January 12, 2022).

3. Check if your system meets the build requisites:[17]
   a. Is Java 8 or above available on your machine?
      `java -version`
      This should return something similar to:
      ```
      openjdk version "11.0.9" 2020-10-20
      OpenJDK Runtime Environment (build 11.0.9+11)
      OpenJDK 64-Bit Server VM (build 11.0.9+11, mixed mode)
      ```

   b. Is Apache Ant installed?
      Apache Ant[18] is a library for building software projects and drives the creation of MEI schemata and guidelines from the ODD sources. Run the following command to see if it is available on your system:
      `ant -version`

      This should return something similar to:
      ```
      Apache Ant(TM) version 1.10.9 compiled on September 27 2020
      ```

4. Initialize the build process:
   a. Switch to your clone's directory:
      `cd [YOUR-CLONE-LOCATION]`

   b. Call the Apache Ant init task:
      `ant init`

5. Run the build process, either for specific parts of MEI (a., b.) or all possible build artifacts (c.):

   a. Build guidelines HTML:
      `ant -lib lib/saxon/saxon-he-10.5.jar build-guidelines-html`

   b. Build a specific customization's RNG schema:
      ```
      ant -lib lib/saxon/saxon-he-10.5.jar
      -Dcustomization.path=[/PATH/TO/YOUR/CUSTOMIZATION/FILE.xml] build-rng
      ```

   c. Build everything (all customizations shipped with this repository, compiled ODDs for each customization, guidelines HTML):
      `ant -lib lib/saxon/saxon-he-10.5.jar`

Acknowledgments

This poster is based on the work of many active members of the MEI community, specifically those who attended the MEI Developer Workshop in November 2020. Without the contributions of this group, neither this poster nor the described changes to the technical setup of MEI would have been possible.

Works Cited

[1] Sperberg-McQueen, C. Michael, and Lou Burnard. "One Document Does It All: Documentation for an ODD System for Tag Set Construction" TEI Working Paper ED W29, December 1991, https://www.tei-c.org/Vault/ED/edw29.tar (accessed January 12, 2022).

17 If any of the below commands return an empty string, please update or install Java or Apache Ant according to an installation instruction matching your operating system (to be found on the internet)

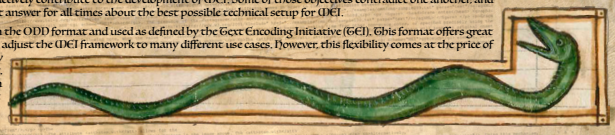
18 See https://ant.apache.org/manual/install.html (accessed January 12, 2022).

ODD STRUCTURES AND WHERE TO FIND THEM

Contributing to the MEI Framework

In the past twenty years, the technical setup of the Music Encoding Initiative data framework has been adjusted several times. Each of those transitions was motivated by the wish to improve the ways in which MEI could be integrated with other formats, to simplify the maintenance of MEI, and to encourage more people to actively contribute to the development of MEI. Some of those objectives contradict one another, and accordingly, there is no single right answer for all times about the best possible technical setup for MEI.

Since 2020, MEI is based (again) on the ODD format and used as defined by the Text Encoding Initiative (TEI). This format offers great flexibility, and makes it possible to adjust the MEI framework to many different use cases. However, this flexibility comes at the price of a certain complexity, which may seem challenging at first sight. This poster intends to give an overview of where to find different types of information, and how to contribute to MEI.



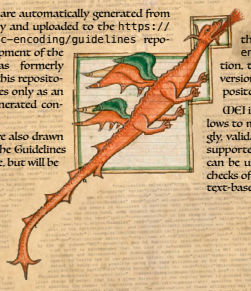
Main Repository

The main repository for MEI can be found at `https://github.com/music-encoding/music-encoding`. This is where all the sources for MEI can be found. Interaction with the MEI framework should happen through this repository: Features can be requested here, issues can be brought up for clarification and technical discussion, and changes can be proposed. While opening Pull Requests here is certainly encouraged, it is not required if you're not familiar with such procedures – just bring up your topic in any convenient form, and eventually, it will be moved to this repository by someone from the technical community. Opening discussions on the MEI mailing list or asking questions on Slack is certainly fine. You will find contact details at `https://music-encoding.org/community/community-contacts.html`.

Guidelines

The MEI Guidelines are automatically generated from the main repository and uploaded to the `https://github.com/music-encoding/guidelines` repository. While development of the Guidelines was formerly conducted in this repository, now it serves only as an archive for generated content, and one must not edit the files there.

The Guidelines embedded into the MEI website are also drawn from this repository. Since 2021, a PDF version of the Guidelines is made available again. It is available on the website, but will be archived in this repository as well.



Schema

The MEI Schema files for official releases are automatically uploaded to the `https://github.com/music-encoding/schema` repository. In addition, the current state of the development version of MEI can also be found in this repository.

MEI is built with the concept of XML Namespaces in mind, which allows to mix it with other namespaced XML formats like TEI. Accordingly, validation based on Document Type Definition files (DTDs) is not supported. Instead, MEI offers schemata in the RelaxNG format, which can be used to validate MEI instances. In addition to the structural checks of RelaxNG, MEI utilizes a large set of Schematron rules for context-based validation that would not be possible without Schematron.

Procedures for Changes

Once per month, on the last Friday in odd months, and on the last Thursday in even months, everyone is invited to join the so-called ODD Fridays. During these meetings, open issues and pull requests for MEI are reviewed in public. When the discussions on GitHub and during these meetings have reached consensus, the proposal will be merged into the development version of MEI, on which the next release will be based. There is no fixed release schedule. Instead, interest groups and individuals are invited to discuss upcoming releases during ODD Meetings. These meetings will be announced on MEI-L and on a dedicated Slack channel. No particular knowledge is expected, and everyone interested is invited to join.

Lancelot-Graal, 4, Folio 524r, Source: gallica.bnf.fr / Bibliothèque nationale de France
Nautical Chart, 1500, Mateo Prunes, Source: gallica.bnf.fr / Bibliothèque nationale de France
The British Library,London, Harley Ms. 3244, fol. 55, Image: Grangerbourne
Book of Flowers, 1608, Koninklijke Bibliotheek, National Library of the Netherlands, Ms.72 A 23, fol. 46
Graphic:@GarryKillian / pch.vector, Freepik.com

Benjamin W. Bohl | bohl@em.uni-frankfurt.de
Johannes Kepper | kepper@edirom.de