Stephen Bresnick and Eyal Shahar

MAS 714- Technologies for Creative Learning

Final Paper

December 12, 2011

CODA

What is Coda?

Coda is a web-based musical software interface that facilitates the identification and sharing of knowledge related to music theory. As a music analysis tool, Coda automatically searches through written music to look for individual musical attributes and identifiable patterns about which the user can add an article or a comment. As more and more users provide articles, the environment evolves from a high-level social discussion about music theory to an intelligent music tutor. The community provides metadata and new pathways to improve the intelligence of the Coda language and provide more detailed descriptions and opportunities for building knowledge about music theory. Likely users of Coda include composers, arrangers, music students, instrumentalists, music innovators, and anybody else interested in participating in an intelligent and ongoing discussion around music theory.

How Coda Works:

Coda is built on top of the Noteflight website database and API. Noteflight is an online music writing application designed to allow users to create, view and share music scores. Within the Coda site, a user can select any score from the Noteflight and view it. At this point, the user can click on a note, chord, staff, series of notes, scale, harmonic or melodic interval, or any other attribute of the musical composition that can be parsed out by Coda, and write a comment. User comments are open-ended, and can consist of any level of sophistication, from simple assertions of liking or disliking the way a composer has employed the music theory, to complex articles delineating the topic of music theory being employed, ways that the topic can be implemented in other areas of music, appropriate uses of a certain concept or topic, or the like. The users will be able to input metadata in the form of tags, labels, and links to other compositions where this topic of music theory is being employed. The greater the number and quality of user inputs, the more intelligent and robust the software becomes, and the more valuable a tool this becomes for music composers, students, and enthusiasts. In addition, users are able to add algorithms that extract musical information to the system, and these produce new valid entries for articles.

Python, Mingus, and Noteflight

Coda is a web-based interface that functions inside the user's browser. The user interactions are gathered within an HTML and JavaScript environment, and kicked over to a server that is running a Python-based library known as Mingus. Mingus interprets the user data, then generates a new HTML page based on the specifications of the user interaction and Mingus' interpretations. Now, the user is able to create an article and share their knowledge with the community.

At the core of Coda's functionality is Mingus, a Python-based module named after legendary jazz bassist and composer Charles Mingus, which interprets and generates music from math-based algorithms and text. Mingus is written in Python and runs in the Python environment, a simple yet powerful open-source computer programming language with simple, straight-forward syntax and a large online support community. Mingus contains algorithms designed to interpret various musical patterns such as tones, note values, diatonic scales, chord intervals and inversions, staves and key signatures. In addition to all of these built-in functions, it is possible to extend Mingus to recognize more complex patterns such as relative minor keys, custom chord changes, modes, or complex rhythmic patterns. Mingus uses math to build the foundations of music, and extending its functionalities is simply a question of defining new patterns and functions.

Although we would like to have allowed the user to compose music directly within the Coda user environment, this was not within our immediate capability for this project. We found a highly effective solution, however, in the form of Noteflight, a Flash-based music composition tool that allows users to create, playback, and share original musical compositions. Once a user has created a musical composition in Noteflight, the score data is available for Coda through use of the Noteflight API. This allows Coda to retrieve the score data in a XML data structure called a *score*. A *score* is an array of *staves*, which in turn are arrays of *measures*. Each *measure* is an array of *notesets*, each representing a certain event at a particular time in the measure and having a particular duration. A *noteset* can consist of a rest, a note or several notes. In summary, the Noteflight API takes Noteflight musical notation and translates the ornate, illustrative qualities of music into simple text, which is now readable by Coda.

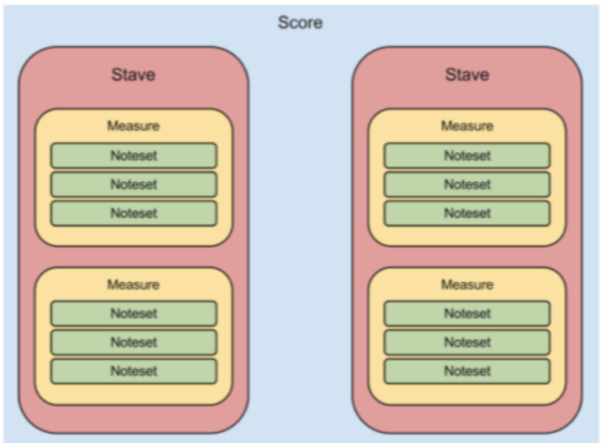

Figure 1: The Data Structure of Noteflight's API

Coda, with the help of Mingus, is now able to parse out the various musical attributes in a Noteflight-created musical composition, allowing the user on the front end to select individual musical events: individual notes, scale runs, chords, rests, measures, clefs, harmonic intervals, melodic intervals, staves, and so forth. When the user clicks on these individual parts, Coda runs a set of music information extraction algorithms. These algorithms produce article entries such as “D minor”, “Quarter Note” or “Do Re So Fa”. Coda then searches its database for existing articles relating to these entries. A window pops up inviting the user to read existing articles or write an article for an empty entry, describing what they have clicked on. The user gives the article a title and writes up a brief comment identifying the note or symbol that has been clicked on, or a complex description of a topic of music theory that is being employed in this particular example. The user then submits the article to the community, and the next time a user clicks on a musical attribute that matches an article that another user has submitted, that user's article comes up providing information about the musical attribute that has been selected.

As Coda's knowledge base grows with each new article submitted, its ability to analyze musical compositions increases. For example, if a user clicks on a series of sequential notes that include E, F#, G, A, B, and C#, Mingus would recognize these notes as being in the key of D Major. However, Coda makes it possible to extend Mingus' capabilities and add a new category that applies to this note progression: E Dorian mode- the second degree of the D Major scale. Now that this article has been linked to this series of notes, future users of Coda will be able to see that the notes E, F#, G, A, B and C# are connected to articles about both "D Major" and "E Dorian Mode".

A version of Coda that is well-populated with community-contributed user data would become a powerful tool for musical analysis, as a newly imported composition would be instantaneously connected to previously written articles describing musical events that are applicable to that composition. Users would be able to link articles to other similar articles, and newly imported compositions to existing compositions that address similar ideas and practices. Coda would become a place for composers to go and receive automatically generated metadata and pattern analysis. Other users would then be able to submit subjective critiques or constructive criticism of the composition, providing the user with suggestions for how to improve or reinterpret their musical composition.

Use Case Scenario

Charles is a young teenager with a new growing interest in music. He has taught himself a few chords on his older sister's guitar, mostly by search through online Do-It-Yourself tutorials and YouTube Videos, but he realizes that he must gain more knowledge music theory in order to advance to the next level. His sister has composed several pieces and uploaded them to Noteflight. Charles enters the Coda website and explores one of his sister's pieces. He selects a note, and the system offers him two articles to read : "D#" and "Quarter Note". Charles begins exploring the concept of pitch and duration.

Charles knows that chords are several notes played together. He selects several notes and reads about some major and minor chords, but does not completely understands how to build a chord. He searches the Noteflight database for "Chords" and finds a score called "Chords Tutorial". Through looking at the score, listening to the examples and reading the Coda articles, he now understands the concept of chord construction.

Charles starts composing his own simple melodies in Noteflight. He plays by ear, and some chords he uses are neither major or minor. Through Coda, he discovers he had actually used a "Sus4 Chord". Coda also informs him that the entry for "Dotted Quarter Note" is empty. Charles makes his first contribution to the knowledge base by writing an article in which he explains that a dotted quarter note is the a note with a duration of one-and-a-half quarter notes.

As Charles explores a friend's piece, he finds an interesting chord progression. Coda tells him that this progression is called a "Plagal Cadence". Charles uses a *plagal* cadence in his own piece, but when he selects it Coda does not show "Plagal Cadence" in the list of entries. Charles has learned some computer programming, so he decides to build a custom algorithm called "Plagal Cadence Detector" and upload it to Coda. Charles now knows that whenever a community member selects a *plagal* cadence in any score, it will be his algorithm that detects it. Charles leaves the article itself empty - he wants to allow another member of the community to feel the sense of fulfilment that comes with contributing to the building of a robust and powerful music analysis tool.

The Community of Coda

Coda performs many functions, including a pattern analysis tool, a music wiki space, and a music theory tutor, but at the heart of Coda is its community of participation. The community members are the ones who provide metadata and new pathways to improve the intelligence of the program. Their input fuels the linear growth of the knowledge base by increasing the number of articles about various topics relating to music theory. And eventually, their contributions to the software allow for an amplification of the software's ability to interpret musical patterns.

In gathering feedback from our classmates in MAS714, we encountered several questions about the ability to maintain the low-floor barrier of entry in the context of a community that becomes increasingly sophisticated with each new article entered. Additionally, we were asked about an expert user's motivation for being part of this community: what would a musical expert personally gain from explaining musical theory to lesser-skilled musicians through online posts? What would motivate expert users to furnish our online community of participation with increasingly complex information about music?

In terms of maintaining a low-floor barrier of entry for novice users, we feel that our design can appeal to users of various skill levels. The initial barrier for entry is fairly low; potential novice members of the community must only come to the site with an appreciation of music, a knowledge of musical notation, an enthusiasm for music theory or a desire to learn more about music theory. Since the comment process is highly open-ended, novice users can enter text expressing adulation or criticism for a section of music, post a question they might have about a particular topic, or identify a basic musical attribute, such as a single note or a rest. Novice users would visit the site aiming to learn more about music theory for the purposes of improving their own composition skills, gaining a stronger understanding of the abstract musical concepts by seeing examples where the concepts are connected to concrete examples, or becoming better instrumentalists or site readers.

Intermediate users would be able to enter information about more advanced music theory topics, such as scales, modes, chord theory or chord progressions, and they would also be able to create links between various compositions that might contain the same topics of exploration. These users would be able to provide qualitative commentary on various uses of music theory topics, more sophisticated analysis of the usage of the music theory topic, and suggestions on how to better employ a certain music theory concept. Intermediate users would mostly consist of amateur composers with a strong knowledge of the basics of music theory who are looking to improve their crafts as composers, to gain new ideas and insights about their compositions, or to share their musical ideas with the rest of the community and elicit feedback from the other members.

Advanced users would bring to the community a strong understanding of music theory and advanced computer programming skills. These users would be well versed in the spirit of the GNU open-source software community and might even be active participants with a strong understanding of Python, Mingus, or JavaScript. These users would enjoy exploring ways that the Coda programming language could be extended, and would be responsible for creating new algorithms that would analyze musical patterns of increased complexity or improve the functionality of the program in some way. Advanced users' contributions would improve the intelligence of the Coda system exponentially instead of incrementally, as they would be helping to create a system that is more robust and more functional. They would create new pathways for users to explore, streamline the software's analytical features, or even invent new math-based musical theories that could be tested, displayed and analyzed within the confines of the Coda community.

In his article about communities of participation, Fischer outlines some of the most important features that compel users to contribute to online knowledge communities like Coda. The first important feature involves the design of the interaction, which must be flexible and expandable. He believes that the meta-design of a project must take into account "the assumption that future uses and problems cannot be completely anticipated at design time." Additionally, “making changes must seem possible,” as well as “technically feasible” (Fischer, p. 45). We feel that the design of our site certainly allows for changes to be “possible” and “feasible,” and the expandability of the software through inviting users to submit new algorithms makes our design flexible enough to take “future uses” of the interface into account.

Additionally, he states that "benefits must be perceived": the contributors to the site must believe that they are receiving some sort of benefit to justify their participation in the culture. These can include professional benefits, social benefits, or personal benefits. In general, we believe that people enjoy having their talents and areas of expertise put to good use. If they see themselves as educators for the "rest of us," they will receive intrinsic motivation for participating in the online community.

But a more compelling argument for expert participation comes in the user’s ability to utilize the power of computer technology to explore music theory, a subject that has evolved in the absence of technology and which might be expanded in exciting ways through the use of technology. Programming music theory into computer language promotes new ways of thinking about music theory. Perhaps, by merging the powerful mathematical processing capabilities of computers with music theory, we could have a community of users who might invent the next

Jazz. In our world of prefabricated pop music and the use of chord progressions that are hackneyed to the point of being cliché, it is valuable to have an ever-expanding resource, such as Coda, to which one can go for inspiration about new ideas and areas to explore within the realm of music composition. By allowing an interface where participants are able to contribute to the knowledge base by authoring informational articles and also to improve the usability, robustness and complexity of the program by contributing new algorithms to the Mingus environment, our site will hopefully become a one-stop shop for information, inspiration, construction of new ideas, and collaboration.

Connections to Course Readings

Coda is meant to serve as the center for a community of learners sharing theoretical knowledge, musical ideas and musical compositions. By decentralizing the knowledge and blurring the boundary between personal creation and general music theory, we strive to facilitate an environment where people are free to share musical ideas, to learn about increasingly complex concepts in order to upgrade their knowledge of music theory, and to share their insights with the rest of the world. Our project connects to several of the readings from the course. Mitch Resnick's article *All I Really Need To Know (About Creative Learning) I Learned (By Studying How Children Learn) In Kindergarten* informed our idea to create an environment for learning music theory incorporating the low floor/ high ceiling/ wide wall philosophy. We are hoping that this composition environment will help novice musicians to jump into the world of music theory, which can be a difficult and highly abstract concept to grasp. Through play and experimentation, novice composers will be able to practice and interact with the theories that other users supply in their comments.

Another course reading that is highly influential in our project design is Turkle and Papert's *Epistemological Pluralism.* Coda will help to make the abstract world of music theory more concrete by allowing users to interact directly with theoretical concepts in real time through musical composition. User comments will furnish a growing library of articles about abstract music theory concepts, and each article in this library will be linked with concrete examples of applications of each concept. Bricoleurs will be able to tinker with music to experiment with the effects of changing different attributes of the music, while learners who favor the "black box" approach will be able to incorporate tagged musical progressions into their own compositions based on whichever theory or pattern they wish to employ.

Finally, Fischer's *Understanding, Fostering and Supporting Cultures of Participation* is highly influential in our design of the sharing platform that will be incorporated into our software model. Fischer envisions a culture of participation where consumer-driven culture will shift to a producer-driven culture. Noteflight harnesses this idea of a producer-driven culture by providing a platform where users can compose music and easily play back their ideas to facilitate iterative revision. Not all music theory aficionados (potential members of this particular community of participation) see themselves primarily as composers, so it makes sense to allow musicians of all interest levels to participate in a variety of ways. Noteflight is a great for sharing by musicians who see themselves primarily as composers, and there are remixing capabilities for composers who are willing to allow other users to edit their compositions (similar to a shared Google Doc). However, there is limited opportunity for users to provide detailed descriptions of changes they might have made to another user's composition. Also, due to the granular nature of the feedback capabilities of Noteflight, it is very difficult for users to learn the more complex or minute details of music theory from one another. We believe that Coda will allow for more varieties of participation and contribution to take place than is currently possible on NoteFlight.

Future Work

Noteflight is a fairly new service and its API is far from being mature. Many key features of the score, including key signature, dynamic markings, and pedal denotations, are not included in the data objects retrieved by the API. Furthermore, when such features appear in the score, they produce inconsistent data objects, resulting in Coda being misinformed about the user's intended selection. Additionally, transforming Coda into a self-sufficient application requires adding features such as search, user log-in, and functionality enabling the creation of scores directly in the Coda website. Such capabilities currently require utilization of the advanced Noteflight API, called "Server API", which is quite cumbersome. Therefore, harnessing capabilities such as user log-in and user search requires tedious software development that is outside the scope of this work.

Future work on Coda will involve solving these issues, preferably in collaboration with Noteflight's developers, or substituting the use of Noteflight with a dedicated, custom-made notation interface to serve as an alternative to Noteflight, which is an entirely different project altogether. Finally, although the interface is supported by the software infrastructure, the functionality allowing users to upload their own algorithms has not yet been implemented, mainly due to time constraints. We stress, however, that this feature will transcend Coda into a vibrant community that teaches and learns music and programming in a process that is truly driven by creativity.

Works Cited:

Fischer, G. (2011). Understanding, fostering, and supporting Cultures of Participation. interactions, 18(3), 42-53.

Resnick, M. (2007). *All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten.* ACM Creativity & Cognition conference, Washington DC, June 2007.

Turkle, S., & Papert, S. (1990). *Epistemological Pluralism.* Signs, vol. 16, no. 1.