



8-2006

## **A Proposal for an Open Source System of Development and Research for Music CAI**

Daniel E. Clouse  
*University of Tennessee - Knoxville*

Follow this and additional works at: [https://trace.tennessee.edu/utk\\_gradthes](https://trace.tennessee.edu/utk_gradthes)



Part of the [Music Commons](#)

---

### **Recommended Citation**

Clouse, Daniel E., "A Proposal for an Open Source System of Development and Research for Music CAI. " Master's Thesis, University of Tennessee, 2006.  
[https://trace.tennessee.edu/utk\\_gradthes/1530](https://trace.tennessee.edu/utk_gradthes/1530)

This Thesis is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact [trace@utk.edu](mailto:trace@utk.edu).

To the Graduate Council:

I am submitting herewith a thesis written by Daniel E. Clouse entitled "A Proposal for an Open Source System of Development and Research for Music CAI." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Music, with a major in Music.

Barbara Murphy, Major Professor

We have read this thesis and recommend its acceptance:

Don Pederson, Gary Sousa

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Daniel E. Clouse entitled "A Proposal for an Open Source System of Development and Research for Music CAI." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Music, with a major in Music.

Barbara Murphy  
Major Professor

We have read this thesis  
and recommend its acceptance:

Don Pederson

Gary Sousa

Accepted for the Council:

Anne Mayhew  
Vice Chancellor and  
Dean of Graduate Studies

(Original signatures on file with official student records.)

A Proposal for an Open Source System  
of Development and Research for Music CAI

A Thesis

Presented for the

Master of Music

Degree

The University of Tennessee, Knoxville

Daniel Emerson Clouse

August 2006

## **Acknowledgments**

I wish to thank my many friends and family who supported me through my degree program at the University of Tennessee. I would like to thank Dr. Barbara Murphy for her patience, perspective, and guidance. I would also like to thank Dr. Les Gay for introducing me to many important concepts, much important literature, and challenging me at every turn to be a better scholar. I also offer thanks to Dr. Don Pederson and Dr. Gary Sousa for offering advice and support, and serving on my committee.

I thank my dearest friends Jamie Warren and TJ Ricer for numerous proof reading sessions and pep talks. TJ picked up the slack in my performing schedule and made it possible for me focus on being a theory major while Jamie kept me from taking myself too seriously to enjoy making music. The two of them kept me grounded, focused, and smiling over all the months.

Finally, I would like to thank my family for their support and praise. Their positive feedback and unending supply of love (and free food) made my journey possible.

## Abstract

The purpose of this thesis is to examine the historical use of music Computer Assisted Instruction (CAI) software to show that research on music CAI has decreased and to propose using a new method of coding and distribution (open source) that might increase research opportunities using music CAI. The reduction in research is due in part to limitations in existing software, as well as the practices of the music community. An open source CAI program called *Mobius* is described as an example of how open source programming can offer new opportunities for music researchers.

CAI software has played a prominent role in the college music school, and has a long history of research and innovation. Early CAI was used in numerous studies to show how effective computers could be at delivering instruction, while reducing the teacher workload at the same time. As computers became more widely adopted, CAI became more commonplace in the music school, and many CAI software programmers sold their programs to fill the growing demand. Modern CAI is now viewed more as a commercial product, and less as a research tool.

CAI can still be used as a powerful research tool. This thesis recommends using open source software development for music CAI since it allows programmers to share the workload of developing software, and allows CAI researchers to use existing open source as the basis for their new research programs. Included in this thesis are storyboards for several key components of an open source CAI program on music fundamentals, including an administrative portion, the actual CAI program, and a custom report builder.

## Table of Contents

<b>CHAPTER 1: INTRODUCTION .....</b>	<b>1</b>
<b>CHAPTER 2: LITERATURE REVIEW .....</b>	<b>4</b>
INTRODUCTION.....	4
CAI IN RESEARCH: MAINFRAME AND MICRO SYSTEMS .....	5
THE PC AND NEW TECHNOLOGY .....	13
CURRENT MUSIC CAI SOFTWARE.....	18
THE INTERNET AND NEW STANDARDS.....	19
<b>CHAPTER 3: CAI AND SOCIETY .....</b>	<b>23</b>
INTRODUCTION.....	23
THE CULTURE OF CAI DEVELOPMENT .....	25
SOFTWARE AS A PRODUCT .....	29
OPEN SOURCE SOFTWARE.....	32
<b>CHAPTER 4: PROPOSAL FOR A OPEN SOURCE CAI SYSTEM.....</b>	<b>36</b>
INTRODUCTION.....	36
MOBIUS SOFTWARE GOALS .....	36
MOBIUS: THE BASE SYSTEM.....	41
THE SOFTWARE TEAM.....	63
USING MOBIUS .....	65
<b>CHAPTER 5: CONCLUSION.....</b>	<b>68</b>

<b>BIBLIOGRAPHY</b> .....	<b>70</b>
<b>APPENDICES</b> .....	<b>74</b>
APPENDIX A: DISSERTATION ABSTRACTS DATA.....	75
<b>VITA</b> .....	<b>88</b>



## List of Tables

Table A.1: Number of Dissertations per Year .....	75
---	----

## List of Figures

Figure 1: CAI research dissertations per year .....	17
Figure 2: N-Tier Software Design.....	39
Figure 3: Class listing (instructor view) .....	43
Figure 4: Class controls (instructor view).....	44
Figure 5: Student listing by class .....	45
Figure 6: Add new student dialog .....	46
Figure 7: Existing homework and add new homework dialog boxes .....	48
Figure 8: Configure a homework assignment dialog .....	49
Figure 9: List of existing reports (Instructor View) .....	51
Figure 10: Choose data set for reports.....	52
Figure 11: Select specific data sets for a report .....	53
Figure 12: Choose the exercises reported.....	54
Figure 13: Select columns for the report .....	56
Figure 14: Organize and aggregate data .....	57
Figure 15: Report results.....	58
Figure 16: List of saved reports.....	59
Figure 17: List of modules installed .....	61
Figure 18: List of modules available for download from the central repository .....	62

## Chapter 1: Introduction

In the past 40 years computers and become permanent fixtures in music education. Computers are used for email, writing papers, and transmitting course materials. Programs and online courseware like *BlackBoard* make it easy for non-programmers to incorporate computers into their instruction. Some music Computer Aided Instruction (CAI) software claims to improve skills, impart knowledge, or just make the learning process more fun. Some programs promise students faster or more consistent results, while others offer a pre-packaged curriculum to lighten the workload of music teachers. CAI promises to give music students personalized lessons and plenty of practice on aspects of theory or other subjects that need extra drill, freeing the music teacher to answer questions the computer cannot. Teachers can focus on helping troubled students, creating innovative teaching methods, or researching pedagogical problems. CAI can provide an ideal mix of private practice for the student and reduce the workload for the teacher so he or she can focus on other concerns.

Research was one of the primary goals of early CAI. Researchers understood that by using computers, they could accurately capture complex data that would be used to build a body of knowledge on perception and effective teaching methods. They incorporated this knowledge into the next generation of CAI, making it better. The hope of the teachers was that the computer could take over most, if not all, of the repetitive tasks of music fundamentals instruction.

Modern commercial software has not completely met this goal. While modern CAI can deliver some kinds of instruction, it cannot deliver instruction on all subjects,

and is not adopted by all schools in the same way. In some cases, CAI is only used a kind of smart flash card system, and in others, computers are not used at all in music instruction. Commercial CAI like *MacGamut* or *Practica Musica* comes with mature curricula, but instructors cannot modify the format of the drills and exercises. Instructors can add new course materials, but only so long as they fit into the format the software uses, since the software cannot be modified. Commercial software does not allow researchers the flexibility to conduct their own research because they cannot modify the software.

In the early days of CAI, research led to new understandings of how people perceived music, and new innovations for delivering music instruction. The new techniques of instruction inspired more research, creating a cycle of research and innovation. The cycle of research and innovation created exciting opportunities for researcher and teachers alike since there were new things to study, and new tools to teach. Since most modern CAI does not allow researchers to capture data important to research, the cycle of research and innovation is broken. CAI users are locked out of the research process, so few new innovations appear based on CAI research. Ironically, even the current leading CAI titles lack scientific research to support their claims of efficacy, and are not designed to facilitate that research. Neither *MacGamut* or *Practica Musica* have any published research reports to support their claims of efficacy, and neither have been used in published research.

This goal of this thesis is twofold: to describe the downfall of CAI research and to describe a method for developing and distributing music CAI software called “open source” that might increase the amount of research that uses CAI. Open source is a

software distribution method that allows public access to program source code. Open source software distribution has become such a powerful idea in software production that the term open source refers to the software that is created, the social movement that supports open source software, and the mindset for developing the software. Most open source programs are free to use, and can be modified by anyone who has motivation to do so. By adopting open source software, CAI researchers can customize software for use in their experiments. By allowing researchers to freely share ideas and program code, research can be conducted that will lead to new innovations being introduced to CAI.

This thesis is organized as follows: in chapter 2, literature is reviewed to show how software was conceived, created, distributed, and used over the last 50 years. Chapter 3 provides an analysis of historical and current practices showing how patterns of music CAI development, distribution, and use have created an environment that limits research. Finally, an alternative method of software development and distribution is proposed in chapter 4 that can allow music CAI researchers to actively participate in the software development process. The use of open source music CAI will result in greater teacher-control over how software is used in education, and new opportunities for researchers to add to the body of knowledge.

## Chapter 2: Literature Review

*Research is an integral part of effective music CAI. Extensive research must be conducted prior to and during system development. The research areas of modeling the student's musical processes, instructional design, system configuration, and data saving and analysis are all basic to CAI. Once a system is operational continued research is needed to evaluate the system's effectiveness and to plan future modification and expansion. Finally, music CAI supports a variety of research applications to inquiries which have long engaged musicians.<sup>1</sup>*

-Rosemary Killam

### Introduction

Modern computers and software are tremendously powerful and can perform any number of tasks, from allowing a college instructor to send messages to their classes via email to providing musical accompaniment that follows the soloist. CAI like *MacGamut* or *Practica Musica* can drill students in musical fundamentals, aural skills, and even musical form. College music teachers and students certainly benefit from computers, but there is little research data to backup claims of software efficacy. Because commercial CAI software promises to deliver musical instruction that is equal to or superior to traditional classroom methods, software makers should provide research to back up their claims. Without data showing that CAI software is an effective method of teaching, the software is merely supported by anecdotal evidence. Early music CAI developers understood that research was fundamental to understanding how students would use and learn from their software. They conducted studies to empirically establish what techniques and tools were effective for delivering instruction via computer.

---

<sup>1</sup> Rosemary N. Killam, Philip Baczewski and Antoinette Corbet, "Research Applications in CAI" *College Music Symposium*, 21 no. 2 (1981): 43.

## CAI in Research: Mainframe and Micro Systems

In 1967 Wolfgang Kuhn and Reynold Allvin of Stanford University developed software that could determine the pitch accuracy of sung melodic patterns.<sup>2</sup> The software used a pitch extraction device and a mainframe computer to determine how accurately a subject could sing melodies.<sup>3</sup> The computer reinforced correct answers by evaluating pitch accuracy and rewarding students who sang the melodies well.<sup>4</sup>

Two years later, in 1969, Ned Deihl also developed ear-training software, though his system targeted instrumentalists.<sup>5</sup> He used modified tape recorders to judge the pitch accuracy of intermediate level clarinet players.<sup>6</sup> Although the system did not provide real-time results for the players, Deihl's work showed that computers could be programmed to evaluate musical sounds.

Don Bitzer first applied computers in college education when he built the PLATO (Programmed Logic for Automatic Teaching Operation) system at the University of Illinois in 1959.<sup>7</sup> His team of engineers and programmers received a grant from the National Science Foundation that gave them enormous resources to meet two goals<sup>8</sup>: (1) to investigate the potential role of computers in the classroom, or as Bitzer put it, "What

---

<sup>2</sup> Peter Webster, "Historical Perspectives on Technology and Music" *Music Educators Journal* 89 no. 1 (2002): 38.

<sup>3</sup> Ibid..

<sup>4</sup> Wolfgang Kuhn, "Computer-Assisted Instruction in Music: Drill and Practice in Dictation" *College Music Symposium* 14 (1974): 89.

<sup>5</sup> Peter Webster, "Historical Perspectives on Technology and Music" *Music Educators Journal* 89 no. 1 (2002): 38.

<sup>6</sup> Ned Deihl, "Computer-Assisted Instruction: Potential for Instrumental Music Education" *Council for Research in Music Education bulletin* 15 (1969): 1.

<sup>7</sup> Peter Webster, "Historical Perspectives on Technology and Music" *Music Educators Journal* 89 no. 1 (2002): 38.

<sup>8</sup> D. Alpert and D. Bitzer, "Advances in Computer-Based Education" *Science New Series*, vol. 167, no. 3925 (1970): 1583.

is educationally possible?”<sup>9</sup> (2) to help develop what he calls “an economically and educationally viable system incorporating the most valuable approaches to teaching and learning.”<sup>10</sup> Bitzer felt these two goals were closely related.<sup>11</sup> By discovering what computers could do to deliver effective instruction, and finding ways to make computerized education affordable, PLATO could use the best pedagogical ideas and be inexpensive enough for many people to use.

To meet the project goals, Bitzer and his team developed four successive CAI systems, PLATO I through IV, to see how far he could push the technology in solving educational goals.<sup>12</sup> The lessons learned from each of the systems were put into successive new systems. In 1972 PLATO IV was released and Bitzer declared that the final incarnation of the PLATO series was a success. He claimed that PLATO IV was “a large-scale system, which, even in a prototype version, would be justifiable in economic terms.”<sup>13</sup> Bitzer attempted to justify the expense of the PLATO system by measuring the cost of traditional instruction in a unit he called a “student-contact hour.” A student-contact hour is a measurement of time the a student receives instruction, so a teacher in front of a class of fifteen students is delivering 15 concurrent contact hours.

CAI on systems like PLATO proved to be effective, but very expensive relative to the education of one student.<sup>14</sup> However, when the cost of the system was spread out among many users, it could be compared to traditional instruction. Bitzer noted in 1970 that “presently available CAI systems ... entail total costs which range between \$2 and

---

<sup>9</sup> Ibid..

<sup>10</sup> Ibid..

<sup>11</sup> Ibid..

<sup>12</sup> Ibid..

<sup>13</sup> Ibid..

<sup>14</sup> D. Alpert and D. Bitzer, “Advances in Computer-Based Education” in *Science New Series*, vol. 167, no. 3925 (1970): 1583.



\$5 per student-contact hour” which includes the costs of student consoles, the central processing unit and management services related to the computer.<sup>15</sup> These figures were roughly ten times the cost of hiring a teacher to do the job.<sup>16</sup> In order to bring the price per student-contact hour down for the PLATO system, Bitzer tried to spread the cost of the system across as many users as possible. PLATO’s developers designed the system to support thousands of users, and allowed other mainframe computers to dial in to share the software. The PLATO team even custom-built their own student stations to save money instead of using commercially available products.<sup>17</sup> PLATO was flexible enough that it could be used for many different disciplines. In fact, by 1970, Bitzer commented that it was used in twenty different disciplines.<sup>18</sup>

By 1975 Fred Hofstetter had developed an entire aural skills curriculum for PLATO.<sup>19</sup> GUIDO (Graded Units for Interactive Dictation Operations) presented the student with an answer form, played a musical example, and then asked the student to answer questions about a musical example.<sup>20</sup> The software let the instructor print weekly reports on student progress, and allowed the instructor to adjust the level of difficulty for each individual student.<sup>21</sup> The software delivered examples to the students that were at the level the teacher had assigned.<sup>22</sup>

GUIDO not only helped drill music skills, it helped researchers study how students learned. Hofstetter said in 1981, “When the GUIDO project began it had not yet

---

<sup>15</sup> Ibid., 1586.

<sup>16</sup> Ibid..

<sup>17</sup> Ibid., 1587.

<sup>18</sup> Ibid..

<sup>19</sup> Fred Hofstetter, “Applications of the GUIDO System to Aural Skills Research 1975-80” *College Music Symposium*, 21 no. 2 (1981): 46.

<sup>20</sup> Ibid..

<sup>21</sup> Ibid..

<sup>22</sup> Ibid..

been determined how effective computer-based delivery was, compared to other forms of music instruction.”<sup>23</sup> Because of this, Hofstetter needed to discover if his system was effective or not. His first experiment with the GUIDO system compared student ear training using GUIDO with student ear training using the traditional tape laboratory.<sup>24</sup> Results showed that students learned better using GUIDO than from the tape lab, so the University of Delaware replaced its tape lab with a computer lab.<sup>25</sup> This initial study legitimized GUIDO and showed how powerful a research tool CAI could be.

Hofstetter conducted more experiments using GUIDO. Three of these studies were designed to determine patterns of errors in student dictation exercises.<sup>26</sup> The first study tested for confusion patterns in harmonic dictation.<sup>27</sup> The students were presented with harmonic dictation examples.<sup>28</sup> They could move through the examples at their own pace, listening to musical examples and choosing their answer on a touch sensitive screen. When they answered incorrectly, the software saved their incorrect responses, as well as the correct responses. Hofstetter used this data to determine that students have seven confusion tendencies for harmonic dictation: bass line confusion, wrong inversion, confusion of chord function, wrong quality, unperceived sevenths, unperceived roots, and favorite responses.<sup>29</sup> Hofstetter’s study allowed him to better understand student problems in perceiving the harmonic exercises and adjust his curriculum to address the confusion tendencies.

---

<sup>23</sup> Ibid., 47.

<sup>24</sup> Ibid..

<sup>25</sup> Ibid..

<sup>26</sup> Ibid., 48.

<sup>27</sup> Ibid..

<sup>28</sup> Ibid..

<sup>29</sup> Ibid., 49.

Hofstetter's second study using GUIDO examined student confusion tendencies in identifying chord qualities and inversions.<sup>30</sup> This experiment showed that students usually confused major chords with other major chord inversions, minor chords with other minor chord inversions, and augmented and diminished chords with each other.<sup>31</sup> Hofstetter used this particular study to dispel the commonly held belief that a major chord in root position is easiest for students to identify.<sup>32</sup> His data showed that students most accurately identify a minor chord in root position.<sup>33</sup> The results of this study are significant because they show how a research tool like GUIDO can improve a teacher's understanding of how students learn.

Hofstetter's third study looked for confusion patterns in rhythmic dictation.<sup>34</sup> The experiment showed that students confused simple note rhythms (quarter, half, and whole notes) with other simple note rhythms, and eighth, syncopated, and triplet rhythms with simple note rhythms.<sup>35</sup>

It is important to note the power that GUIDO gave Hofstetter as a researcher. Since students using GUIDO could work through the material at their own pace and on their own schedule, Hofstetter needed only to step back and allow them to do so. The computer allowed Hofstetter to collect a large amount of data and analyze it easily. To conduct the same experiment with manual methods, it would be necessary for a teacher to administer many tests to his students using a piano and prepared answer sheet, and then take hours to either enter the data into a computer or analyze the data by hand. GUIDO

---

<sup>30</sup> Ibid.

<sup>31</sup> Ibid..

<sup>32</sup> Ibid..

<sup>33</sup> Ibid..

<sup>34</sup> Ibid..

<sup>35</sup> Ibid..

automatically saved the correct and incorrect answers to a database for easy statistical analysis.<sup>36</sup>

GUIDO was not the only music curriculum written for PLATO. MUSFUND, written by Dorothy Gross and Roger E. Foltz in 1979, also operated inside the PLATO system.<sup>37</sup> MUSFUND was a collection of computer lessons that targeted music fundamentals and theory rather than aural skills.<sup>38</sup> MUSFUND consisted of “twenty drills and two practice tests and [included] music notation, scales, intervals, chords and terminology.”<sup>39</sup> The MUSFUND project was designed to determine if music CAI could effectively fill the need for good music fundamentals instruction.<sup>40</sup>

Gross and Foltz implemented the MUSFUND software at the University of Nebraska and the University of Minnesota in two different ways. At Nebraska the software was integrated into the regular curriculum, while at Minnesota the software was optional coursework outside of the regular curriculum. Gross and Foltz prepared a pre-test/post-test study and conducted exit interviews with the students.<sup>41</sup> They compared scores and interviews of CAI and non-CAI music fundamentals students and found that students who used MUSFUND showed higher degrees of comprehension of music fundamentals than those who did not.<sup>42</sup> Further evaluation of the results revealed that the rate of student attrition dropped when the software was used in the curriculum.

---

<sup>36</sup> Ibid..

<sup>37</sup> Dorothy Gross and Roger E. Foltz, “Ideas on Implementation and Evaluation of a Music CAI Project” *College Music Symposium*, 21 no. 2 (1981): 22.

<sup>38</sup> Ibid., 23.

<sup>39</sup> Ibid.

<sup>40</sup> Ibid., 24.

<sup>41</sup> Ibid.

<sup>42</sup> Ibid.

Rosemary Killam conducted several significant studies using CAI on microcomputers while at Stanford University. In one study, she used a PDP-10 timesharing computer system to study whether note length (the note durations were .1 and .2 seconds) had any effect on student performance in interval recognition.<sup>43</sup> She showed that note duration made little difference in the accuracy of interval recognition.<sup>44</sup> In a similar study on triads, she found that duration of tone (.1 and .2 seconds) *was* a factor in triad quality perception. Also, the mode of triad presentation, as chords, ascending arpeggios, and descending arpeggios, impacted how students performed.<sup>45</sup>

In 1984 Rosemary Killam stated that the requirements of music CAI are “sound, real-time interaction, individualization, student records and research.”<sup>46</sup> Her requirements for music CAI show a balance between the needs of the student (sound, interactivity, individualization) and the needs of the teacher (student records, research). Sound is an obvious requirement for music software, and sounds that are as close to “real” as possible help the students transfer their skills and knowledge in the real world. The software must also interact with the student in a manner that gives the user a sense of control, while at the same time presenting appropriate information and responses to keep the student progressing through the material. The software must be able to “individualize” content based on the progress of the student; the software must allow the teacher to assign tasks to certain students.

---

<sup>43</sup> Ibid.

<sup>44</sup> Ibid.

<sup>45</sup> Rosemary Killam, Paul V. Lorton Jr, and Earl Schubert, “Perception of Triads” unpublished research paper, Stanford University, 1975.

<sup>46</sup> Rosemary Killam, “An Effective Computer-Assisted Learning Environment for Aural Skill Development,” *Music Theory Spectrum* 6 (1984): 53.

Record keeping is an important task for CAI software. By building software that keeps records, teachers can not only evaluate how their students are progressing through the software, but also evaluate where students have the most difficulty with the software. Killam argues that this kind of ongoing research should be a part of good CAI.<sup>47</sup> Research on how students use software should allow teachers to recommend or implement changes to the software. For instance, if a significant number of students consistently perform poorly on one exercise while performing well on others, the teacher can examine the exercise that causes them difficulties to find ways to improve it. The teacher can then change some aspect of the program and study whether those changes make a difference in student performance.

Kuhn, Killam, and Lorton discussed how computers are necessary and effective research tools for the largely unexplored realm of music education.<sup>48</sup> They state:

Music, although a traditional and historical area of formal education, has many areas awaiting systematic exploration. The amount and precision of data required for research in these areas necessitates a computerized system. ... With the musical sound presentation under computer control and with the system monitored [sic] and recorded in detail, precise research and flexible instruction are made feasible.<sup>49</sup>

The computer allows music, a subjective medium, to be studied objectively. The data collected by the computer can be used to understand how students learn music. Better understanding of student perception helps teachers create more effective teaching strategies.

---

<sup>47</sup> Rosemary N. Killam and others, "Research Applications in CAI" *College Music Symposium*, 21 no. 2 (1981): 37.

<sup>48</sup> Paul Lorton Jr., Rosemary Killam, and Wolfgang Kuhn, "A Computerized System for Research and Instruction in Music" *Computers in Education: Proceedings of the IFIP 2<sup>nd</sup> World Conference*, 1975.

<sup>49</sup> *Ibid.*

## The PC and New Technology

During the 1970's, as computers were becoming more common, researchers started writing articles discussing how to build CAI systems.<sup>50</sup> These “how-to” articles were new to the community, and reflected a shift in scholarly publication away from research. By sharing their thoughts on their own software development cycle, researchers shared the knowledge they gained from building the software with new CAI programmers.

Computers became much more commonplace in the music school with the introduction of the personal computer (PC), and the audience for CAI grew because of it. PCs allowed users to build and use software with relative ease. Researchers who had used mainframe computers to conduct research could implement their software on PCs for less money, and they had a potentially larger audience for their software, but they lost the centralized data collection that mainframe computers used. For CAI researchers who were used to the centralized data storage of mainframes, this was a serious impediment to experimental design. Programmers had to deal with the data storage hurdle, or write software that did not save data. Some programmers authored how-to articles, while others started selling their software to the burgeoning PC market. Research was becoming less of a focus for the CAI community.

---

<sup>50</sup> Killam contributed several of such articles, including: Paul Lorton Jr., Rosemary Killam, and Wolfgang Kuhn, “A Computerized System for Research and Instruction in Music” *Computers in Education: Proceedings of the IFIP 2<sup>nd</sup> World Conference*, 1975; Rosemary Killam, Philip Baczewski and Antoinette Corbet, “Research Applications in Music CAI” *College Music Symposium* 21 no. 2 (1981); Rosemary Killam, “An Effective Computer-Assisted Learning Environment for Aural Skill Development,” *Music Theory Spectrum* 6 (1984).

PCs were of several different types, and many were not compatible with other kinds of PCs. This meant that some programs could only run on certain kinds of hardware. According to G. David Peters, “the perplexing question of what computer to adopt became a major issue for the educational community.”<sup>51</sup> The music community was split into factions, each rallying around a particular type of computer and the software that ran on it. Apple computers gained a following due to their graphics capabilities and ease of use. The Macintosh computer, with built-in sound capabilities, emerged to replace the Apple IIe in the mid 1980s.<sup>52</sup> Most PCs, including the Apple IIe, included everything the programmer needed to start developing software, including a programming language, like BASIC or HyperCard.

BASIC (Beginner's All Purpose Symbolic Instruction Code) was a commonly used programming language that came pre-installed on computers from Apple, IBM, and Commodore.<sup>53</sup> It used an English-like syntax and was fairly easy to understand. BASIC emerged in 1964 and was used commonly in micro- and mini-computer programs, so much of the body of knowledge learned by programmers using BASIC on a mainframe could be transferred to the PC. Since BASIC was in common use on many machines for a long time, software written for PC in BASIC was fairly mature compared to the age of PC technology.

HyperCard was another very prominent language in CAI development. Like BASIC, it used an English-like syntax for source code, but it ran only on Macintosh

---

<sup>51</sup> G. David Peters, “Music Software and Emerging Technology” *Music Educators Journal* 79 no. 3 (1992): 24.

<sup>52</sup> Ibid..

<sup>53</sup> Gary Wittlich, John Schaffer and Larry Babb, *Microcomputers and Music* (Englewood Cliffs, NJ: Prentice Hall Professional Technical Reference, 1986), x.



computers. Released in 1987, it gained rapid adoption because of its integrated media capabilities and ease of use. Users could build HyperCard applications to display information or manipulate data. HyperCard applications utilized the familiar Macintosh graphical user interface, so they were visually appealing with only a little work.

In 1983, at about the same time that the PC was gaining momentum, digital keyboard manufacturers agreed on a set of standards for describing musical events in a digital format: Musical Instrument Digital Interface (MIDI).<sup>54</sup> MIDI is designed to enable a computer to communicate with electronic instruments. MIDI became very successful since it effectively defines a musical language for electronic equipment, allowing a computer to communicate a broad range of external devices. MIDI is platform independent, and also allows for musical input to the computer from MIDI devices.

Though MIDI's original target was the music production industry, it found many willing users in music CAI development. G. David Peters first used MIDI in conjunction with computerized music instruction in 1984.<sup>55</sup> Peters' software, called *Keyboard Blues*, used a MIDI keyboard and worked with both IBM/DOS and Apple computers.<sup>56</sup> Peters reported that more than 50 programs using MIDI for music instruction were released in the eight years (1984-1992) following the release of his software.<sup>57</sup> The 1980s saw a huge increase in the number of music CAI programs that were produced and distributed.

---

<sup>54</sup> Peter Webster, "Historical Perspectives on Technology and Music" *Music Educators Journal* 89 no. 1 (2002): 38.

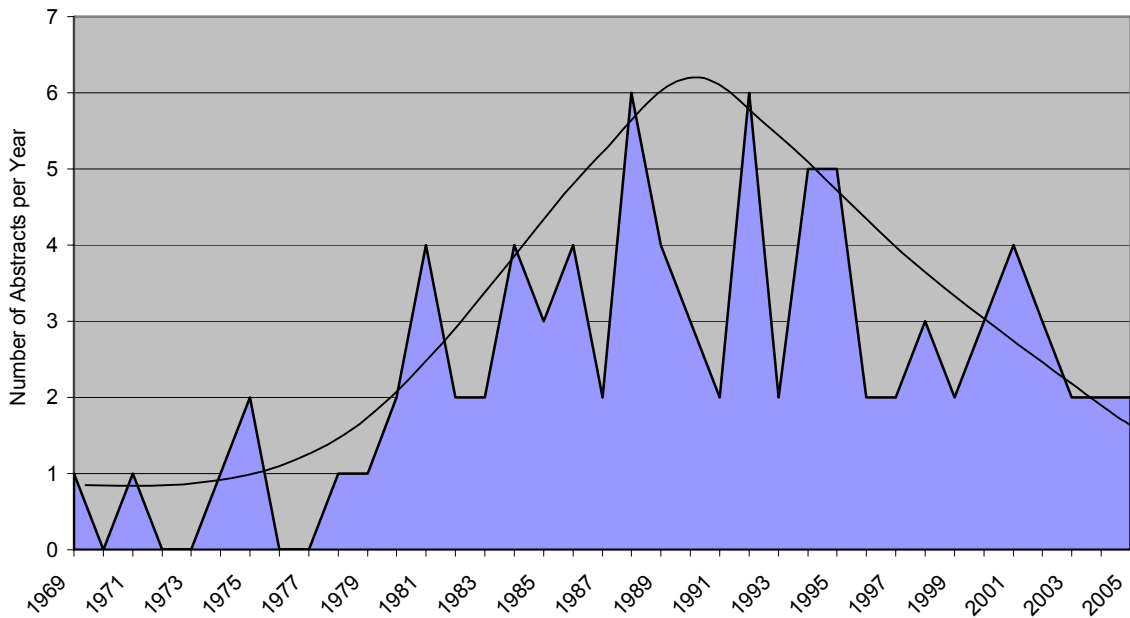
<sup>55</sup> G. David Peters, "Music Software and Emerging Technology" *Music Educators Journal* 79 no. 3 (1992): 22.

<sup>56</sup> Ibid..

<sup>57</sup> Ibid..

Despite the growing number of CAI programs since the advent of the PC, the use of CAI for research purposes has become less frequent since the early to mid 1990s. It is easiest to see the trend towards less research by examining the number of dissertations published that use CAI for research. Another indicator of diminishing research is the frequency of publication of research findings in scholarly journals.

The frequency of CAI oriented dissertations is shown in Figure 1. The data for Figure 1 was collected from the Dissertation Abstracts database OCLC FirstSearch. The following search terms were used: “kw: computer and kw: music and kw: instruction.” The search yielded 187 records that were filtered manually to make sure the data was relevant to CAI research in music. It was discovered that 88 relevant dissertations were left. These were grouped chronologically to develop the graph shown in Figure 1. The data for this graph and the search results are shown in Appendix A. The graph in Figure 1 shows a bell curve, with the most research being published between 1989 and 1993. The earliest dissertation shown on the graph is from 1969, and research increased in frequency in the two decades following that point. In 1989 and in 1993 there were 6 dissertations published that used CAI for research. The frequency of dissertations on music CAI diminishes after 1993 but does not completely disappear. The graph indicates that as computers became more popular and widespread, research increased. However, after 1993 there were fewer dissertations published per year that used CAI in music research.



**Figure 1: CAI research dissertations per year**

Another indicator of how CAI research is diminishing is the amount of attention it receives from music organizations. For example, in 1981, the College Music Society (CMS) dedicated an entire issue of their journal to CAI and research. However, since then the CMS has not repeated this act, nor have they focused on computers in research. Likewise, *The Journal of Music Theory* published several articles on CAI in the 1960's and 1970's, but none afterwards. *Music Theory Spectrum* printed a number of articles on CAI in the mid to late 1980's, and a fair number of software reviews. However, after that the amount of CAI research to appear was limited. *Music Theory Online*, founded in 1993, has no articles on CAI or computer music instruction. The Association for Technology in Music Instruction (ATMI) holds annual conferences, but the focus of the

presentations is more geared towards how-to than presentation of research findings.<sup>58</sup> Finally, the Special Interest Group for Computer Uses in Education, which published many research and how-to articles on uses of technology in the classroom (not specifically music related), was disbanded after 2001.

Many of the more recent articles about CAI are reviews of functionality and user interfaces, and do not discuss CAI's current research potential. For example, Deron McGee's 2000 survey of aural skills software notes how research in cognitive science from the past two decades helps music teachers understand how students perceive music,<sup>59</sup> and he even states that the future of aural skills instruction relies on conducting research and integrating it into music teaching materials.<sup>60</sup> Yet, in his discussions of each of his four reviewed software packages he never mentions how the software has been or can be used for research. He focuses on how the software looks, how the teacher can use it to assist instruction, and whether or not the software supports various technologies like USB.

### **Current Music CAI Software**

There are many CAI programs available today; two of the most widely distributed and respected programs are *MacGamut* and *Practica Musica*.<sup>61</sup> *Practica Musica* was released in 1987 by *Ars-Nova Software, LLC*.<sup>62</sup> It includes exercises in dictation as well

---

<sup>58</sup> Conference schedules for 1999-2005 are available at [http://atmionline.org/index\\_conf.html](http://atmionline.org/index_conf.html).

<sup>59</sup> Deron McGee, "Aural Skill, Pedagogy, and Computer-Assisted Instruction: Past, Present, and Future," *Journal of Music Theory Pedagogy* 14 (2000): 115.

<sup>60</sup> *Ibid.*, 134.

<sup>61</sup> Deron McGee, "Aural Skill, Pedagogy, and Computer-Assisted Instruction: Past, Present, and Future," *Journal of Music Theory Pedagogy* 14 (2000): 115.

<sup>62</sup> *Ibid.*

as theory, and employs a mastery model for drill and practice.<sup>63</sup> Ann Blombach released *MacGAMUT* only a year later, in 1988.<sup>64</sup> *MacGAMUT* includes drills for aural skills using the mastery model.<sup>65</sup> Both packages were originally written for the Mac platform, though both now run on both Macintosh and Windows.<sup>66</sup> Each program offers built-in sounds as well as MIDI for listening to musical examples.<sup>67</sup> The most important feature of both programs is that they save data to allow teachers to assign letter grades and make sure that students are progressing through the computer curriculum.

However, even though *MacGamut* and *Practica Musica* save data on student performance, no academic research has been performed using either program. The lack of research is largely due to the way the software is setup and marketed. Researchers cannot legally modify the software to trap new data, and the programs do not offer tools to examine the existing data. The data saved by both programs are in a proprietary format and standard data compilation tools cannot read their formats to access the stored data.

### **The Internet and New Standards**

In recent years there have been some important technological developments, including the Internet, open source, and database solutions for data storage. The Internet enables people to communicate instantly over distances, allowing new social groups to form, and new ideas to be shared. Because of its popularity, the Internet has helped drive

---

<sup>63</sup> Ibid.

<sup>64</sup> Robert Skinner, "Music Software," *Notes*, 2nd Ser., Vol. 45, No. 3. (1989): 537.

<sup>65</sup> Deron McGee, "Aural Skill, Pedagogy, and Computer-Assisted Instruction: Past, Present, and Future," *Journal of Music Theory Pedagogy* 14 (2000): 115.

<sup>66</sup> Ibid..

<sup>67</sup> Ibid..

the price of computer hardware down as well, allowing almost anyone to use computers. The volume of traffic on the web created a demand for centralized data storage programs, called databases, to keep track of web users' information. New technologies and approaches have arisen to meet the needs of Internet users and providers., including open source software and database technology.

Open source software is a number of things. It is a methodology for producing software with public program source code. It is a label for the software produced by the process. And finally, it is the name of the social movement that produces and distributes the software. Open source software is free to download, use, modify, and distribute. The source code is available so that other people can modify it or improve it, and give the improvements back to the community. Open source software is usually released under a license agreement like the General Public License (GPL), which effectively makes the software source code public property. The conditions of the GPL allow anyone to view and use the code under the condition that they do not sell it or claim it as their own property.

Open source software development has flourished because of the communication channels provided by the Internet. Programmers can email other programmers, post their completed work to websites, and download other people's software to see how it is built. The open source movement has produced software that powers most of the Internet, including the *Linux* operating system, the *Apache* web server software, languages like PHP, and data storage programs like *MySQL*.

Data storage is important on the Internet, so powerful data storage mechanisms have arisen to meet the need. Web applications track millions of pieces of data, including

user logins, security codes, and content. Programs like *MySQL*, *MS SQL Server*, *Oracle*, and *PostgreSQL* use a standards-compliant language called Structured Query Language (SQL) to insert and retrieve data. SQL allows programmers to manipulate data the same way in every database. Extensible Mark-up Language (XML) is another important data storage and manipulation standard. It is a way of building a self-describing data storage format using plain text files.

The Internet provides new tools and languages for running software on most kinds of computers. Most modern computers support a web browser, and web browsers are designed with standard capabilities, like rendering HTML into viewable pages. Web browsers also support plug-ins like Java applets and *Flash* programs that can play movies, mp3 files, and MIDI files. Cross-platform media capabilities and centralized data storage offered by web applications help overcome some of the problems that PC programmers and researchers encountered before the Internet.

The idea of taking advantage of both the power of distributed and centralized computing has resulted in some interesting web applications for music education. Websites such as *MusicTheory.net* deliver tutorials, drills, and utilities for learning music fundamentals and music theory. *MusicTheory.net* does not track student performance or activity over the long term, and is limited to tracking performance for each exercise. *eMusicTheory.com* allows teachers to establish virtual classrooms for their students, give out simple theory assignments, and have the computer grade the assignments. The *eMusicTheory.com* software can be run from the website or downloaded to run offline, giving the student a chance to practice at home without an Internet connection. The software is cross platform, and integrates with the MIDI capabilities of modern

computers. *eMusicTheory.com* collects student usage information automatically for the teacher, including score, time required, and who has completed the exercises. When used from the website, the software saves all the information to a database so that teachers can view reports later on.

*eMusicTheory.com* presents an interesting possibility of delivering CAI over the web, but it has some drawbacks. The program cannot be easily modified to trap different data, and the raw data is not made available through the web interface, so it does not offer any research opportunities to its users. *eMusicTheory.com* also has technical and logistical flaws in its user interface, so people are not likely to adopt it. For instance, the MIDI compatibility is not completely cross platform, and the administrative interface is not intuitive or self-explanatory.

While CAI seems to have made steady inroads into the music school, it also has lost its power as a research tool. Early CAI was tied to research, and the information gained from early CAI research was important to the music community. Early CAI pioneers proved the value of research for the music community, indicating CAI that does not promote research does not serve the best interests of the music community. While the current CAI market supports a number of commercially successful programs, there has not been a lot of new growth or innovation in the past decade in CAI. To further understand the problem faced by CAI users and researchers, we must understand how people develop, distribute, and use CAI software.



### Chapter 3: CAI and Society

*Electronic technologies and the industries that supply them, are not simply the technical and economic context within which 'music' is made, but rather, they are among the very preconditions for contemporary musical culture.*<sup>68</sup>

- Paul Theberge

#### Introduction

The use of computers in music instruction creates a complicated and rich social framework for gathering and sharing information. The early adopters of computers in music education shared new ideas through articles and conferences, and adapted existing technologies to create new ways of teaching and learning. In examining the past, the important issue is not the hardware, or the software itself, but how people chose to use computers in music education, and how they shared that information with others. Different groups of people adopted varying technologies to meet their needs, and in doing so gave the technology meaning and life.

CAI was also important to music schools because the early adopters legitimized its use through research. The lessons the early CAI programmers learned impacted what features were built into new CAI, how that CAI should be used, and who should use it. In the early days, the context of music CAI software was that of exploration, cutting edge science, and improved pedagogy. The future was filled with promise. CAI developers took advantage of many different kinds of technology like MIDI to improve and bring new meaning to their software. With time and changes in computer culture, CAI became a commercial product and research became less of a concern for CAI developers.

---

<sup>68</sup> Paul Theberge, *Any Sound You Can Imagine: Making Music/Consuming Technology* (Hanover, NH: Wesleyan University Press of New England, 1997), 151.

Research was still an important aspect of academic music culture, but because of larger societal pressures CAI was no longer viewed as a research tool.

This chapter addresses the social pressures that changed CAI into a commercial product. Among the social forces at work are the changes in computer culture over the decades, the fact that music CAI programmers are often music theory professors, and the loss of knowledge due to technology obsolescence. People have come to use CAI in specific ways because of changes in society, the demands placed on music teachers who develop software, and the volatility of computer knowledge, hardware, and software.

Modern CAI software offers effective pedagogy based on the research conducted decades ago, but it does not allow researchers to capture new data. For example, *MacGamut*, a well-known and widely used CAI package, does not allow researchers to save new data to see how students use the software, nor does it allow researchers to examine the data captured in new ways.<sup>69</sup> Students and teachers can only use *MacGamut* in specific ways, limiting opportunities for researchers to try new ideas.

Since modern CAI developers do not set out to limit research opportunities for others, there must be other forces shaping the development of CAI over the years, outside of the developers' desire to create software. Some of these cultural forces are rooted in the way music teachers teach, the programming tools and languages that are available, and the way musicians view computers.

---

<sup>69</sup> According to Melba Leyshon of MacGamut Software, Inc. in an email to author (November 4, 2005), MacGamut enjoyed distribution of 25,000 copies in 2005 alone.

## **The Culture of CAI Development**

Development of PLATO or other mainframe software in the 1960s practically demanded a long-term structured approach, since the resources required to develop mainframe CAI applications were so great. A single individual simply lacked the time, knowledge, and money to put together the entire system alone, so formal processes evolved to communicate system parameters and results of tests to other team members. The PLATO project required a staff to install, configure, and maintain all of the hardware and software the machine used, and Bitzer even had a staff of engineers to develop plans and working models for the student workstations.<sup>70</sup> To work with the varied groups of engineers, support staff, programmers, and testers, Bitzer must have devised a process for directing the project personnel and communicating his plan. Every project involving more than one person has a communication and management structure. Bitzer did not publish descriptions of the group dynamics of his project, but given the longevity and success of the PLATO project, we can be assured that there was some successful structure in place.

The advent of the PC in the late 1970s changed how software was developed. Hobbyists could write PC software on their own computer. A single person could embark on a software project with little planning, and a relatively modest budget. The CAI production process became less of a group effort, and less structured. Many programmers failed to give adequate attention to the planning process, and did not

---

<sup>70</sup> D. Alpert and D. Bitzer, "Advances in Computer-Based Education" in *Science New Series*, vol. 167, no. 3925 (1970): 1583.

manage their projects well.<sup>71</sup> As a result, the increasing number of CAI programs released to the public were of poorer quality. Many of the programs, written in BASIC or HyperCard, were designed for personal use, not for research or publication.

Programmers who produce music CAI are often music teachers. There are three inherent problems with the “music teacher as programmer” model. First, few music teachers have a significant background in software development, having spent their formative years practicing and studying music. Second, few music teachers have an abundance of time on their hands to handle teaching and *all* of the tasks involved in bringing a successful software product to completion, including software design, project management, software development (coding), quality assurance, marketing, and support. Finally, most schools of music do not provide an environment conducive to software production.

Music teachers rarely have any formal training in software development. Many are drawn into CAI programming to help their students, or out of personal interest. Research by Steve McConnell shows that even highly trained computer programmers often lack the knowledge and skill to successfully bring a project to completion.<sup>72</sup> According to McConnell, successful software production takes experience more than the ability to write program code: it takes good planning and management, and adherence to software development processes that are proven to reliably bring software to

---

<sup>71</sup> Steve McConnell, *After the Gold Rush: Creating a True Profession of Software Engineering* (Redmond, WA: Microsoft Press, 1999), 4.

<sup>72</sup> *Ibid.*, 11.

completion.<sup>73</sup> Therefore, in order to successfully produce software, the CAI developer needs experience, project management skills, discipline, as well as the ability to program.

Steve McConnell claims that 75% of software projects use something called code-and-fix development because they lack discipline, planning, and good management.<sup>74</sup> Code-and-fix development is a programming style in which a programmer undertakes a software project with inadequate planning, resulting in frequent fixes and changes of scope. Code-and-fix programming results in a high risk of project failure, longer timelines, and poor program code.<sup>75</sup> This approach to programming is widespread because it provides a false sense of progress. The programmer can see working code early on in the process, but the lack of planning involved in code-and-fix programming ends up costing large amounts of time later in the project as portions of the program need to be rewritten to accommodate changes. More experienced software companies like Microsoft or IBM reduce their risk of failure by resisting the urge to produce code before they have a clear plan. Music CAI developers, struggling to show progress in their CAI project, often do not avoid this trap. By making this mistake, they cost themselves valuable time and money.

Music teachers often lack the time in their schedules to spend huge amounts of time on a software project. Producing software is time consuming, and college music teachers are subject to the same duties as other faculty, including teaching, scholarship, and research. During a software project, the initial phases of software production, including gathering requirements, designing data structures, and planning the execution

---

<sup>73</sup> Ibid., 20.

<sup>74</sup> Steve McConnell, *After the Gold Rush: Creating a True Profession of Software Engineering* (Redmond, WA: Microsoft Press, 1999), 11.

<sup>75</sup> Ibid..

of the project, takes many man-hours. The whole project, including planning, coding, testing, quality assurance, and implementation takes substantially more time. Even after a software project has been completed and is being used, it may have problems that require support hours, which takes time away from improvements and upgrades. Few teachers can provide the kind of time it takes to develop and support a successful software project along with their other teaching duties. In order to complete the software successfully, the CAI programmer must either be extremely knowledgeable and efficient, must cut corners, or must have help.

Music departments rarely have more than one or two people who know how to program computers. The music teacher/programmer might have a colleague who has some technical ability, but it is rare for him to find a colleague who can help develop software. Music schools also do not offer experienced project managers to support the CAI project. So the entire workload lies on the shoulders of the teacher/programmer, making it more likely for him to run over time and budget by a wide margin.

The final problem with the aural skills teacher as programmer model comes from the duty of faculty members to “publish or perish.” College teachers are not only required to teach, but to write for scholarly journals, publish books, and contribute their knowledge to the college community. The teacher/programmer can publish his software, or articles *about* his software in journals or in a book. These publication about CAI as a teaching tool removes the focus from research. CAI becomes a product for teaching and publication, but not for research.

## Software as a Product

When CAI software is finished, it can be sold to recoup some of the costs of development. There are two avenues for selling CAI: publishing it through an established publisher, or selling it on his own. While college tenure committees look upon publication through a reputable publisher more favorably, both paths have implications for the software and for the programmer.

Some CAI developers choose to distribute their software through traditional textbook publishing companies. Publishing companies, especially textbook publishers, actively try to find new ways to sell more books. One way they do this is to pair software with an existing textbook. This pairing often makes sense in terms of pedagogy, since an existing textbook can provide tested and proven materials, while the CAI can provide drills and homework. Programs like *Computer Assisted Software Project for Aural Skills Reinforcement (CASPAR)* and *Music for Ear Training* coordinate with established ear-training textbooks.<sup>76</sup> By pairing software with a textbook it has a double effect on the legitimacy of the software. It strengthens the legitimacy since the software is paired with a prominent textbook, but it also weakens the legitimacy since the software is seemingly an add-on to the textbook. This arrangement can be positive for the programmer since most of the work of distribution and marketing is taken care of by the publishing company, but the publisher can discontinue the software if it does not improve book sales.

If the CAI programmer decides to market the software on his own, he becomes not only a programmer, but also an entrepreneur. With this arrangement he must manage

---

<sup>76</sup> Leo Kraft, *New Approach to Sight Singing, 2nd Edition* (New York: W.W. Norton) and Benjamin, Horvitt, and Nelson, *Music for Sight Singing, 3rd Edition* (Belmont, CA: Wadsworth)

business aspects like advertising and budgeting, in addition to programming, support, and academic duties. The software changes status from research project to business development, and the software's viability is tied to making money. Having two jobs at once puts the teacher/programmer in an awkward position, since his or her academic duties are not lessened. Even if the software successfully provides good instruction to the students and eases the teaching workload, the instructor is still obligated to teach, conduct research, and publish while running the software business.

Ann Blombach, who wrote *MacGAMUT*, was a college music teacher before she retired. She created MacGamut Music Software International long before she retired to help her sell her software. By forming her own company instead of using a traditional publisher, she took on many of the challenges of distributing software. However, she also was able to hire extra help and shift some of the pressure of software development, marketing, and distribution off of her own shoulders. A software company can focus on the business of producing software, and does not deal with teaching and scholarship like college teachers. Software companies hire software specialists and project managers to make the process of developing software more streamlined and stable. The software company can even suffer the loss of a programmer and remain viable. Furthermore, the software company is under no obligation, even implied, to share their technology with the community at large. College professors must "publish or perish," so much of the research they perform is shared with the academic community, but a software company can keep their methods and techniques secret.

When the teacher/programmer completes and distributes his software, there are still some pitfalls to be dealt with. According to Michael Tiemann:



The simplistic view of a software company is that once you've created some software that people want to buy, the act of printing copies of that software and distributing it is not unlike printing money: the cost of goods is negligible, and the margin nearly perfect. ... The concept of software support was seen as a degenerate by-product of some flaw in the software product process, and that by minimizing software support investment, one could maximize profits.

This not only frustrated users, but it was bad for the software as well. Features that were easy to implement were often dismissed as "non-strategic." Without access to source code, features that customers would otherwise be able to implement themselves remained points of speculation and contention. And ultimately vendors (and their marketing departments), not customers, defined the arena of competition with a myriad of useless but easy-to-express features. Free market economics had been turned upside down.<sup>77</sup>

Tiemann's experience showed that keeping source code hidden from the public actually drove the market away from what the end users needed and wanted. This effect on the software market can be seen in modern CAI: software reviews and articles focus on features that have little to do with how CAI can be used to produce a better learning environment, such as the attractiveness of the user interface.

Tiemann's observations also are relevant in terms of technical support and software upgrades. Software must be kept current so that it runs on current operating systems, takes advantage of technical improvements in hardware, and adds features so that users will pay for new versions. If these things do not happen, the software will become obsolete. Software updates are an integral part of the software development lifecycle, and are crucial to the adoption of software. According to McConnell, deciding when to release software is not a matter of eliminating defects, but deciding how many

---

<sup>77</sup> Michael Tiemann, "Future of Cygnus Solutions: An Entrepreneur's Account" in *Open Sources: Voices from the Open Source Revolution*, ed. Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA, 1999), 77.

critical, serious, or cosmetic defects are acceptable.<sup>78</sup> Because of this, new software is *bound* to have problems, and the programmer must release updates to fix the problems and improve functionality.

The stress of producing and maintaining software can be enough to drive programmers out of the software business. Without a programmer to support it, CAI software becomes obsolete, and the ideas expressed in source code are lost to history. Even commercial software programmers who specialize in writing software for software producing organizations have a fairly high attrition rate due to stress, resulting in an industry demographic dominated by the young, while older workers move on to less high-pressure jobs in management or other fields.<sup>79</sup>

## **Open Source Software**

The problems of programmer attrition and closed source programming create a void of information for the CAI community, but open source programming offers a solution. Open source software is released with all the source code, available for viewing or changing, under the condition that no one can “own” the source code. People can generally use open source software any way they decide to, modify it for their own purposes, and distribute it so others can enjoy the improvements. Open source programs can be shared among many different programmers, each working on a part of the program that interests him. Sharing the workload not only relieves stress on programmers, but it creates a more stable knowledge base. No longer does one person

---

<sup>78</sup> Steve McConnell, *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last* (Redmond, WA: Microsoft Press, 1998), 224.

<sup>79</sup> Steve McConnell, *After the Gold Rush: Creating a True Profession of Software Engineering* (Redmond, WA: Microsoft Press, 1999), 30.

control all the knowledge of how a CAI program works. If a programmer *does* stop working on the program, all of the source code that powers his section of the program is available for someone else to pickup where he left off.

Many businesses, individuals, and even university computer science departments have discovered that open source software allows them to maximize their investment in software. Software like the open source Linux operating system has a tremendous following from programmers who use it and contribute to it.<sup>80</sup> Businesses adopt Linux because they know that there are thousands of programmers improving it and supporting it. Linux programmers add features that they themselves need and want, so the software more accurately provides for users needs. Programmers benefit from open source programs because they can study software code that is used in “production” business systems to learn how it was built, and how to build their own.

The open source approach allows for the establishment of a freely available body of knowledge for music CAI software developers. New programmers wishing to learn how the program works need only examine the source code of existing applications. Teachers wishing to study how students react to different instruction delivery methods need only implement their new ideas on top of existing software.

One foreseeable difficulty in adopting open source practices in music CAI development is the issue of ownership. Because many people see software as a tangible good, they expect to pay a software company to use software. People expect the software company to own the rights to the software, and to control access by locking down the

---

<sup>80</sup> The name “Linux” is a reference to Linus Torvalds, the creator of the operating system kernel. Linux stands for “Linus’s Unix.” Torvalds did not create the entire operating system. Independent contributors built the rest of the operating system over the course of several years, which can more formally be called GNU/Linux. GNU is a looping acronym meaning, “GNU’s Not Unix.”

source code and charging users to use the software. However, there is a software-licensing schema called the General Public License (GPL) that enables the public to own software. The GPL outlines how any person can use and distribute the software and source code. It is a legally binding licensing agreement that prevents businesses from appropriating public code as their own, thereby protecting the integrity of open source software. There are many software packages already available under the GPL, including *MySQL* database server, *Apache* web server, and many varieties of the *Linux* operating system.

Richard Stallman notes that since the 1980's people expect software to cost something, and as a result hesitate to adopt software that is free.<sup>81</sup> This is not strictly true on the Internet, where people routinely use web software like email without hesitation. But for desktop software, Stallman's statement is true. End users expect technical support and updates from their software manufacturer to keep their software current, and they have grown accustomed to paying for those things. They are uncomfortable adopting software that is free because they expect to get what they pay for and are unsure how long free software will be available or usable. In reality, open source software is free, and some of it is very good, and is updated regularly to fix defects and improve features.

The idea of ownership is also important because it is associated with the ability to make money from the software. The software industry in general maintains fairly stringent controls over their software source code. The theory is that if their source code were to be put into the hands of the public, the software company could no longer charge

---

<sup>81</sup> Richard Stallman, *Free Software, Free Society* (Boston, GNU Press, 2002), 35.

money for the software. However, open source software can support profitable businesses. Open source software businesses do not sell the open source software as their own since it is public property, but they can offer value-added services like hosting, distribution, custom programming, and support. They can even build and sell special modules and features. For example, the *Linux* operating system is developed and supported in part by companies that provide installation services, support, custom modifications, and other services.

While existing CAI packages like MacGamut and Practica Musica limit research opportunities in CAI, open source is an opportunity to change that. Open source can provide researchers with free CAI tools to undertake studies. That research can be used to improve the CAI and provide the best pedagogy possible. Open source software distribution combined with the power of modern computers can give researchers opportunities they do not have today. By developing a strategy to take advantage of the exciting opportunities offered by open source, and by providing carefully developed tools for researchers, it is possible to build CAI that both offers good pedagogy and research opportunities.

## Chapter 4: Proposal for a Open Source CAI System

*[The] greatest scarcity in the United States is not technical innovation, but rather the willingness to work together for the public good. It makes no sense to encourage the former at the expense of the latter.*

–Richard Stallman<sup>82</sup>

### Introduction

In this chapter a program called *Mobius* is proposed that is designed to deliver music fundamentals drills and dictation exercises via a web interface, and distributed as an open source program.<sup>83</sup> *Mobius* will be made available under the GPL so that researchers can download and modify the source code and contribute their ideas to the music community as source code. This proposal discusses the goals of the software project as well as who will build the software, what tools will be used, and how team members will work together. Detailed storyboards displaying key elements of the program will be shown. One of the key elements is a “data driller” that allows a researcher to create custom reports from existing data saved in the software. By providing researchers with powerful data extraction tools, the *Mobius* system can be used for research purposes.

### Mobius Software Goals

The *Mobius* software project has four goals. (1) to provide superior features and functionality. (2) to be designed in such a way that researchers and new programmers

---

<sup>82</sup> Richard Stallman, *Free Software, Free Society* (Boston, GNU Press, 2002), 124.

<sup>83</sup> *Mobius* refers to a Mobius strip, a loop that has one continuous side. It was chosen because it symbolizes a continuous loop of research and innovation in music CAI: research leads to innovation, and innovation leads to research. There should be no boundary between the two.

can quickly and easily modify and improve the program. (3) to be platform neutral, i.e., the software must be able to be delivered over the web. (4) to use centralized data storage.

In order for *Mobius* to make a positive impact in research, it must have excellent functionality so that people will want to use it. It should include a complete curriculum for aural skills and music fundamentals, including modules on written and aural intervals, scales, chords, as well as melodic and harmonic dictation. The software should be flexible enough to allow the instructor to configure and assign specific exercises to their students.

The curriculum for *Mobius* should be modular. Each skill the program teaches should be contained inside a program module that can be installed into the *Mobius* system and configured to behave the way the teacher wants. A module is a section of program code that contains all of the instructions necessary to deliver drills or exercises on a particular subject. For instance, an interval drill module would contain all of the code necessary for the teacher to assign interval drills to students, the students to use the drills, and the system to save their data. Each skill module should allow several input methods. For instance, for intervals, scales, and chords students could either notate the interval or identify the interval type from a list. Modules should also allow the instructor to enter new examples to present different material.

The software source code should be well structured, well documented, and easy to understand so that programmer can contribute new modules and new functionality to the system. The source code should be divided into sections that make up components of the software. For example, one segment of code controls the user interface, another

communicates with the database, and a third segment that makes grading decisions. Grouping similar code together is called n-tier software design. Each “tier” or layer performs a specific function. Figure 2 shows how the tiers communicate with each other. Each layer only communicates with the layer directly above or below it. The top layer, the presentation layer, is the user interface code, and can only access data by communicating with the layer beneath it. N-tier design keeps program code well structured. All the database calls are in the bottom layer, all the complicated decision-making code is in the logic layer, and the code that controls the user interface is in the presentation layer.

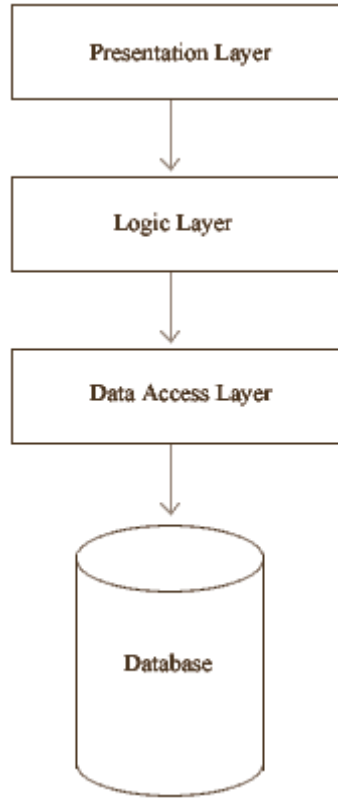
All code will be documented internally and externally. Internal documentation- comments inside the source code- helps the programmer understand what the code does. External documentation- information in print and digital format- should include information on how to use the software, how it works, how it is designed, and even specifics on what various sections of source code do.

*Mobius* should be cross-platform, MIDI compatible, have centralized data storage, and utilize the Internet. To meet these goals, *Mobius* will be written in PHP and *Flash*, and use *MySQL* for data storage. There is a plug-in for *Flash* called *flashMidi* that allows it to take advantage of MIDI capabilities, making it an ideal solution for developing a front end for web-based music CAI software.<sup>84</sup> *FlashMidi* is open source, so it is free to download and use. *Flash* movies can include the *flashMidi* component by

---

<sup>84</sup> Personal website of Alexis Isaac, <http://www.alexisisaac.net/products/flashMidi/>. Currently, there is only a functional version of *flashMidi* for PC, but the Macintosh version is in development and should be released by late Summer 2006.





**Figure 2: N-Tier Software Design**

referencing it so that each module can use MIDI without having to include all of the *flashMidi* source code.

Using the Internet and careful programming, *Mobius* will be cross platform. Every major operating system supports at least one web browser. All the major web browsers follow a set of standards for presenting HTML, and support the *Flash* plug-in. By following HTML standards and using Flash programs, *Mobius* can run on most major operating systems. In order to guarantee that the software works the same on every machine, *Mobius* will include a routine to check the version of the web browser to make sure it offers the needed features. If the browser is obsolete, *Mobius* will notify the user they need to upgrade, and not let the user log in until they use a browser with the required features.

Integrated security is another positive aspect to using the Internet to deliver *Mobius*. If needed, *Mobius* can take advantage of 128-bit data encryption now in use on the web. Data encryption is a way to make the communication between the web browser and the web server completely private, and encryption technology is built right into most web browsers, including *Internet Explorer*, *FireFox*, and *Safari*. Encryption is a necessary feature for *Mobius* since the students need to enter a username and password to get into their account, and the system will store personal information like grades.

Cheating is an important logistical and security concern in using the web for CAI delivery. The *Mobius* software on the web server will support many users simultaneously, and keep track of their respective identities. The software will require a login from the users so that it can pull up their account from a web browser at any location. This will not prevent students from accessing each other's accounts, so a

mechanism should be built into *Mobius* that allows scores collected at certain computers, such as a lab, to count for test or homework grades. The students can take tests in a supervised area for grades, but work at home for practice and homework. The system will save data from exercises performed both inside and outside the lab for research purposes. This mechanism will assure that students can perform the exercises in a controlled environment and do not cheat, while still allowing them the freedom to work from home.

### **Mobius: The Base System**

*Mobius*' base system must manage students, teachers, and administrators, as well as classes, modules, and all data related to student interaction. It will allow administrators to install new modules, and manage teachers and classes. Teachers will be able to login to manage their classes, view grades, and assign homework and tests. Students who log into the system will be presented with exercises that have been assigned to them and will be able to see a report on their progress through the exercises. The system will also include a report generation tool called a data driller that allows researchers to look at different reports and views of how their students use the software.

The administrator account will be a special account that has special powers and special limitations. This account can control any other account that it creates, so student and teacher records in the database will be marked with the unique identification number of the administrator who created them. Assigning users to an administrator allows the system to support multiple schools, since each school or location is assigned an administrator. The administrator account will be limited in that it cannot also be a

teacher account. If the system administrator is also a teacher in real life, he must create a teacher account and log in to the software as that teacher to access his teacher controls.

The administrator functions and teacher functions are kept separate on purpose. The teacher controls will be designed to make managing students and homework easy, while the administrator controls will be designed to make managing teachers and classes easy.

When a teacher logs into the system, he or she will be presented with a tool called the class manager to manage classes, homework and view grades. The class manager gives the instructor the ability to create new students, group students into classes, and assign homework to those classes. The class manager allows the teacher to select any module installed on the system and configure it for the class that will use it. Figure 3 shows how the system allows them to create and manage their classes for a semester. Adding a new class is as simple as entering a name. Classes are merely a way of grouping exercises and students into logical groups.

Figure 4 shows the options a teacher has for configuring a class. He or she can view and add students in his/her class, and setup the modules used for the class, like interval or chord spelling. The instructor can give and edit assignments and view reports on how the class and individual students are progressing. The reports are provided by the data driller tool, which is described more below.

Figures 5 and 6 show how students are entered into the system and managed. Underneath the user intuitive user interface, new records are created that allow new students to use the modules and also allow information to be recorded and associated with each student. The data driller tool can be used to generate reports on the students' activity. The system can be configured to collect demographic information like age,

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Class\_mgr1.htm - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Print Mail Stop

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Class\_mgr1.htm

# Mobius

Expanding the Horizons of Music Learning

- My Account
- My Classes
- Reports
- Configuration
- Log Out

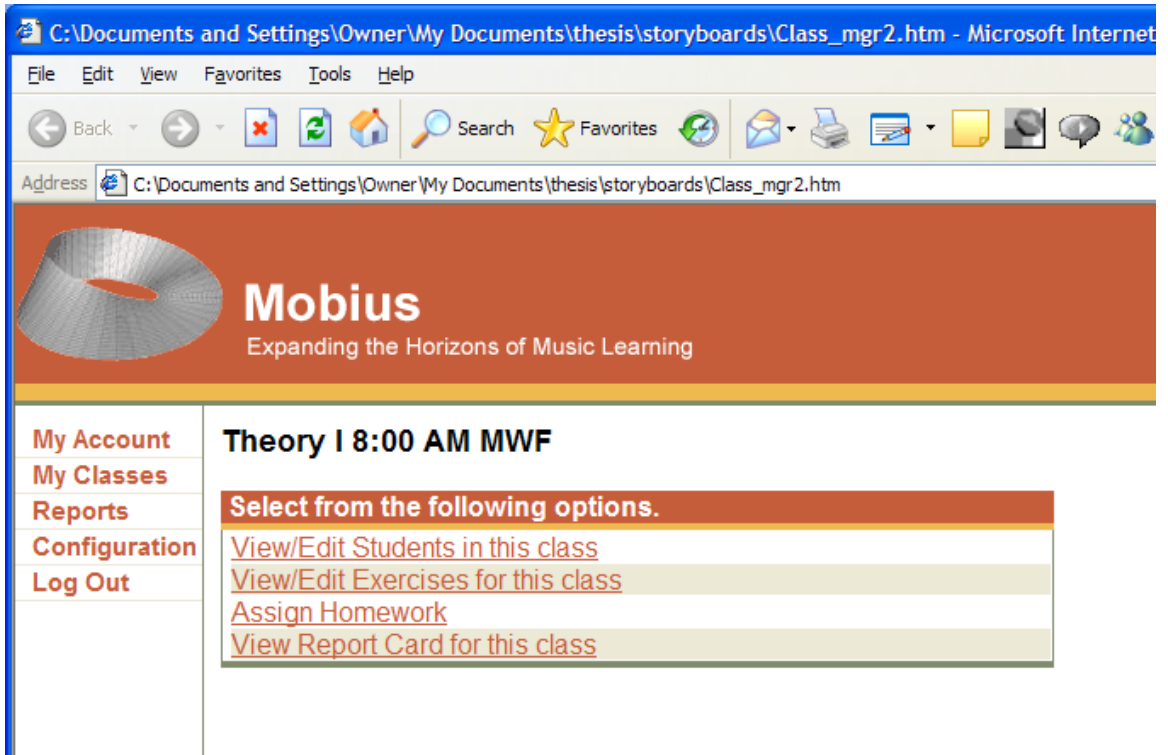
## My Classes

Your students are grouped into the following classes.

- [Theory I 8:00 AM MWF](#)
- [Theory I 10:00 AM MWF](#)
- [Ear Training II 9:30 AM TH](#)
- [Ear Training III 11:00 AM TH](#)

[Add a New Class](#)

**Figure 3: Class listing (instructor view)**



**Figure 4: Class controls (instructor view)**

The screenshot shows a Microsoft Internet Explorer browser window. The address bar displays the local file path: C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Class\_mgr3.htm. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains icons for Back, Forward, Stop, Refresh, Home, Search, Favorites, and other utilities. The page header features a Mobius strip logo and the text "Mobius Expanding the Horizons of Music Learning".

On the left side, there is a navigation menu with the following items: My Account, My Classes, Reports, Configuration, and Log Out. The main content area is titled "Students in Theory I 8:00 AM MWF" and contains a list of student names, each on a separate line with a light beige background:

- Casey, Jim
- Chillingworth, Roger
- Dimmesdale, Arthur
- Graves, Muley
- Joad, Tom
- Prynne, Hester
- Wainwright, Agnes
- Wilson, Ivy
- Wilson, John

At the bottom right of the list, there is a link: (Add a New Student)

Figure 5: Student listing by class





gender, and instrument/voice. This information can be mined in conjunction with other variables like test scores or time taken to complete exercises in order to create reports and statistical findings for publication.

Figure 7 shows how the instructor can give assignments. A list of assignments already given is shown in the center panel, followed by a panel that allows new assignments to be created. The system allows the instructor to name the exercise, set start dates and due dates, and choose which module to use. Figure 8 shows the second screen of the assignment creation process. The instructor will configure the module so that it meets his needs. In this example, the instructor can change the interval drill module configuration so that it focuses on various intervals, accepts partial credit, uses various clefs, etc.

Teachers will be able to access the system data, view reports, and generate new reports using a tool called the data driller. The data driller tool is a significant component in the *Mobius* software because it allows access to captured data. Teachers can use the reports to assess how their students are performing, or conduct research. The data driller will allow teachers to create new reports based on criteria he or she chooses, and then save the criteria for that report so it can be run again later. Reports can be designed to show data from a certain time period, such as the Fall 2005 semester, or the current week's worth of data. The data driller will provide teachers with a flexible research tool, and will hopefully show them how they can use CAI to perform research.


The reports generated by the data driller will be useful to teachers and administrators alike. From a teaching standpoint, reports can be run to see in which subjects students are succeeding or having difficulty. The teacher will be able to easily

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Class\_mgr4.htm - Microsoft In

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Class\_mgr4.htm



# Mobius

Expanding the Horizons of Music Learning

- My Account
- My Classes
- Reports
- Configuration
- Log Out

## Assignments for I 8:00 AM MWF

### Current Assignments

Exercise	Start Date	Due Date	
Interval Drill II	8/15/2008 8:00 AM	11/15/2008 5:00 PM	<a href="#">Edit</a>
Melodic Dictation I	8/15/2008 8:00 AM	9/15/2008 5:00 PM	<a href="#">Edit</a>
Melodic Dictation II	8/15/2008 8:00 AM	10/15/2008 5:00 PM	<a href="#">Edit</a>
Harmonic Dictation I	10/15/2008 8:00 AM	11/15/2008 5:00 PM	<a href="#">Edit</a>

### Add New Assignment

Assignment Name:

Start Date:      
 Due Date:

Exercise:

Figure 7: Existing homework and add new homework dialog boxes

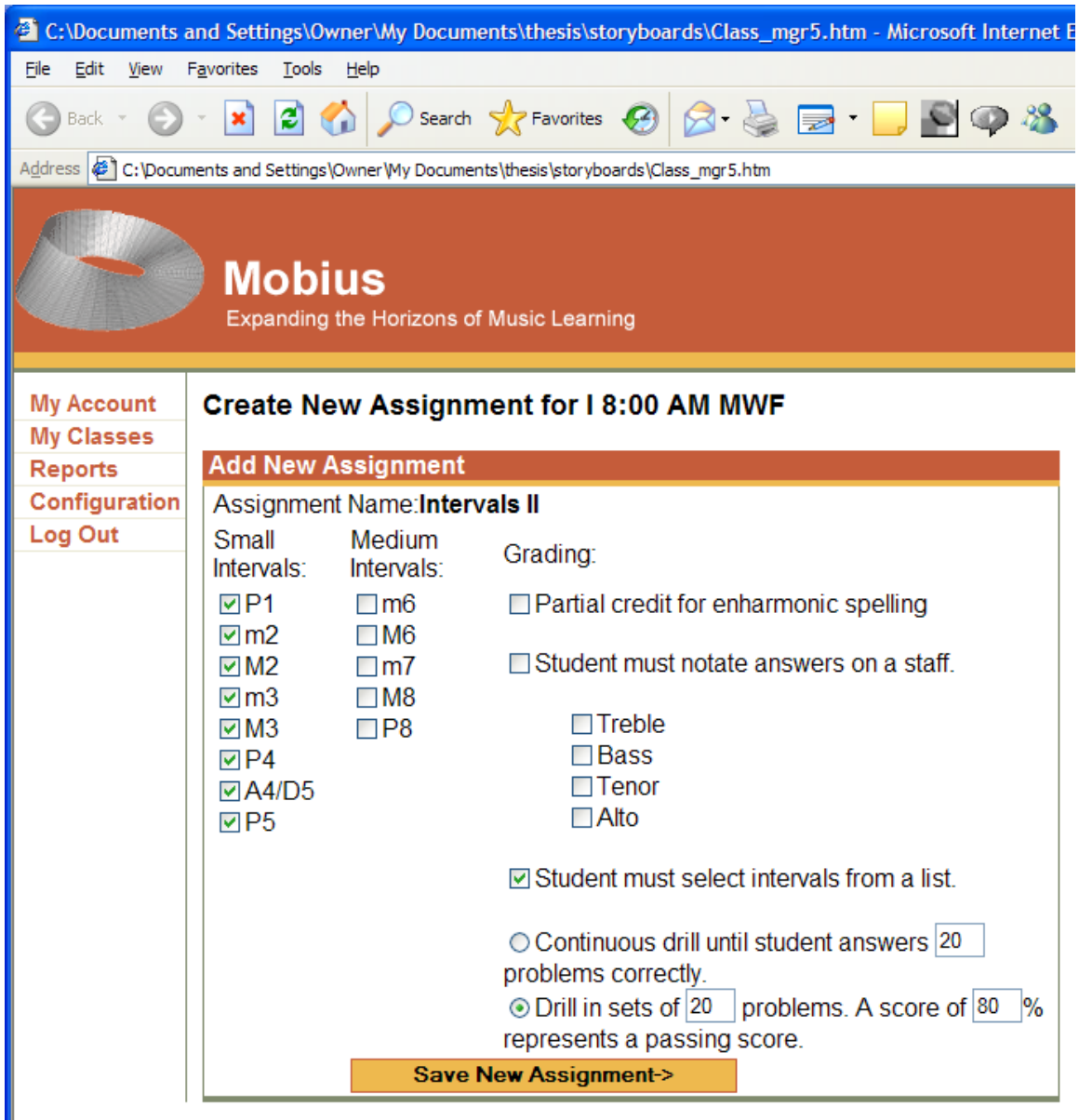


Figure 8: Configure a homework assignment dialog

recognize students who need personal attention, and can tell if there is a particular set of exercises in which all the students have difficulty. The instructor can verify that the student has put appropriate amounts of time into understanding the material and intercede if no new progress is being made. The system administrator can use reports to detect technical problems in the system by viewing how students access the system, how often, and for how long. For instance, a report showing that a student logged into the system frequently in a short period could indicate that the student is having technical difficulties and needs assistance. Researchers can use the data driller to access the system data to investigate how students learn using the system. The data driller can produce reports comparing various variables, including age, gender, or voice/instrument versus test scores or the amount of time required to complete an exercise.

To create a report, a teacher will be able to log in to the system and select the “Reports” option from the menu. Figure 9 shows an example of how the start screen of the data driller tool will look. The screen lists existing reports that were created earlier. To create a new report, the teacher must click the “Create New Report” link. Figure 10 shows the first page of the new report generator. Teachers will be able to access records for this semester, all semesters, or choose which semesters from which the data is pulled. Figure 11 shows the screen displayed when the instructor chooses to select which semesters are mined for data. When the instructor selects the set of data he or she wants to view, clicking the “Next” button will allow them to select which exercises will be included in the report. Figure 12 shows the list of exercises for the semesters chosen. The instructor can choose to select any of the exercises, but should be aware that different exercises may be configured differently. Researchers should be take into account the

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller1.htm - Microsoft Intern

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller1.htm

**Mobius**  
Expanding the Horizons of Music Learning

**My Account**  
**My Classes**  
**Reports**  
**Configuration**  
**Log Out**

**Data Driller**

**Existing Reports**

- [Total/Avg Minutes by Class](#)
- [Avg Scores by Exercise](#)
- [Avg Scores by Age](#)
- [Avg Score by Instrument/Voice](#)
- [Avg Time by Instrument/Voice](#)
- [Total Time by Gender](#)

[\(Create New Report\)](#)

**Figure 9: List of existing reports (Instructor View)**



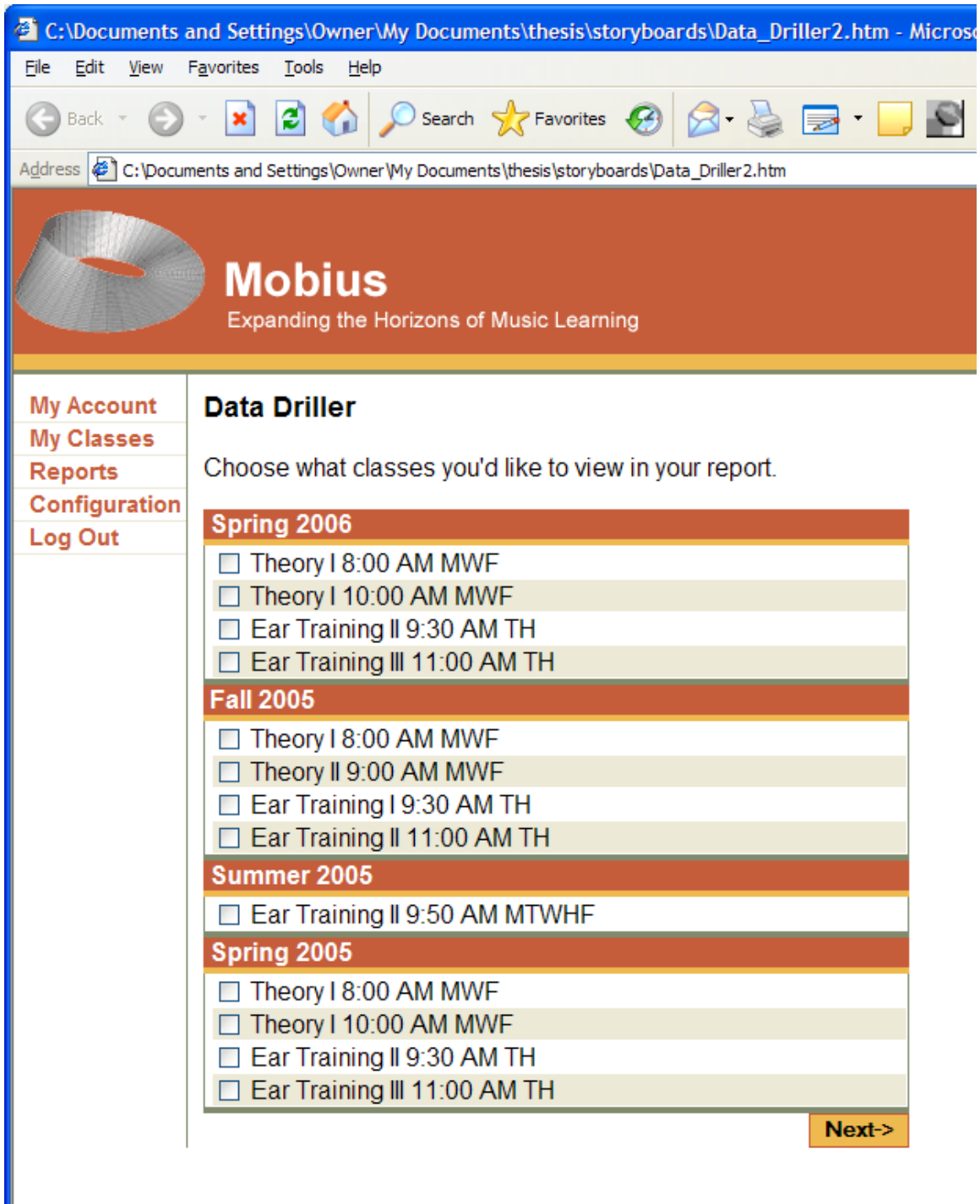


Figure 11: Select specific data sets for a report

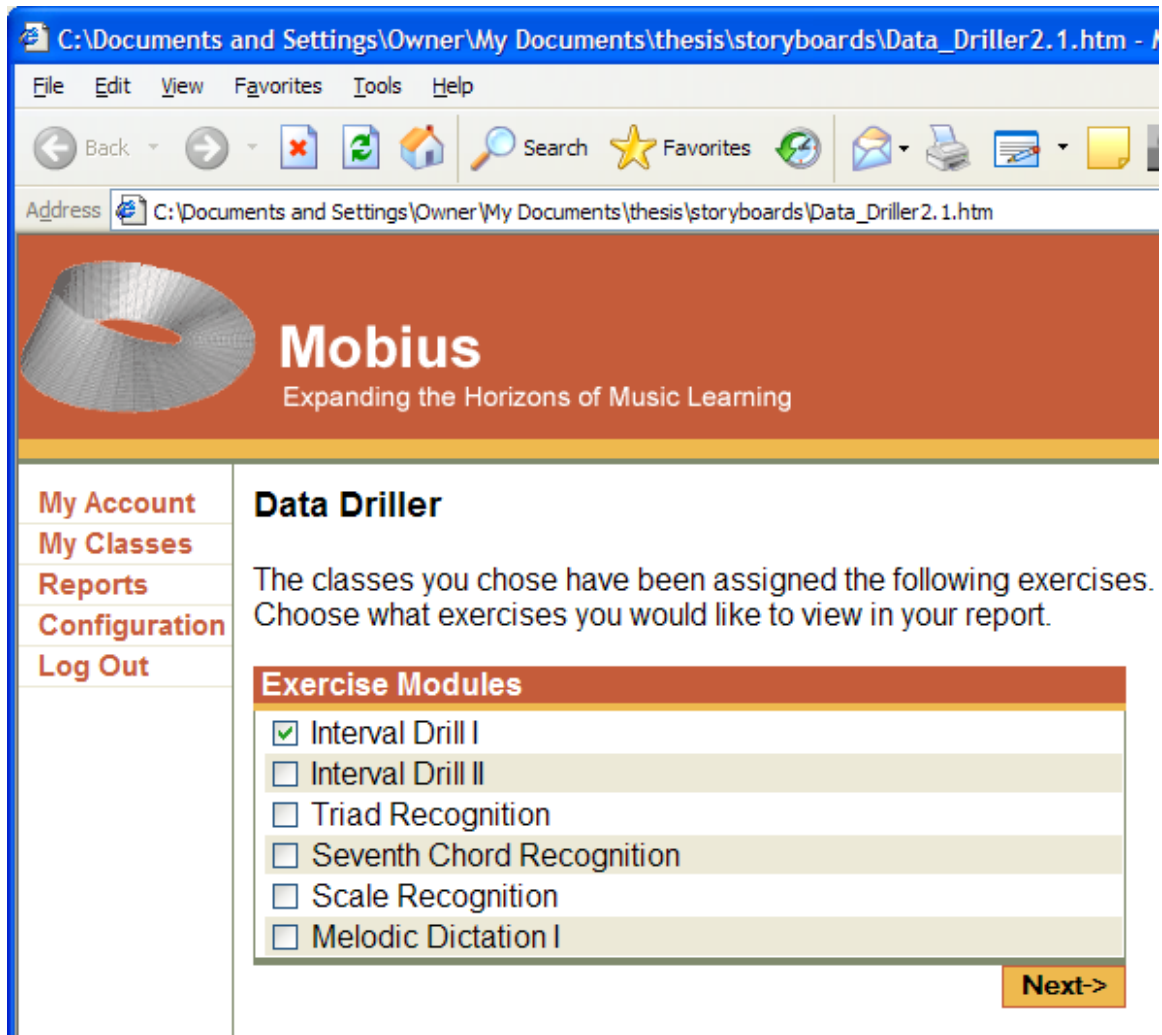


Figure 12: Choose the exercises reported



potential confounding effects of differently configured modules and only generate reports based on like data.

Figure 13 shows the next step of the report generation process. The instructor can choose which data to include in his or her report. They can select multiple fields by holding the select key down as they click fields with the mouse. The fields available for the report will vary based on the modules reported in the data. Figure 13 shows a sample of some of the generic kinds of data available for the report.

The final step of the report generation process, shown in Figure 14, is to decide how to treat the columns selected on the previous page. Numeric columns can be averaged or summed, for aggregate data. Comparison operators such as “Like” or “Equals” can be used for both text and number fields to filter data. A “Group By” filter can be used to organize the data.

Figure 15 shows the report generated after the organization and refinement page. Shown are the columns, “Class,” “Name,” and “Time Spent.” The report is grouped by “Class,” it is ordered by “Name” alphabetically, and “Time Spent” is summed and averaged. The report shown in Figure 15 allows the teacher to see how much time their students have worked on the system, and the average amount of time per session.

If the report is to the teacher’s liking, he or she can save it to the existing reports collection so they can view it again later with the same data. Figure 16 shows the “Save Report” dialog. Once a report is saved, it can also be modified to view other data. The process is similar to that of creating a new report.

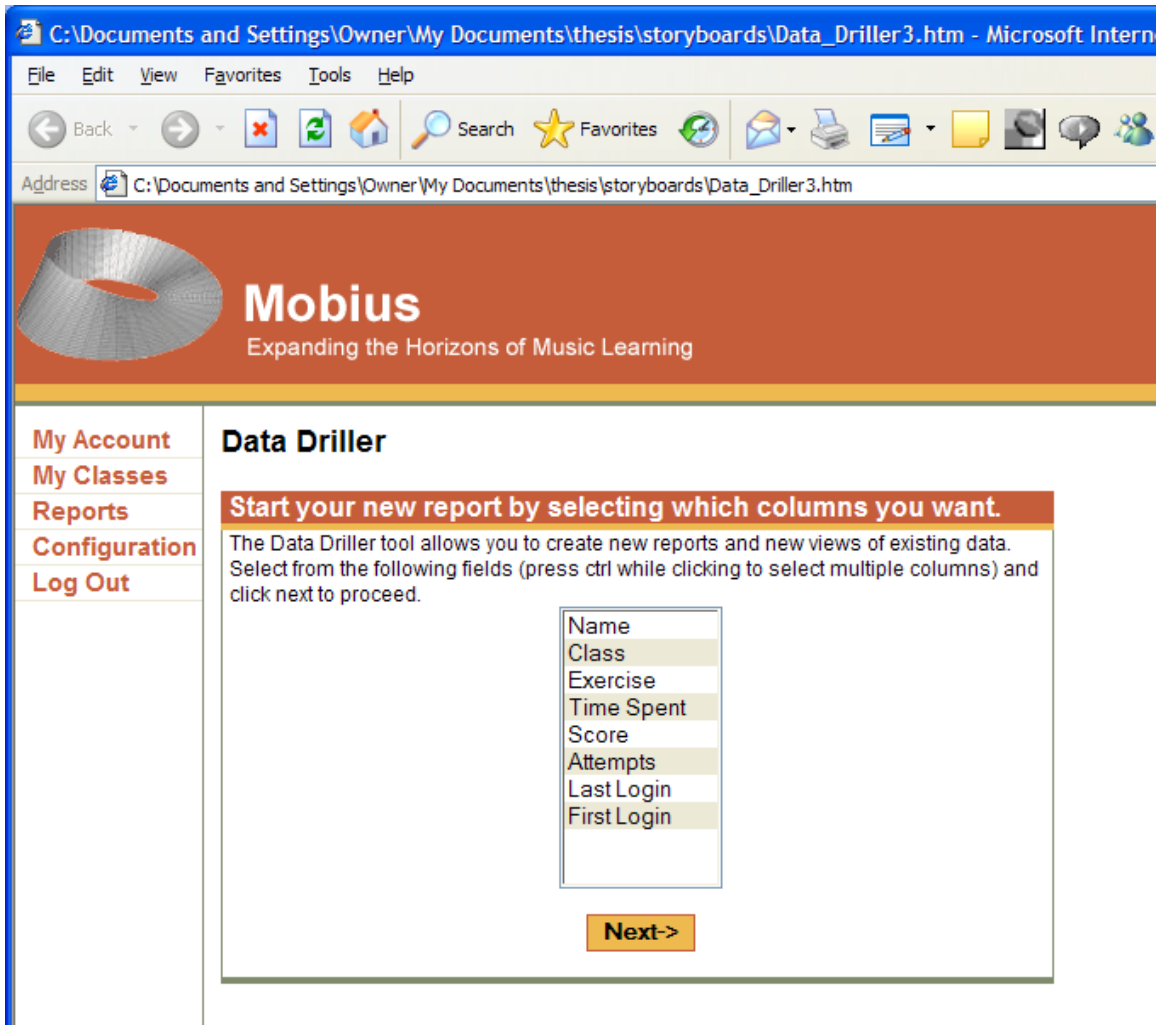



Figure 13: Select columns for the report

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller4.htm - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller4.htm



**Mobius**  
Expanding the Horizons of Music Learning

**My Account**  
**My Classes**  
**Reports**  
**Configuration**  
**Log Out**

**Data Driller**

Organize and refine the data you selected.

Column	GroupView Type	Filter
Name <input type="checkbox"/>	Order By Ascending	None
Class <input checked="" type="checkbox"/>	Normal	None
Time Spent <input type="checkbox"/>	Average (Numbers only)	None
Time Spent <input type="checkbox"/>	Sum (Numbers only)	None

**Next->**

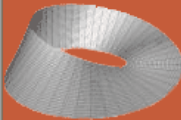
**Figure 14: Organize and aggregate data**

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller5.htm - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Stop

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller5.htm



**Mobius**  
Expanding the Horizons of Music Learning

**My Account**  
**My Classes**  
**Reports**  
**Configuration**  
**Log Out**

**Data Driller**

**Report Generated**

Class (Group By)	Name	Time Spent (Average)	Time Spent (Sum)
Ear Training I 9:30 AM TH	<a href="#">Casey, Jim</a>	12 min	55 min
	<a href="#">Chillingworth, Roger</a>	33 min	1011 min
	<a href="#">Dimmesdale, Arthur</a>	25 min	406 min
	<a href="#">Graves, Muley</a>	15 min	59 min
	<a href="#">Joad, Tom</a>	28 min	251 min
	<a href="#">Prynne, Hester</a>	9 min	9 min
	<a href="#">Wainwright, Agnes</a>	51 min	366 min
	<a href="#">Wilson, Ivy</a>	22 min	125 min
	<a href="#">Wilson, John</a>	16 min	106 min

[\(Save This Report\)](#)

**Figure 15: Report results**

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller6.htm - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Data\_Driller6.htm

# Mobius

Expanding the Horizons of Music Learning

- My Account
- My Classes
- Reports
- Configuration
- Log Out

## Data Driller

Name your new report something easy to understand later. Don't use a name that is already listed in your reports below.

Report Name:  **Next->**

### Existing Reports

- [Total/Avg Minutes by Class](#)
- [Avg Scores by Exercise](#)
- [Avg Scores by Age](#)
- [Avg Score by Instrument/Voice](#)
- [Avg Time by Instrument/Voice](#)
- [Total Time by Gender](#)

Figure 16: List of saved reports

The various modules that deliver course instruction, drills, and exercises will be administered from a central control panel called the module manager. The module manager will provide an easy-to-use interface for the administrator to install and manage modules in the *Mobius* software, and teachers to configure and use those modules. Part of the module management system is available through the class manager, which allows a teacher to enter new students, group them into classes, give assignments, and build reports on student progress. The other part of the module management system would only be available to a system administrator. The administrator module manager can perform system-wide changes, and any changes made here would impact anyone who uses the system, not just the students in one particular class. Figure 17 shows an example of how the system lists installed modules, offers options for configuration, deletion of modules, and installation of new modules.


When the system administrator wants to install new instruction modules for teachers to use in their classes he or she can click the “Install New Modules” button and be taken to the screen shown in Figure 18. The screenshot shown in Figure 18 shows modules that are available for installation, whether from files that already exist on the server, or modules that exist in a central repository in another location. Modules that exist on the server will be installed and the system administrator shown a configuration screen for the new module. Configuration screens for each module will be different, and may even allow configuration at an instructor level. After the module is installed and configured, the teachers who use the system will have access to the new module for use in their classes.

C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Mod\_mgr1.htm - Microsoft Internet

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address C:\Documents and Settings\Owner\My Documents\thesis\storyboards\Mod\_mgr1.htm



# Mobius

Expanding the Horizons of Music Learning

- My Account
- My Classes
- Reports
- Configuration
- Log Out

## Available Exercises

The following exercise modules are installed on this system.

Interval Drill 1	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Interval Drill 1 Modified	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Interval Drill 2	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Melodic Dictation 1	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Melodic Dictation 2	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Tuning Drill	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Harmonic Dictation 1	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>
Harmonic Dictation 2	<a href="#">Edit Configuration</a>	<a href="#">Uninstall</a>

[Install New Modules->](#)

**Figure 17: List of modules installed**

**My Account**  
**My Classes**  
**Reports**  
**Configuration**  
**Log Out**

### New Modules Available

The following exercise modules are available to be installed on this system.

Interval Drill 1	Author: Davie Jones	<a href="#">Install</a>
Description: This module drills the student on written intervals in all clefs. It uses the mastery model.		
Interval Drill 1 with interactive help	Author: Davie Jones	<a href="#">Install</a>
Description: This module drills the student on written intervals in all clefs and coaches them through the process. It does not record scores since the answers are coached, but it is similar in format to Interval Drill I and can teach students how it works.		
Aural Interval Drill	Author: Dr. John Silver	<a href="#">Install</a>
Description: This module drills the student on aural intervals.		
Chord Speller	Author: Dr. Livesey	<a href="#">Install</a>
Description: This module drills the student on written chords in all clefs. It uses the mastery model.		
Melody Dictation Level I	Author: James Hawkins	<a href="#">Install</a>
Description: Leads the student through melodic dictation exercises.		
Melody Dictation Level II	Author: James Hawkins	<a href="#">Install</a>
Description: Leads the student through melodic dictation exercises.		

You can upload new modules to the system in Gzip and Zip files. See the specifications detailed in the [Developers Guide](#).

[Browse](#)

[Upload a New Module->](#)

**Figure 18: List of modules available for download from the central repository**



## **The Software Team**

The *Mobius* software team will be comprised of volunteer programmers, designers, and content specialists. These volunteers will be recruited by posting advertisements on bulletin boards for music technology and music software, and email listservs like that run by ATMI. Volunteers who join the project team will be able to download the source code and volunteer for pending tasks. The team's purpose is not only to build the software, but also to promote adoption of the software, and build a stable body of knowledge for open source CAI.

The *Mobius* team consists of a project manager, a software design team, coders, quality testers, and technical writers. Often, people will do several of these tasks, like a programmer writing documentation on the code he wrote. The project manager's role is to coordinate all the other team members and to assure that the project progresses on schedule. The project manager also has executive authority on any changes to the initial design, since changes to the design after coding has started usually costs a lot time. It is the responsibility of the project manager to resolve any conflicts within the team, and to make sure that each portion of the project is completed.

The design team will receive a list of requirements from the project manager and create a technical design for the software. The project manager uses the design to assemble a list of tasks for the project. Some tasks cannot be completed until some other portion of the project is completed, so the project manager must assemble the task list based on the sequencing of events. Once the order of tasks is determined, the project manager can communicate the project needs to the coding team.

There are software packages and websites available to automate the project management process, such as *TUTOS*, *WebCollab*, and *dotProject*, and Sourceforge.net. Project management software lets a project manager run a project through a website. The project manager can add people to a project, assign tasks to them, keep a library of files, and track the completed tasks. The programs allow the manager to build a virtual project plan and keep track of what tasks are completed and how the project is progressing. Programmers can log in to see their assigned tasks, make comments about their progress. Sourceforge.net provides these services for free, and allows the public to see that status of a project and download files that the administrator decides are public.

Project management software allows people to communicate and store information in a standard location, but there is a different tool for sharing and managing program source code between different team members. Source code control software (also known as version control) like *CVS*, *CVSNT*, and *MS SourceSafe* is specialized software that saves all current and previous versions of a program's source code in a central location. This makes it practically impossible to lose work once it is saved to the repository. If a programmer wants to make changes to existing code in the repository, he must check it out; similar to how a library operates. Other users cannot checkout the same chunk of code until it has been checked back in. The version control software notes who checked out the code, and allows the programmer to save his or her work periodically. Once all changes are finished and approved, the programmer checks the software back in to the repository, and the source code control software will allow others to check it out. Source code control software allows a programmer to get all the latest approved versions of code to assure that every piece of code is functional and complete.

There are many different coding styles, even for programmers who use the same language, so it is important for the project manager to declare coding conventions before the project begins. Coding conventions are a list of rules that the programmers must follow as they develop code for the project. Rules include how to name variables, how to structure code so that it is consistent with other code in the project, and how the code is organized structurally. *Mobius* will use the Gforge.net PHP coding conventions.<sup>85</sup>

### **Using Mobius**

Since *Mobius* is web based, it must be installed on a web server before people can use it. For those who have access to a web server on which to install *Mobius*, installation begins with downloading the software from the project web page. The project website will provide installation instructions and tools. For those who do not have access to a web server on which to install *Mobius*, there will be two options. The *Mobius* project website will include a demonstration version of the software so people can log in and see how it works as teachers and students. People who wish to use the drills can log in as a student and practice for free. However, because the site is for demonstration, the default settings and data will be restored on a regular basis so this option is not practical for classroom use. The second option is to sign up to use someone else's installation of the software. Since *Mobius* is delivered through the web, another music department or individual who has the software installed can grant access to people in other locations. For example, a large regional university could install and configure the software to provide memberships for students and teachers at smaller local colleges for free, or for a

---

<sup>85</sup> Gforge, CDE. "PHP Coding Standards." <http://gforge.org/docman/view.php/1/2/file2.html>.

small fee. The fee would help to pay for the hosting costs incurred for the software, like bandwidth, technical support, or paying a specialist to maintain the server. By sharing the software, larger schools can help defray their own costs for hardware and manpower, while letting smaller schools use the software. The regional university could use one installation of *Mobius* as an outreach program to make sure that students at smaller schools are receiving appropriate pedagogy.

Programmers who want to work on the software, or who are just curious how it works, can download it for free from the *Mobius* project website. Once *Mobius* is downloaded and installed, the next step is for the programmer to familiarize himself with the software, code, and documentation. The programmer can use the configuration tools like the module manager and class manager to see how the software works, set up exercises to see how they behave, and use the data driller to see what information is captured by the system. Once the new programmer has a good grasp on what the system does, he can examine the source code to see how certain aspects of the program work. The programmer can make modifications to the *Mobius* software code to experiment and learn. If he or she makes a mistake and cause the software to stop working, they only need to reinstall it.

Programmers who develop new modules for *Mobius* can submit them to the project website to be added to the module repository. Others can download and use modules from the repository for their own purposes, including modification and redistribution. Teachers and users can add features and improvements that they want to use, and can share them with others. By enabling researchers to modify modules and store new data, research opportunities open up. Data collected through the *Mobius*

system can be used to determine what methods work best for web-based CAI delivery, and those methods can be implemented for all to use.

A typical research scenario might ask a question such as; is there a relationship between the time it takes a student to identify an aural interval and how accurately choose the correct answer? The hypothesis could be that students who choose their answers quickly are likely to have higher score, since they clearly know the answer. To research this hypothesis using *Mobius*, the research can download and modify the standard interval recognition module. The module already presents and scores interval drills, so the only modification the researcher needs to make is to have it store the time it takes the student to answer each question. Once the modified module is tested, it can be uploaded and activated into *Mobius*. The researcher then must create a “homework” assignment for students to access. The assignment can be integrated into an existing class, or test subjects can be added to the system and given the assignment. At the end of the research cycle, the researcher can use the data-drill to extract the raw data he needs for analysis, or generate a report from the system.

*Mobius* can enable long-term research projects that other software projects cannot address. *Mobius* can track students for their entire time using the software. If the students use *Mobius* over the course of several years, a module that records the time for students to choose an interval could detect changes in student ability over weeks, months and semesters. By collecting information over the long-term, *Mobius* can help understand trends in student performance.

## Chapter 5: Conclusion

Music CAI's roots in research have provided a wealth of knowledge and good pedagogy for today's students. Since the days of early mainframes, computers have been used to deliver musical instruction and study how students learn. The data collected in those studies provided important knowledge for teachers, and allowed programmers to develop software to deliver better music instruction. PCs allowed programmers to distribute their music CAI software to many more people than was possible on mainframes, and MIDI allowed PCs to interface directly with instruments and banks of sounds. CAI improved rapidly in quality and capability, but the emphasis on research began to wane.

Research became less of a focus in CAI development in the early to mid 1990s for several reasons. First, closed source software distribution practices contributed to the lack of research by preventing programmers from creating experiments based on existing software. Second, CAI software came to be seen as more of a commodity than a research opportunity. Third, the culture of the music school does not support software development as research, only as publication. Research became less of a focus as the music community focused more on the CAI product and the results that is promised in the classroom.

Open source software can help to solve some of the problems of conducting research in music CAI. It can provide CAI programmers a way of sharing technical information and source code. Since no one can own the rights to open source software, researchers and teachers can use it any way they wish, and the focus can be on new ways to deliver instruction. Open source software development practices allow for a whole

team of programmers, designers, and contributors to work together, spreading the workload among many people so that the limitations of one programmer will not limit the future of the software.

By using open source ideas, *Mobius* can become a powerful research and learning tool. Researchers will be able to use and modify the program so they do not have to write their own system from scratch. Other researchers will be able to use a similarly configured *Mobius* as the basis for their experiments, incorporating the lessons learned by others. *Mobius* will also allow people to build their own modules and share them with others. It will enable a new generation of researchers to conceive and execute their music CAI experiments with effective and tested tools. CAI programmers can build a body of knowledge for CAI software, including effective and ineffective teaching methods, effective and ineffective programming methods, and a stable body of public source code that grows and improves over time. The body of knowledge created in *Mobius* using open source development techniques will spark further research and produce new teaching methods and improved pedagogy.

## **Bibliography**



- “An Advisory for Music Faculty and Administrators: NASM Standards – Technology” in *NASM Handbook 2003 – 2004*, 84.
- Alpert, D. and D. Bitzer. “Advances in Computer-Based Education” in *Science New Series*, vol. 167, no. 3925 (1970): 1582-1590.
- Cheney, Lynne V.. *Tyrannical Machines: A Report on Educational Practices Gone Wrong and Our Best Hopes for Setting Them Right*. Washington D.C.: National Endowment for the Humanities, 1990.
- Deihl, Ned. “Computer-Assisted Instruction: Potential for Instrumental Music Education” *Council for Research in Music Education bulletin* vol. 15 (1969): 1-7.
- Tiemann, Michael. “Future of Cygnus Solutions: An Entrepreneur's Account” in *Open Sources: Voices from the Open Source Revolution*, ed. Chris DiBona, Sam Ockman, and Mark Stone. Sebastopol, CA: O’Reilly, 1999.
- Gross, Dorothy, and Roger E. Foltz. “Ideas on Implementation and Evaluation of a Music CAI Project” *College Music Symposium*, 21 no. 2 (1981): 22-26.
- Hofstetter, Fred. “Applications of the GUIDO System to Aural Skills Research 1975-80” *College Music Symposium*, 21 no. 2 (1981): 46-53.
- Killam, Rosemary. “An Effective Computer-Assisted Learning Environment for Aural Skill Development,” *Music Theory Spectrum* 6 (1984): 53.
- Killam, Rosemary N., Philip Baczewski, Antoinette Corbet, Paul Edward Dworak, Jana Kubitza, Michael Morgan and Lawrence Woodruff. “Research Applications in CAI” *College Music Symposium* 21 no. 2 (1981): 43.

- Kuhn, Wolfgang. "Computer-Assisted Instruction in Music: Drill and Practice in Dictation" *College Music Symposium* 14 (1974): 89- 101.
- McConnell, Steve. *After the Gold Rush: Creating a True Profession of Software Engineering*. Redmond, WA: Microsoft Press, 1999.
- McConnell, Steve. *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last*. Redmond, WA: Microsoft Press, 1998.
- McGee, Deron. "Aural Skill, Pedagogy, and Computer-Assisted Instruction: Past, Present, and Future," *Journal of Music Theory Pedagogy* 14 (2000): 115-134.
- Peters, G. David. "Music Software and Emerging Technology" *Music Educators Journal* 79 no. 3 (1992): 22-27.
- Schaffer, John. "Intelligent Tutoring Systems: New Realms in CAI" in *Music Theory Spectrum* 12 no. 2 (1990): 224-235.
- Shrader, David L.. "Microcomputer-Based Teaching" in *College Music Symposium* 21 no. 2 (1981): 27-36.
- Skinner, Robert. "Music Software." *Notes* 45 no. 3. (1989): 537-541.
- Stallman, Richard. *Free Software, Free Society*. Boston: GNU Press, 2002.
- Taylor, Timothy. *Strange Sounds: Music, Technology, and Culture*. New York: Routledge, 2001.
- Theberge, Paul. *Any Sound You Can Imagine: Making Music/Consuming Technology*. Hanover, NH: Wesleyan University Press of New England, 1997.
- Webster, Peter. "Historical Perspectives on Technology and Music" *Music Educators Journal* 89 no. 1 (2002): 38-45.

Wittlich, Gary. "Computer Applications: Pedagogy." *Music Theory Spectrum* 11 no. 1,  
Special Issue: The Society for Music Theory: The First Decade (1989): 60-65.

Wittlich, Gary, John Schaffer and Larry Babb. *Microcomputers and Music*. Englewood  
Cliffs, NJ: Prentice Hall Professional Technical Reference, 1986.

## Appendices

## Appendix A: Dissertation Abstracts Data

**Table A.1: Number of dissertations per year**

Year	Number of Studies
1969	1
1970	0
1971	1
1972	0
1973	0
1974	1
1975	2
1976	0
1977	0
1978	1
1979	1
1980	2
1981	4
1982	2
1983	2
1984	4
1985	3
1986	4
1987	2
1988	6
1989	4
1990	3
1991	2
1992	6
1993	2
1994	5
1995	5
1996	2
1997	2
1998	3
1999	2
2000	3
2001	4
2002	3
2003	2
2004	2
2005	2
<b>Total:</b>	<b>88</b>

## Search results

Filtered Dissertation Abstracts Online results for:  
kw: computer and kw: music and kw: instruction.  
Records found: 88

Missing numbers indicate results that were omitted from the original search. The list of dissertations was filtered by reading the abstract to ascertain that the dissertation was relevant to the topic. The results shown below are the computer output for the search, including the capitalized sections.

1. Music tools: Rhythmic dictation educational software  
Author: Hubbard, Bruce Degree: M.A. Institution: California State University, Dominguez Hills 0582 Year: 2005
2. Computer-assisted instruction for music theory education: Rhythm in music  
Author: Chew, Deborah Yvonne Degree: M.A. Institution: California State University, Dominguez Hills 0582 Year: 2005
3. Computer-assisted music instruction as supplemental sight-singing instruction in the high school choir  
Author: Ewers, Marla Sue Degree: Ed.D. Institution: University of Illinois at Urbana-Champaign 0090 Year: 2004
5. Quartal harmony for guitar: A CAI field test  
Author: Spores, Craig Degree: M.A. Institution: California State University, Dominguez Hills 0582 Year: 2004
9. Will a music and spatial-temporal math program enhance test scores? An analysis of second-grade students' mathematics performance on the Stanford-9 Test and the Capistrano Unified School District CORE Level Test  
Author: Rafferty, Kevin Neal Degree: Ed.D. Institution: University of Southern California 0208 Year: 2003
10. The comparative effects of computer-mediated interactive instruction and traditional instruction on music achievement in guitar performance  
Author: Green, Bryan Richard Degree: Ph.D. Institution: The University of British Columbia (Canada) 2500 Year: 2003
16. The effects of computer-based instruction on achievement of four, five and six-year-old children in the Yamaha Music Education System Primary One Course  
Author: Bailey, Darrell Lee Degree: Ed.D. Institution: University of Illinois at Urbana-Champaign 0090 Year: 1989

19. The effectiveness of computer-assisted instruction on the development of rhythm reading skills among middle school instrumental students  
Author: Smith, Kenneth Harold Degree: Ph.D. Institution: University of Illinois at Urbana-Champaign 0090 Year: 2002
20. The comparison of the effects of two computer-based music instructional programs in teaching piano note reading to adults through two different delivery systems  
Author: He, Hui-Chieh Judy Degree: Ph.D. Institution: University of Illinois at Urbana-Champaign 0090 Year: 1995
21. Teaching styles and faculty attitudes towards computer technology in teaching and learning at a college in Ontario  
Author: Lloyd, David George Degree: Ph.D. Institution: University of Toronto (Canada) 0779 Year: 2002
22. Reinventing music theory pedagogy: The development and use of a CAI program to guide students in the analysis of musical form  
Author: Sterling, Jennifer Elizabeth Degree: Ph.D. Institution: University of Maryland College Park 0117 Year: 2002
25. A comparative study of the effects of computer-based expository and discovery methods of instruction for fostering the aural recognition of musical concepts  
Author: Hopkins, Michael Thomas Degree: Ph.D. Institution: University of Michigan 0127 Year: 2001
26. The effects of computer-assisted instruction and cognitive style on sight playing among university group piano students  
Author: Hagen, Sara L. Degree: Ph.D. Institution: The Florida State University 0071 Year: 2001
27. The effect of time in computerized versus classroom instruction on the ability to correctly pronounce English words phonetically transcribed into the International Phonetic Alphabet  
Author: Dekaney, Elisa Macedo Degree: Ph.D. Institution: The Florida State University 0071 Year: 2001
30. Comparative learning methods of cognitive computer-based training with and without multimedia blending  
Author: Salinas, Fidel Michael, Jr. Degree: Ed.D. Institution: University of the Pacific 0173 Year: 2001
31. Formative research on the refinement of Web-based instructional design and development guidance systems for teaching music fundamentals at the pre-college level

Author: Chuang, Wen-Hao Degree: Ph.D. Institution: Indiana University 0093 Year: 2000

32. Computer-assisted instruction and sequencing within the studio: A focus on Halsey Stevens' "Sonata for Trumpet and Piano"

Author: Zifer, Timothy James Degree: D.M.A. Institution: The Louisiana State University and Agricultural and Mechanical College 0107 Year: 2000

33. A new approach to computer-assisted instruction in music theory for elementary and middle school children

Author: Bowyer, Donald William Degree: D.A. Institution: University of Northern Colorado 0161 Year: 2000

37. The evaluation of two self-instruction learning approaches assessing music knowledge and simple music keyboard performance skills

Author: Tomczak, Larry Mark Degree: Ed.D. Institution: University of Cincinnati 0045 Year: 1999

39. COMPUTER-ASSISTED INSTRUCTION IN EAR-TRAINING AND ITS INTEGRATION INTO UNDERGRADUATE MUSIC PROGRAMS DURING THE 1998-1999 ACADEMIC YEAR

Author: SPANGLER, DOUGLAS RAYMOND Degree: M.MUS. Institution: MICHIGAN STATE UNIVERSITY 0128 Year: 1999

45. THE EFFECTIVENESS OF COMPUTER-ASSISTED INSTRUCTION IN SELECTED SECONDARY SCHOOLS IN LOS ANGELES AND ORANGE COUNTIES OF SOUTHERN CALIFORNIA

Author: YUNE, JOSEPH TAE-JUNG Degree: M.M. Institution: UNIVERSITY OF SOUTHERN CALIFORNIA 0208 Year: 1998

49. ANALYSIS OF MUSICAL CREATIVITY IN MIDDLE SCHOOL STUDENTS THROUGH COMPOSITION USING COMPUTER-ASSISTED INSTRUCTION: A MULTIPLE CASE STUDY

Author: EMMONS, SCOTT EVERETT Degree: PH.D. Institution: THE UNIVERSITY OF ROCHESTER, EASTMAN SCHOOL OF MUSIC 0891 Year: 1998

50. AN EVALUATION OF THE EFFECTIVENESS OF A GROUP PIANO PROGRAM USING ELECTRONIC KEYBOARD AND COMPUTER TECHNOLOGY

Author: SHENDER, MARIE Degree: ED.D. Institution: COLUMBIA UNIVERSITY TEACHERS COLLEGE 0055 Year: 1998

54. THE EFFECTS OF COMPUTER-ASSISTED KEYBOARD INSTRUCTION ON METER DISCRIMINATION AND RHYTHM DISCRIMINATION OF GENERAL MUSIC EDUCATION STUDENTS IN THE ELEMENTARY SCHOOL (CAI)



Author: ARMS GILBERT, LINDA Degree: ED.D. Institution: TENNESSEE STATE UNIVERSITY 0840 Year: 1997

58. AN EXAMINATION OF THE EFFECT OF WRITING MELODIES, USING A COMPUTER-BASED SONG-WRITING PROGRAM, ON HIGH SCHOOL STUDENTS' INDIVIDUAL LEARNING OF SIGHT-SINGING SKILLS

Author: PRASSO, NINA MARLENE Degree: ED.D. Institution: COLUMBIA UNIVERSITY TEACHERS COLLEGE 0055 Year: 1997

62. THE DEVELOPMENT AND EVALUATION OF A COMPUTER-ASSISTED MUSIC INSTRUCTION PROGRAM AS AN AID TO SCORE STUDY FOR THE UNDERGRADUATE WIND BAND CONDUCTING STUDENT

Author: HUDSON, MARK EDWARD Degree: PH.D. Institution: UNIVERSITY OF FLORIDA 0070 Year: 1996

67. THE COMPARISON OF THE EFFECTS OF TWO COMPUTER-BASED MUSIC INSTRUCTIONAL PROGRAMS IN TEACHING PIANO NOTE READING TO ADULTS THROUGH TWO DIFFERENT DELIVERY SYSTEMS

Author: HE, HUI-CHIEH JUDY Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1995

70. An exploratory study to incorporate supplementary computer-assisted historical and theoretical studies into applied music instruction

Author: Kim, Sara Junghwa Degree: Ed.D. Institution: Columbia University Teachers College 0055 Year: 1996

72. The effectiveness of a computer-based courseware program for teaching jazz improvisation

Author: Fern, James Lloyd, Jr. Degree: D.M.A. Institution: University of Southern California 0208 Year: 1995

74. MUSIC-INSTRUCTIONAL STRATEGY-INTEGRATION: THE MODERATING EFFECT OF MUSIC IN THE DESIGN OF COMPUTER-BASED INSTRUCTION

Author: HARDY, RODOLPH DONALD Degree: PH.D. Institution: GEORGIA STATE UNIVERSITY 0079 Year: 1995

76. A META-ANALYSIS OF THE EFFECT OF COMPUTER-ASSISTED INSTRUCTION ON THE ACADEMIC ACHIEVEMENT OF STUDENTS IN GRADES 6 THROUGH 12: A COMPARISON OF URBAN, SUBURBAN, AND RURAL EDUCATIONAL SETTINGS (SIXTH-GRADE, TWELFTH-GRADE, URBAN EDUCATION, RURAL EDUCATION)

Author: CHRISTMANN, EDWIN PATRICK Degree: PH.D. Institution: OLD DOMINION UNIVERSITY 0418 Year: 1995

81. TEACHING YOUNG CHILDREN MUSIC FUNDAMENTALS IN A COMPUTER LEARNING ENVIRONMENT

Author: LEE, YU-WEN Degree: ED.D. Institution: COLUMBIA UNIVERSITY TEACHERS COLLEGE 0055 Year: 1994

83. THE DEVELOPMENT AND TESTING OF A COMPUTER-ASSISTED INSTRUCTIONAL PROGRAM TO TEACH MUSIC FUNDAMENTALS TO ADULT NONMUSICIANS

Author: PARRISH, REGENA TURNER Degree: ED.D. Institution: THE UNIVERSITY OF ALABAMA 0004 Year: 1994

85. DICTATION TUTOR: THE EFFECTIVENESS OF A CURRICULUM-SPECIFIC TUTORIAL IN THE ACQUISITION OF AURAL DISCRIMINATION SKILLS AT THE COLLEGE LEVEL

Author: HESS, GEORGE J., JR. Degree: D.A. Institution: UNIVERSITY OF NORTHERN COLORADO 0161 Year: 1994

86. INVESTIGATION OF THE EFFECT OF TEACHER-DEVELOPED COMPUTER-BASED MUSIC INSTRUCTION ON ELEMENTARY EDUCATION MAJORS

Author: LIN, SHEAU-YUH Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1994

91. TOOLS, GUIDELINES, AND STRATEGIES FOR THE DEVELOPMENT OF COMPUTER-ASSISTED-INSTRUCTION LESSONS BY NON-PROGRAMMING MUSIC TEACHERS (HYPERCARD)

Author: RICHMOND, CLARENCE FLOYD Degree: D.A. Institution: BALL STATE UNIVERSITY 0013 Year: 1994

95. LEARNING STYLE AND MUSIC INSTRUCTION VIA AN INTERACTIVE AUDIO CD-ROM: AN EXPLORATORY STUDY

Author: FORTNEY, PATRICK MICHAEL Degree: PH.D. Institution: UNIVERSITY OF MIAMI 0125 Year: 1993

97. THE DESIGN AND EVALUATION OF A COMPUTER-ASSISTED ERROR DETECTION SKILLS DEVELOPMENT PROGRAM FOR BEGINNING CONDUCTORS UTILIZING SYNTHETIC SOUND SOURCES (CAI)

Author: GRUNER, GREG L. Degree: D.A. Institution: BALL STATE UNIVERSITY 0013 Year: 1993

98. THE EFFECTIVENESS OF CAI AND LECTURE AS INSTRUCTIONAL STRATEGIES FOR TEACHING VOCAL ANATOMY AND FUNCTION TO UNDERGRADUATE MUSIC STUDENTS WITH DIFFERENT LEARNING STYLES

Author: ESTER, DON PAUL Degree: PH.D. Institution: THE UNIVERSITY OF NEBRASKA - LINCOLN 0138 Year: 1992

100. INTELLIGENT MUSIC LISTENING: AN INTERACTIVE HYPERMEDIA PROGRAM FOR BASIC MUSIC LISTENING SKILLS  
Author: GOODSON, CAROL ANN Degree: PH.D. Institution: THE UNIVERSITY OF UTAH 0240 Year: 1992

101. DEVELOPMENT AND TRIAL OF A COMPUTER-BASED INTERACTIVE VIDEODISC PROGRAM IN A COURSE IN FUNDAMENTALS OF CONDUCTING  
Author: FRY, RAYMOND JAY Degree: ED.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1992

102. A COMPUTER-BASED EVALUATION OF PITCH MATCHING SKILLS OF COLLEGE FRESHMAN STUDENTS IN MUSIC (MUSIC STUDENTS)  
Author: ETMEKTSOGLU, IOANNA E. Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1992

103. EFFECTS OF COMPUTER-ASSISTED INSTRUCTION IN A MASTERY LEARNING/COOPERATIVE LEARNING SETTING ON THE PLAYING ABILITIES AND ATTITUDES OF BEGINNING BAND STUDENTS (COOPERATIVE LEARNING)  
Author: KASSNER, KIRK Degree: PH.D. Institution: UNIVERSITY OF OREGON 0171 Year: 1992

104. AURAL REINFORCEMENT AND KINESTHETIC REINFORCEMENT AS VARIANTS OF THE RESPONSE MODE IN COMPUTER-ASSISTED HARMONIC AURAL SKILLS TRAINING (REINFORCEMENT)  
Author: POLOT, BARTON LEE Degree: PH.D. Institution: THE UNIVERSITY OF MICHIGAN 0127 Year: 1992

106. A COMPUTER-ASSISTED PROGRAM FOR THE SELECTION OF BAND MUSIC RELATIVE TO THE DIFFICULTY RATING OF INDIVIDUAL INSTRUMENTS (MUSICAL INSTRUMENT RATING, PERFORMANCE EVALUATION)  
Author: SAVILLE, KIRT RAYMOND Degree: ED.D. Institution: UTAH STATE UNIVERSITY 0241 Year: 1991

109. THE DEVELOPMENT AND TRIAL OF COMPUTER-BASED INTERACTIVE VIDEODISC COURSEWARE FOR TEACHING SKILLS IN THE VISUAL DIAGNOSIS OF SELECTED PROBLEMS IN TROMBONE PERFORMANCE  
Author: ATWATER, DAVID FRANKLIN Degree: ED.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1991

112. THE EFFECTIVENESS OF A COMPUTER-ASSISTED INSTRUCTION PROGRAM IN RHYTHM FOR SECONDARY SCHOOL INSTRUMENTAL MUSIC STUDENTS

Author: ORTNER, JOHN MICHAEL Degree: PH.D. Institution: STATE UNIVERSITY OF NEW YORK AT BUFFALO 0656 Year: 1990

113. DESIGN AND TRIAL OF A COMPUTER-ASSISTED SYSTEM SUPPLYING PRACTICE IN ERROR DETECTION FOR PRESERVICE INSTRUMENTAL MUSIC EDUCATORS

Author: JONES, DAVID LELAND Degree: D.M.A. Institution: UNIVERSITY OF GEORGIA 0077 Year: 1990

114. THE DEVELOPMENT OF A BEGINNING VIOLIN CURRICULUM INTEGRATING A COMPUTER MUSIC STATION WITH THE PRINCIPLES OF COMPREHENSIVE MUSICIANSHIP

Author: STRANGE, CHERYL MAY Degree: ED.D. Institution: COLUMBIA UNIVERSITY TEACHERS COLLEGE 0055 Year: 1990

118. THE EFFECTS OF COMPUTER-BASED INSTRUCTION ON ACHIEVEMENT OF FOUR, FIVE AND SIX-YEAR-OLD CHILDREN IN THE YAMAHA MUSIC EDUCATION SYSTEM PRIMARY ONE COURSE (MUSIC EDUCATION, FOUR-YEAR-OLD, FIVE-YEAR-OLD)

Author: BAILEY, DARRELL LEE Degree: ED.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1989

119. THE EFFECT OF INSTRUCTIONAL METHOD ON HANDICAPPED STUDENTS' COMPREHENSION OF SPECIFIC MUSIC CONCEPTS: COMPUTER VERSUS NONCOMPUTER INSTRUCTION (COMPUTER INSTRUCTION)

Author: ISLER-HAMILTON, ESTHER JANE Degree: PH.D. Institution: STATE UNIVERSITY OF NEW YORK AT BUFFALO 0656 Year: 1989

120. THE DEVELOPMENT OF A COMPUTER-BASED INTERACTIVE MULTIMEDIA PROGRAM FOR TEACHING INTERPRETIVE ASPECTS OF WIND INSTRUMENT NOTATION (MULTIMEDIA PROGRAM, NOTATION)

Author: ADAMS, STEVEN M. Degree: D.M.A. Institution: UNIVERSITY OF SOUTHERN CALIFORNIA 0208 Year: 1989

122. THE IMPLEMENTATION OF A MODEL PROGRAM OF COMPUTER-ASSISTED INSTRUCTION FOR CHILDREN'S CHOIRS IN A CHURCH SETTING

Author: SKELTON, DENNIS LANE Degree: D.M.A. Institution: THE SOUTHERN BAPTIST THEOLOGICAL SEMINARY 0207 Year: 1988

127. THE EFFECTS OF COMPUTER-ASSISTED MUSIC INSTRUCTION ON ACHIEVEMENT OF SEVENTH-GRADE STUDENTS

Author: KING, RICHARD VERN Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1988

128. DEVELOPMENT OF COMPUTER ASSISTED-INSTRUCTION FOR USE IN TEACHING ARABIC MUSIC THEORY

Author: SAIF, YASIN RAMADAN Degree: PH.D. Institution: UNIVERSITY OF SOUTHERN CALIFORNIA 0208 Year: 1988

129. THE DEVELOPMENT OF A MIDDLE SCHOOL GENERAL MUSIC CURRICULUM: A SYNTHESIS OF COMPUTER-ASSISTED INSTRUCTION AND MUSIC LEARNING THEORY (CAI)

Author: NELSON, BETH JOHANNA PEARCE Degree: D.M.A. Institution: THE UNIVERSITY OF ROCHESTER, EASTMAN SCHOOL OF MUSIC 0891 Year: 1988

130. THE INTEGRATION AND EVALUATION OF "MUSICLAND" IN A MUSIC LISTENING COURSE AND ACOUSTICS COURSE FOR TENTH GRADE STUDENTS (CAI)

Author: MEE, ROBERT ARTHUR Degree: PH.D. Institution: THE UNIVERSITY OF ROCHESTER, EASTMAN SCHOOL OF MUSIC 0891 Year: 1988

132. EFFECTIVENESS OF COMPUTER-ASSISTED INSTRUCTION IN DEVELOPING MUSIC READING SKILLS AT THE ELEMENTARY LEVEL

Author: HESSER, LOIS ANNETTE Degree: ED.D. Institution: STATE UNIVERSITY OF NEW YORK AT ALBANY 0668 Year: 1988

136. THE EFFECTIVENESS OF USING COMPUTER-ASSISTED INSTRUCTION WITH BEGINNING TRUMPET STUDENTS

Author: WEEKS, DOUGLAS GILMAN Degree: ED.D. Institution: BOSTON UNIVERSITY 0017 Year: 1987

137. THE EFFECT OF COMPUTER INSTRUCTION ON THE VERTICAL/HORIZONTAL MUSIC READING SKILLS OF THE GRAND STAFF FOR STUDENTS ENROLLED IN SENIOR HIGH SCHOOL BEGINNING KEYBOARD CLASSES

Author: HOLLAND, MARIANNE Degree: PH.D. Institution: UNIVERSITY OF SOUTH CAROLINA 0202 Year: 1987

138. A COMPARISON OF TWO SEQUENCES OF AURAL INTERVAL IDENTIFICATION DRILL ADMINISTERED TO COLLEGE STUDENTS THROUGH COMPUTER-ASSISTED INSTRUCTION (EAR TRAINING)

Author: KONECKY, LAWRENCE WAYNE Degree: MUS.ED.D. Institution: THE UNIVERSITY OF SOUTHERN MISSISSIPPI 0211 Year: 1986

140. THE DEVELOPMENT OF MELODIC CONCEPTS IN ELEMENTARY SCHOOL AGE CHILDREN USING COMPUTER-ASSISTED INSTRUCTION AS A SUPPLEMENTAL TOOL

Author: WHISTON, SANDRA KRISTINE Degree: PH.D. Institution: THE OHIO STATE UNIVERSITY 0168 Year: 1986

142. EFFECTIVENESS OF A COMPUTER-ASSISTED INSTRUCTION PROGRAM IN MUSIC FUNDAMENTALS APPLIED TO INSTRUCTION FOR ELEMENTARY EDUCATION MAJORS

Author: JACOBSEN, JEFFREY RICHARD Degree: D.M.E. Institution: UNIVERSITY OF NORTHERN COLORADO 0161 Year: 1986

144. THE EFFECTS OF A MICROCOMPUTER-ASSISTED TUNING PROGRAM ON JUNIOR HIGH SCHOOL STUDENTS' PITCH DISCRIMINATION AND PITCH-MATCHING ABILITIES (INTONATION, COMPUTER-ASSISTED)

Author: GLASS, JACQUALINE SHERRIE Degree: PH.D. Institution: UNIVERSITY OF MIAMI 0125 Year: 1986

147. DEVELOPMENT AND VALIDATION OF A COMPUTER-ASSISTED INSTRUCTIONAL LESSON FOR TEACHING INTONATION DISCRIMINATION SKILLS TO VIOLIN AND VIOLA STUDENTS

Author: EISELE, MARK JOSEPH Degree: D.MUS.ED. Institution: INDIANA UNIVERSITY 0093 Year: 1985

148. THE EFFECT OF DIFFERENTIAL FEEDBACK ON BEGINNING GUITAR STUDENTS' INTONATIONAL PERFORMANCE IN TUNING STRINGS (COMPUTER-ASSISTED INSTRUCTION (CAI))

Author: CODDING, PEGGY ANN Degree: PH.D. Institution: THE FLORIDA STATE UNIVERSITY 0071 Year: 1985

151. THE DEVELOPMENT OF INDIVIDUALIZED MUSIC LEARNING SEQUENCES FOR NON-HANDICAPPED, HANDICAPPED AND GIFTED LEARNERS USING THE LOGO MUSIC VERSION COMPUTER LANGUAGE

Author: MECKLEY, WILLIAM ALLEN Degree: PH.D. Institution: THE UNIVERSITY OF ROCHESTER, EASTMAN SCHOOL OF MUSIC 0891 Year: 1985

153. A COMPUTER-BASED TRAINER FOR MUSIC CONDUCTING: THE EFFECTS OF FOUR FEEDBACK MODES (CAI, PSYCHOMOTOR)

Author: SCHWAEGLER, DAVID GARY Degree: PH.D. Institution: THE UNIVERSITY OF IOWA 0096 Year: 1984

154. THE DEVELOPMENT AND IMPLEMENTATION OF A COMPUTERIZED PRESCHOOL MEASURE OF MUSICAL AUDIATION (APTITUDE, ABILITY, TESTING)

Author: FORSYTHE, ROSEMARY Degree: PH.D. Institution: CASE WESTERN RESERVE UNIVERSITY 0042 Year: 1984

155. DEVELOPMENT OF THE MUSIC LISTENING STRATEGY - TEMPO: COMPUTER ASSISTED INSTRUCTION IN MUSIC LISTENING

Author: TURK, GAYLA CLAIRE Degree: PH.D. Institution: UNIVERSITY OF KANSAS 0099 Year: 1984

157. THE DEVELOPMENT AND EVALUATION OF A MICROCOMPUTER-ASSISTED MUSIC INSTRUCTION PROGRAM FOR THE IMPROVEMENT OF TONAL MEMORY (CAI)

Author: ROBINSON, RUSSELL LOWELL Degree: PH.D. Institution: UNIVERSITY OF MIAMI 0125 Year: 1984

158. A SET OF MICROCOMPUTER PROGRAMS TO AID IN THE ANALYSIS OF ATONAL MUSIC

Author: RUSSELL, ROBERTA CRAM Degree: D.M.A. Institution: UNIVERSITY OF OREGON 0171 Year: 1983

159. COMPUTER-ASSISTED PROGRAMED INSTRUCTION TO TEACH PITCH AND RHYTHM ERROR-DETECTION SKILL TO COLLEGE MUSIC EDUCATION STUDENTS

Author: DEAL, JOHN JEFFREY Degree: PH.D. Institution: THE UNIVERSITY OF IOWA 0096 Year: 1983

162. A STUDY OF THE TACHISTOSCOPE IN TEACHING RHYTHMIC SIGHT-READING

Author: WRIGHT, TERRY OLEAN Degree: PH.D. Institution: THE LOUISIANA STATE UNIVERSITY AND AGRICULTURAL AND MECHANICAL COL. 0107 Year: 1982

164. AURAL-VISUAL INTERVAL RECOGNITION IN MUSIC INSTRUCTION: A COMPARISON OF A COMPUTER-ASSISTED APPROACH AND A TRADITIONAL IN-CLASS APPROACH

Author: SHANNON, DON WAYNE Degree: D.M.A. Institution: UNIVERSITY OF SOUTHERN CALIFORNIA 0208 Year: 1982

165. MISTI: A COMPUTER-ASSISTED INSTRUCTION SYSTEM IN MUSIC THEORY AND FUNDAMENTALS

Author: KUYPER, JON QUENTIN Degree: PH.D. Institution: THE UNIVERSITY OF IOWA 0096 Year: 1981

166. THE EFFICACY OF COMPUTER-BASED AND TAPE-RECORDED ASSISTANCE IN SECOND-SEMESTER FRESHMAN EAR-TRAINING INSTRUCTION

Author: GARTON, JANET CLAIRE Degree: PH.D. Institution: THE LOUISIANA STATE UNIVERSITY AND AGRICULTURAL AND MECHANICAL COL. 0107 Year: 1981

168. COMPUTER-ASSISTED MUSIC INSTRUCTION UTILIZING COMPATIBLE AUDIO HARDWARE IN COMPUTER-ASSISTED AURAL DRILL  
Author: WATANABE, NAN TEIKO Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1981
170. THE DEVELOPMENT OF CAI PROGRAMS FOR TEACHING MUSIC FUNDAMENTALS TO UNDERGRADUATE ELEMENTARY EDUCATION MUSIC METHODS CLASSES  
Author: WILSON, MARY LOUISE PRICE Degree: PH.D. Institution: THE LOUISIANA STATE UNIVERSITY AND AGRICULTURAL AND MECHANICAL COL. 0107 Year: 1981
174. THE DEVELOPMENT OF AN OBJECTIVE SIGHT SINGING ACHIEVEMENT TEST EMPLOYING ELECTRONIC MEASUREMENT APPARATUS  
Author: GRAVES, DAVID LEE Degree: ED.D. Institution: UNIVERSITY OF GEORGIA 0077 Year: 1980
175. THE EFFECT OF COMPUTER-BASED INSTRUCTIONAL MATERIALS IN A PROGRAM FOR VISUAL DIAGNOSTIC SKILLS TRAINING OF INSTRUMENTAL MUSIC EDUCATION STUDENTS  
Author: SANDERS, WILLIAM HUSTON Degree: PH.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1980
180. COMPUTER-ASSISTED INSTRUCTION IN MUSIC: A PROGRAM IN RHYTHM FOR PRESERVICE ELEMENTARY TEACHERS.  
Author: LINDEMAN, CAROLYNN ANDERSON Degree: D.M.A. Institution: STANFORD UNIVERSITY 0212 Year: 1979
181. A STUDY OF THE CONTRAST BETWEEN COMPUTER-ASSISTED INSTRUCTION AND THE TRADITIONAL TEACHER/LEARNER METHOD OF INSTRUCTION IN BASIC MUSICIANSHIP.  
Author: VAUGHN, ARTHUR CLARENCE, JR. Degree: PH.D. Institution: OREGON STATE UNIVERSITY 0172 Year: 1978
183. COMPUTER-ASSISTED INSTRUCTION IN MUSIC: A SURVEY WITH ATTENDANT RECOMMENDATIONS.  
Author: JONES, MORGAN JOHN Degree: PH.D. Institution: NORTHWESTERN UNIVERSITY 0163 Year: 1975
184. THE EFFICACY OF COMPUTER ASSISTED INSTRUCTION COMPARED WITH TRADITIONAL TEACHER-TAUGHT AND SELF-TAUGHT METHODS OF TEACHING BEGINNING MUSIC THEORY.  
Author: COOPER, ROSE MARIE Degree: PH.D. Institution: THE UNIVERSITY OF NORTH CAROLINA AT GREENSBORO 0154 Year: 1975



185. FEASIBILITY OF COMPUTER-ASSISTED INSTRUCTION FOR INSTRUMENTAL MUSIC EDUCATION.

Author: PETERS, GEORGE DAVID Degree: ED.D. Institution: UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN 0090 Year: 1974

186. COMPUTER-ASSISTED INSTRUCTION IN THE PUBLIC SCHOOL GENERAL MUSIC CLASS: A COMPARATIVE STUDY

Author: VON FELDT, JAMES RONALD Degree: D.M.A. Institution: UNIVERSITY OF MISSOURI - KANSAS CITY 0134 Year: 1971

187. A COMPARISON OF RESPONSE-SENSITIVE AND RESPONSE-INSENSITIVE DECISION RULES IN PRESENTING LEARNING MATERIALS IN MUSIC THEORY BY COMPUTER-ASSISTED INSTRUCTION

Author: HULLFISH, WILLIAM ROUSE, JR. Degree: ED.D. Institution: STATE UNIVERSITY OF NEW YORK AT BUFFALO 0656 Year: 1969

## Vita

Daniel Emerson Clouse was born in Reed City, MI on April 16, 1976. He went to Central Michigan University and graduated in May, 2000 with a B.S. in Music Performance with a major in Tuba Performance and a minor in Management Information Systems. Mr. Clouse will graduate from the University of Tennessee, Knoxville, in August 2006 with an M.M. in Music Theory and Technology.

Mr. Clouse worked in Cincinnati, OH as a computer programmer, web developer, database administrator, and network engineer while freelancing as a professional musician. He has written software or websites for organizations such as Fannie Mae, Wells-Fargo Bank, Chiquita, Proctor and Gamble, and the state of Florida.

He has studied and played tuba with many renowned teachers and performers, including Don Harry, Sam Pilafian, Brian Bowman, and Deanna Swoboda. He has played with many prestigious ensembles, including the Cincinnati Brassband and the Tennessee Brass. Daniel won the Tubonium Tuba Solo Competition held at Gustavus Adolphus College in St Peter, MN in March 2006.