UNIVERSITY OF PORTSMOUTH

Systems Engineering Research Group
Department of Mechanical and Design Engineering

NEW MUSIC TECHNOLOGY SYSTEMS AND METHODS TO ASSIST TEACHERS

By

Alexandre LASSAUNIERE

Dipl. Ing., MEng, CEng, MIEE

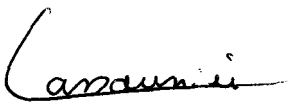
This Thesis is submitted in partial fulfilment of the requirement for the award of the degree of Doctor of Philosophy of the University of Portsmouth, following research conducted by the author with the Systems Engineering Research Group, Department of Mechanical and Design Engineering, University of Portsmouth. The research was in part funded by: the Department of Trade and Industry and EPSRC through a Teaching Company Scheme in collaboration, with Counterpoint MTC Ltd.

The author has not been a registered candidate for another award by the University of Portsmouth or any other University during the Research programme.

June 2003

DECLARATION OF CONFORMITY

I hereby declare that the research documented in this Thesis has been undertaken by the author and that any work included that was not undertaken by the author has been appropriately attributed.



A. Lassauniere

ABSTRACT

This dissertation describes the creation of new music technology and information systems: electronic music interfaces, keyboard and audio networks, a novel information system called an electronic student assessment and data management system and new intelligent tutoring systems.

Keyboard systems are used by music teachers. Existing systems offered a range of useful features using a number of technologies. However, they also had drawbacks, which prevented users from performing efficiently. Students had to use the new systems with portable keyboards for their lessons and were required to learn how to play the keyboard by the national curriculum. However, students' music and keyboard playing skills varied. A new intelligent tutoring system was created to overcome this problem by delivering customised tutoring on music and keyboard skills through the new systems.

A broad base of research was completed and the following were the tangible research successes. The new music technology systems consisted of a network of new electronic music interfaces that could be remotely controlled from a computer using a new communication protocol. New object oriented control structures to interface with non-computer networks were defined to enable the accurate setting and display of the system status, as well as management of communication-problems and abnormalities. The novel information system was a new three-tier information system created to help teachers manage students and their work.

The new music technology and information systems used a new ergonomic graphical user interface. A new mobile teaching system using these systems (tablet PCs, software remote control and local station control) allowed teachers to perform their teaching anywhere. These systems were used as a test bed to implement artificial intelligence. A new habits capturer, using a new stamping method was created to anticipate user's actions from monitored activity. Also, a new computer aided instruction system to teach basic music-keyboard skills was created. This new system used the new music technology systems (through a novel interfacing method using keylighting, text-to-speech and key capturing technologies), and a new music-keyboard skills expert used as a music concept translator.

The computer aided instruction system formed the base for the creation of a new intelligent computer aided instruction system (also called intelligent tutoring system) to teach basic music-keyboard skills in a custom way. This new intelligent system used a new methodology. The new methodology showed that a tutor module that gathered knowledge, expertise and assessment functions was viable. The new intelligent system also used a new generic tutoring base. Finally, a new learning curves plotter was created to display student modelling information from the new intelligent system.

It has gone from having three computers to boasting £40,000 worth of equipment in a long-overdue new music block, in which two rooms are equipped with a KAAN (Keyboard and Audio Network) system.

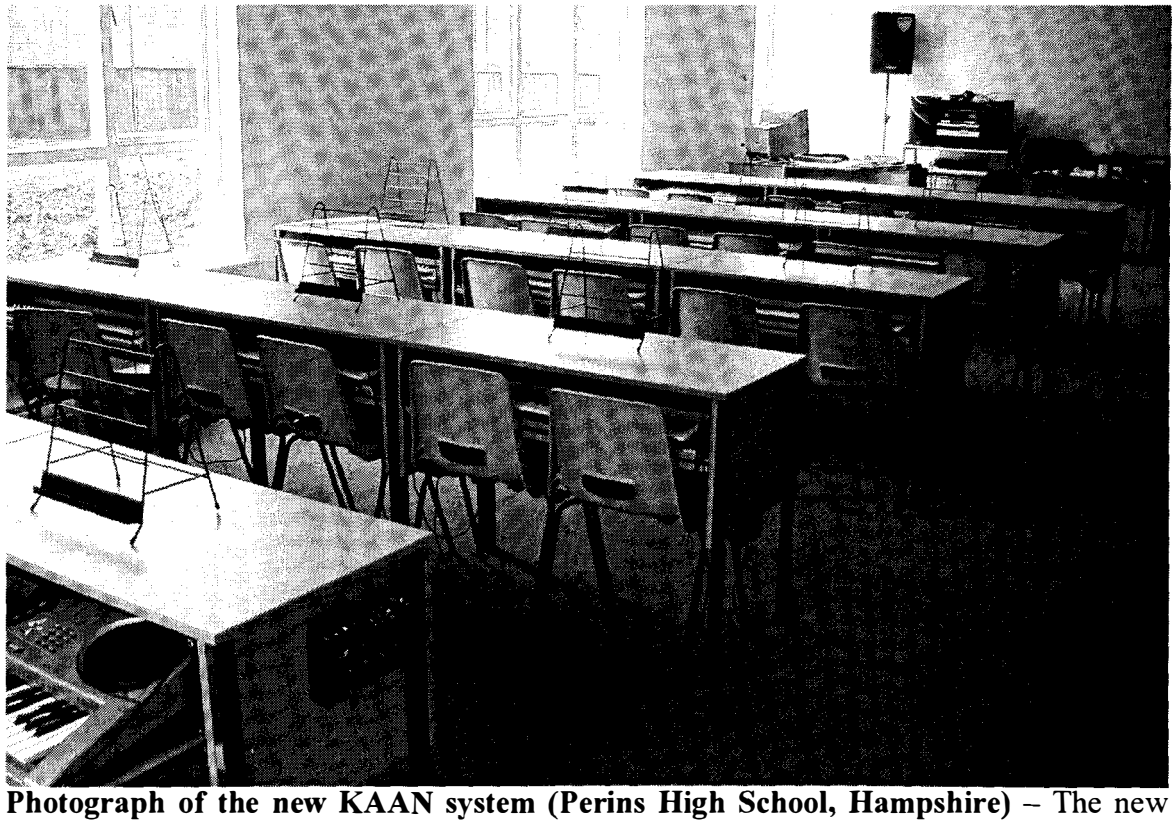
The teacher says it means students no longer have to share a keyboard and can work independently. The system divides each keyboard into two and, with headphones, two students can play their half at any pitch and in any instrument.

However, they can also play with another person or as part of a larger group without having to move around the room. "It gives total flexibility to each student as well as letting them combine in all sorts of ways," the teacher explains.

In her view, students are not asked to share a piece of paper in art, so they should not have to share in music either.

Holy Trinity is one of the first schools to install KAAN, which also allows music files to be shared and lets teachers monitor students and record their performances. Although the system was only installed a couple of months ago, it is proving popular with both teachers and students and the teacher believes it is already helping to improve students' compositions. Another room at the school houses 15 computers with their own keyboards, and a range of soundcards and music software.

Chris Johnston 10/05/2002
Times Educational Supplement, Sounds to inspire (extracts)



Photograph of the new KAAN system (Perins High School, Hampshire) – The new KAAN system is marketed by Counterpoint MTC Ltd (Worthing, UK). The design for this new system was provided by the author. The detail of this work is described as part of this dissertation.

A PITCHOUNE

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to the following:

Specifically:

To Dr David Sanders, Dr Giles Tewkesbury and Dr Alex Gegov for agreeing to supervise the research and for their help, guidance and encouragement throughout. Many thanks to them for the speed with which they assisted with the last steps of the writing-up.

Again to Dr David Sanders, for providing the initial opportunity to conduct this research programme and for the management of academic matters related to the research.

And to Dr Giles Tewkesbury, for providing the expertise on electronics and software engineering and the immensely helpful collaboration on the development of the hardware.

My sincere thanks go to:

Peggy, my wife, for all her support, encouragement and technical expertise.

Counterpoint MTC Ltd for sponsoring the research and for enabling me to assimilate my engineering skills and test my research in a commercial environment.

Andrew Close, James Marchant and Duncan McKrill at Counterpoint MTC Ltd for their support, assistance, expertise, and links with teachers, which were valuable and helpful to the research.

My friends Benoit Jaworski and John O'Gadhra for their patience and the time they spent reviewing the thesis.

All the testers, who tested the tutoring system.

TCD for the sponsorship of the first two years of the research and their representative Roger Hollely.

Other important people to thank:

My family and family-in-law, supportive and encouraging throughout.
My friends from Portsmouth and France for their support and entertainment.
Colleagues past and present at Counterpoint MTC Ltd.

CONTENTS

GLOSSARY – ACRONYMS

A-D	Analogue to Digital
AES/EBU	Audio Engineering Society/European Broadcasting Union
AI	Artificial Intelligence
ALU	Arithmetic and Logic Unit
ANN	Artificial Neural Network
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CAI	Computer Aided Instruction
CD	Compact Disk
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Units
CRC	Cyclic Redundancy Check
D-A	Digital to Analogue
DBMS	DataBase Management System
DCOM	Dynamic Component Object Model
DIN	Deutsches Institut für Normung
Dll	Dynamic Link Library
DSP	Digital Signal Processing
E-R	Entity Relationship
EEPROM	Electrically Erasable Programmable Ream-Only Memory
EMI	Electronic Music Interface
ESAAMS	Electronic Student Assessment And data Management System
FTP	File Transfer Protocol
HCI	Human-Computer Interaction
HTTP	HyperText Transfer Protocol
I/P	Input
I/O	Input / Output
IBM	International Business Machine
IC	Integrated Circuit
ICAI	Intelligent Computer Aided Instruction
ICT	Information and Communication Technology
IEC-	International Electrotechnical Commission
IEEE	Institute of Electronic Engineer (US Institute of Electrical and Electronic Engineers)
ISO	International Standard Organisation
ISP	In System Programming
IT	Information Technology
ITS	Intelligent Tutoring System
KAAN	Keyboard And Audio Network
KAI	Keyboard Aided Instruction
LAN	Local Area Network
MAC	Macintosh (Apple)
MAC (address)	Media Access Control
MIDI	Musical Instrument Digital Interface
MP3	Mpeg layer 3
MS	Microsoft

NC	National Curriculum
NIC	Network Interface Card
O/P	Output
OCX	Active X
OMG	Object Management Group
OOP	Object Oriented Programming
OP AMP	Operational Amplifier
OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
RAM	Random Access Memory
RF	Radio Frequency
ROM	Read Only Memory
RSxxx	Recommended Standard
S/PDIF	Sony/Philips Digital Interface
SPI	Serial Peripheral Interface
SQL	Structured Query Language
STP (cable)	Shielded Twisted-Pair
TCP/IP	Transmission Control Protocol / Internet Protocol
TCS	Teaching Company Scheme
TDM	Time Division Multiplexing
TV	Television
UART	Universal Asynchronous Receiver Transmitter
UML	Unified Modelling Language
UTP (cable)	Unshielded Twisted-Pair
VB	Visual Basic
VCR	Video Cassette Recorder
WIN	Windows
WAN	Wide-Area Network
WAV	Wave file
WMA / WMV	Windows Media Audio / Video

CHAPTER 1. INTRODUCTION

1.1. Teaching music in the United Kingdom (UK)

Many institutions used music technology systems such as computers or music keyboards to assist music teachers [Driver (1994) and So *et al.* (1994)]. Teachers could use computers with specialised software such as sequencing or music knowledge assessing tools. Some computers were connected via their sound card to a portable keyboard (as shown in Figure 1-1) using the Music Instrument Digital Interface (MIDI) standard and were networked to share recorded resources, which could also be found in classrooms [Lehrman and Tully (1993), Smoliar *et al.* (1995), Penfold (1995), Buick and Lennard (1997) and Krogh (1998)]. Other more sophisticated devices were installed in mini studios to produce high quality digital recording. Finally, audio systems were created to connect music keyboards together and to record or group students from a teaching position [Aoyagi and Hirata (1992), Counterpoint MTC Ltd (n.d.1, n.d.2), Korg Corporation (n.d.), Nicholas Haines (n.d.), Rane (n.d.) and Music Central Inc. (n.d.)].

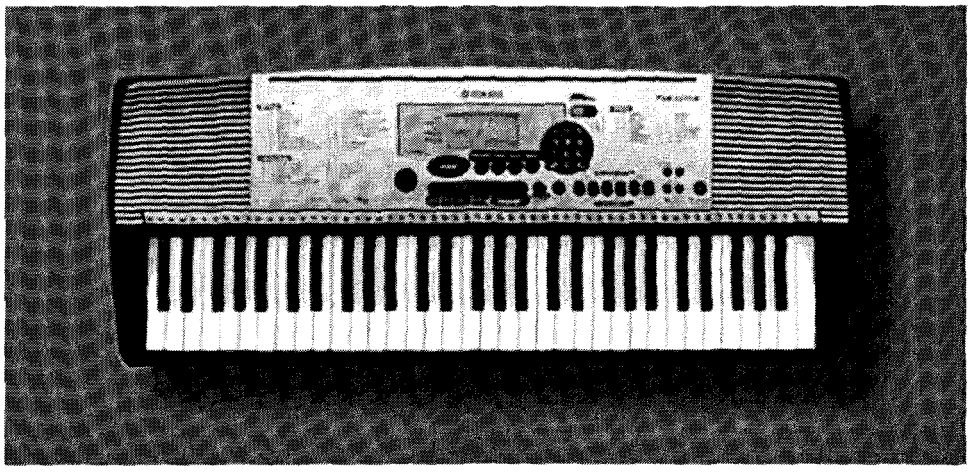


Figure 1-1: Example of a portable keyboard

[Yamaha PSR 225GM model, taken from Yamaha Corporation (n.d.2)]

Music teachers had a wealth of technological options to assist in their teaching. However, none of the cited tools were designed specifically for teaching and many had drawbacks for effective use in a classroom.

In a classroom environment, a teacher might have 30 or more students with 16 or more keyboards. Furthermore, in the UK, teachers often had two students sharing a single keyboard. To handle this effectively, the teacher needed a way to interact with students both individually and collectively.

1.2. Motivation

Portable music keyboards were excellent resources for ensemble performance, which all students could play. Also, many students were intrigued by them and behaved well, which gave a disciplinary advantage to teachers. Thus portable keyboards became popular in music classrooms. However, these 'beneficial' portable keyboards were not used effectively and raised more drawbacks than advantages. This had deterred teachers.

Music technology systems, connecting portable keyboards, were designed to create a music infrastructure for ready-to-use keyboards. Keyboards were then used more effectively. Available music technology systems used purely audio, with analogue audio signals from standard portable keyboards routed to a front desk. There, audio signals could be grouped and mixed together or recorded onto a tape recorder. Other music technology systems added some control facilities using the MIDI protocol on specifically designed devices.

With such music technology systems, teachers could listen or speak to any student from a teaching position and group them for teamwork. However, these audio and MIDI controlled music technology systems were limited as some offered inadequate flexibility, a lack of connectivity for other instruments or devices, and poor audio performance and/or quality.

1.3. Research Aim and Objectives

The thesis set out and described in this dissertation is that:

New intelligent music technology systems and methods could be created to assist teachers

During the research, a new music technology system was created in order to assist music teachers more effectively. The new music technology system was designed in consultation with music teachers and assessed by them.

The objectives were to identify and design a new and novel flexible teaching-centred music technology system and to investigate new methods to assist music teachers in their teaching.

For the new Keyboard And Audio Network system, the specific objectives were to:

- Collect teachers' requirements for a new music technology system.
- Investigate existing music technology systems and technologies.
- Investigate and define the new music technology system design.
- Investigate and define communication protocols.
- Investigate, define and design a control interface between the music technology system and teachers.
- Investigate, define and design a new information system.
- Investigate and implement methods to improve teachers' mobility.

For the new Intelligent System, the specific objectives were to:

- Collect feedback from teachers about the new music technology and information systems.
- Investigate solutions using Artificial Intelligence.
- Investigate, define and design methods to deliver new knowledge to students and to assess their answers.
- Implement a new Computer Aided Instruction system.
- Investigate, define and design an Intelligent Tutoring System.

- Propose and implement a new structure for the new intelligent system.
- Propose generic methods using the new structure for other fields.

1.4. Methodology

The new music technology systems were considered as required flexible and controlled infrastructure on which new teaching tools could be applied including new software and intelligent tools.

The research initially focused on creating a fully functional new music technology system called a Keyboard And Audio Network. This was supported by the later creation of a new software application called an Electronic Student Assessment And data Management System based on the information system created as a tool for the new music technology system. Various new software tools were also created to install and maintain the new music technology and information systems.

Subsequent research considered the implementation of intelligent assistance to help solve remaining issues. Two solutions were investigated for feasibility and utility and one was fully researched. The selected solution aimed at implementing an Intelligent Tutoring System within the new music technology systems to teach basic music and keyboard skills to new students.

1.4.1. Keyboard And Audio Network system

Existing music technology systems were benchmarked to capture benefits and issues to be solved, as well as using audio technology. Four audio networking technologies were investigated to determine the most appropriate for implementation. New Electronic Music Interfaces were then designed as network nodes. A new communication protocol was created to allow remote control of the new electronic interfaces from a control device. The best control device was found to be a computer and new control software was designed to interface with

the new music technology system, using Human-Computer Interaction methods to improve usability.

1.4.2. Electronic Student Assessment And data Management System

From teachers' requirements, a new information system was implemented. Due to the success of the new tool, a new separate information system called Electronic Student Assessment And data Management System was created that built on the base of the new music technology control system, and targeted teachers of any subject. Both systems have since been marketed by a collaborating company and results from working systems have been collected. It was decided to create a dual mode music technology and information system to implement new features on both systems.

1.4.3. Intelligent Tutoring System

A new Intelligent Tutoring System was created to overcome issues with beginners learning music. The students had to be taught basic keyboard and music skills, and this was a requirement of the National Curriculum for year 7 students in the United Kingdom. The new intelligent system was designed generically so that it could be used later in other fields of teaching. A new structure was proposed to reflect the structure of the new intelligent system used with the new music technology system and compared with an existing Intelligent Tutoring System structure based on work from Gerlič (1998).

1.5. Research claims

Research into new music technology systems and new Intelligent Tutoring Systems has been undertaken. A broad base of research was completed and a foundation platform from which further research and development could be

effected was created. In respect of the research work completed the following were the tangible research successes:

MAIN SYSTEMS CREATED

- Electronic Music Interface.
- Keyboard And Audio Network and communication protocol.
- Electronic Student Assessment And data Management System.
- Computer Aided Instruction.
- Intelligent Computer Aided Instruction (also called Intelligent Tutoring System) with learning curves plotting.

NEW METHODS

- Control using ergonomic user interfaces.
- Object Oriented control for interfacing.
- Teaching-centred information management.
- Methods that allow teachers to move around.
- Capturing of habits (with a new stamping method).
- Music-keyboard skills expert.
- Interfacing with KAAN.
- ITS methodology.
- Generic tutoring.

The key contribution was the creation of a new Intelligent Tutoring System using the new music technology systems to assist music teachers. The utility broadening of the new systems, including the new generic tutoring base also assisted teachers from other fields.

1.6. Overview of the dissertation

Chapter 2 gives an overview of the field of music teaching, concentrating on teaching in the UK, and especially of the use of keyboards and music technology systems. The chapter then describes available hardware, firmware and software technologies that could be used for the creation of a new music technology

system. The chapter finishes with an overview of Artificial Intelligence techniques and applications that could be used.

Chapter 3 describes the creation of the new KAAN system hardware and firmware. The chapter shows the design of the new Electronic Music Interfaces and networking configurations to form a KAAN system. The chapter then explains the definition of the new communication protocol used to remotely control the new KAAN system.

Chapter 4 considers the design of the new software for the new KAAN system and introduces the creation of the new ESAAMS system. The chapter first defines interfacing and ergonomic issues and solutions and then investigates object oriented control structures to obtain real-time accurate information from the music technology system. The chapter continues with the creation of a new teaching-centred information system embedded within the control software, which defined the base of the new ESAAMS system.

Chapter 5 reviews the new music technology and information systems utility and usability using feedback from teachers using the new systems. Two main issues are considered separately in two sub-projects to define a useful solution. These sub-projects consider the scope of the research for its feasibility and utility. A new habits capturer using a new stamping method was created for the first sub-project. For the second sub-project, a new interfacing method between the computer and students via the new KAAN system was created with a basic music-keyboard skills expert to create a potential ITS. Both sub-projects were tested and compared. The second sub-projects seemed to be of more use for teachers using the new KAAN-ESAAMS systems.

Chapter 6 describes the research performed on the sub-project selected in Chapter 5. The chapter describes the creation of a new Computer Aided Instruction (CAI) system using the new expert and interface methods created in Chapter 5. The new CAI system helped validate a new tutoring base, storing knowledge on basic keyboard and music skills. The chapter finishes with the creation of a new Intelligent CAI system (ICAI, also called ITS) through the

implementation of a new student model. This provided customised teaching as well as learning curve plotting. The new intelligent system was tested with a number of students to validate and improve the new intelligent music technology systems and methods.

Chapter 7 concludes the research findings and discusses some successes and failures of the research. The conclusions from each of the chapters are discussed and further avenues of research are summarised. These are discussed in the context of creating and developing new music technology systems and methods to help music teachers teach music.

CHAPTER 2. LITERATURE SEARCH

The areas of expertise to overview could be split in three parts:

- How music education is performed in the UK.
- Technologies that would be useful in the creation of the new music system.
- Application of Artificial Intelligence (AI) on non-intelligent systems.

This chapter reviews the literature in these areas to set the scene for the research discussed in the dissertation.

2.1. Instruction in UK

2.1.1. National Curriculum

The National Curriculum (NC) for Music (Key stages 1 to 3) is defined by the British Department for Education and Employment (n.d.) and sets out a clear, full and statutory entitlement to learning for all students. It determines the content of what will be taught, and sets attainment targets for learning. It also determines how performance will be assessed and reported. The NC for the appropriate Key stages was consulted to check the teaching guidelines during the research.

In Besanet (2002) expenditure on Information and Communication Technology was estimated to be £493m:

- £9,000 / average primary school in 2002-03.
- £43,000 / average secondary school in 2002-03.

Half was to be spent on desktop and laptop computers, the rest ear-marked for spend on peripheral and networking equipment, technical support, software and content provision.

From equipment supplier catalogues [Counterpoint MTC Ltd (n.d.3)], budget for music equipment is spent on:

- Instruments (acoustic, electric, analogue or digital).
- Sound recording (studios).
- Computer systems (computer network with music software).
- Public Address and HiFi systems (amplifier and playback).
- Other music systems.
- Material and resources (books, internet resources).

2.1.2. Device Aided Instruction

Music and audio devices manufacturers tried to expand their share of the market by marketing easy-to-use keyboards and software. However, they found that there were needs to acquire knowledge about music outside education to allow their products be used at home and by non-professionals. They decided to embed music teaching in their products and/or to market music teaching software. Teachers then started to use these new tools in their lessons. Examples are:

A) Computer Aided Instruction.

B) Keyboard Aided Instruction.

A) COMPUTER AIDED INSTRUCTION SYSTEMS

Computer Aided Instruction (CAI) is specialised software that may or may not interface with MIDI devices through the computer sound card. Three products [Music Lessons from MiBAC (n.d.) and Rising Software, Auralia (n.d.1), and Musition (n.d.2)] were reviewed and are described in detail in Appendix 8.1, page 235.

MUSIC lessons had three main fields of study:

- Note reading: displayed 4 notes on staff and student had to play each note on the keyboard (different clefs available).
- Key signatures against major/minor keys.
- Play 8 notes of a determined scale.

AURALIA and *MUSITTION* were produced by the same company (Rising Software), *AURALIA* being based around practical music concepts, whereas *MUSITTION* was aimed at music theory. Some tests included singing recognition from the audio input of the sound card.

- *AURALIA:*
  - Intervals and scales.
  - Chords.
  - Rhythm.
  - Pitch and melody.
- *MUSITTION:*
  - Music reading.
  - Terms and symbols.
  - Key centres.
  - Instruments.

This software worked around simple lessons and a series of tests that a student had to answer either with the mouse or a connected MIDI keyboard. Once students responded correctly to their test they progressed to a more difficult level, otherwise the CAI system could give them (or not) the correct answer and another attempt. Students could record their level to stop and restart the lesson at the same point. Statistics could be given concerning their results.

B) KEYBOARD AIDED INSTRUCTION SYSTEMS

Keyboard Aided Instruction (KAI) was new and no standard had been agreed between keyboard manufacturers. The concept was called Yamaha Education Suite for Yamaha (n.d.2). The concept derived from step piano lessons and lessons on how to play built-in or downloaded songs.

2.1.3. Music Advisors' opinion on the use of keyboards

MIDI portable keyboards and PCs tended to be widely used in music education due to the broad range of features and the ease of use they offered advisors [Odam (1997), Rogers (1997), Salaman (1997), Odam and Walters (n.d.)]. Existing keyboards featured:

- MIDI standard [Hinton (2003)] to connect to other MIDI keyboards and/or devices such as computers.
- Sequencers to record on board multi-tracked work. This feature could have been popular if more keyboards offered a facility to save work at the end of a class.
- Full size (piano size) keys and 'touch sensitive' feature for musical expressiveness.

From these features, benefits were multiple:

- Resource for ensemble performance.
- Theoretically possible to provide all students with performing skills.
- Use of timbre/voices with a variety of sounds.
- Headphones (no noise → peaceful working atmosphere).
- All students could play.
- Fingering was straightforward and built up manipulative skills.
- Comparatively cheap.
- Most students were intrigued by them → they behaved well, which gave a disciplinary advantage.
- 'Rightness' and 'wrongness' of students' work on keyboards were easier to detect and measure than most other activities in the music classroom.
- Little trouble in noting students' progress.
- Keyboards were useful in teaching of staff notation (statutory requirement of the national curriculum).
- Use of single finger chords and of rhythm accompaniment.
- Sequencing.

In the opinion of music advisors [Odam (1997), Rogers (1997), Salaman (1997), Odam and Walters (n.d.)], keyboards were not used as efficiently as expected because of the way teachers used them:

- Not adequate as sole equipment.
- Intrusive 'Demo' facility.
- Headphones (students usually in solitary confinement ➔ divided students and lessons were not always productive).

One of their solutions was to have the keyboards set out, fixed and ready to use, and keyboard labs were useful. Keyboard labs usually connected all keyboards to a central console where:

- The teacher could:
  - Listen to any instrument (without the students necessarily knowing).
  - Speak directly to individual students or the whole class.
  - Switch off some instruments from the console.
- Two students sat per keyboard and could listen and play at the same time.

Their drawbacks were:

- Expensive and liable to break down.
- Two students per keyboard meant that:
  - Both students worked on the same melody 1 or 2 octaves apart.
  - Both students composed in pair, with no precise goal defined.
- (Language) labs had met with only moderate success.
- No one established best practice for music laboratories.

However, these new music devices and computer-aided systems brought new teaching methods. The teacher remained the teaching expert as well as an engineer ready to install, support and maintain the resources with more or less success.

From the music advisors' observations, teaching tended to focus more on group work. The size of groups varied. There were challenges to strike a balance between the needs of the individual and the needs of the group in such a short time as that afforded by music lessons.

Two main points were highlighted:

- How to develop keyboard skills in non-keyboard players.
- It was vital that research focused on the use and development of keyboards as a resource.

2.1.4. Existing Music Technology Systems

Music technology systems were developed to create an infrastructure for music teachers to better use their keyboards. Available music technology systems could be used purely for audio, where analogue audio signals from standard portable keyboards were routed to a front desk. There, audio signals were grouped and mixed together or recorded onto a tape recorder [see Figure 2-1, Counterpoint MTC Ltd (n.d.1 and n.d.2) and Nicholas Haines (n.d.)].

Other music technology systems might have added some control facilities using the MIDI protocol on specifically designed devices [see Figure 2-2, Music Central Inc. (n.d.)], or were fully digital [Korg Corporation (n.d.), Rane (n.d.) and Yamaha Corporation (n.d.1)].

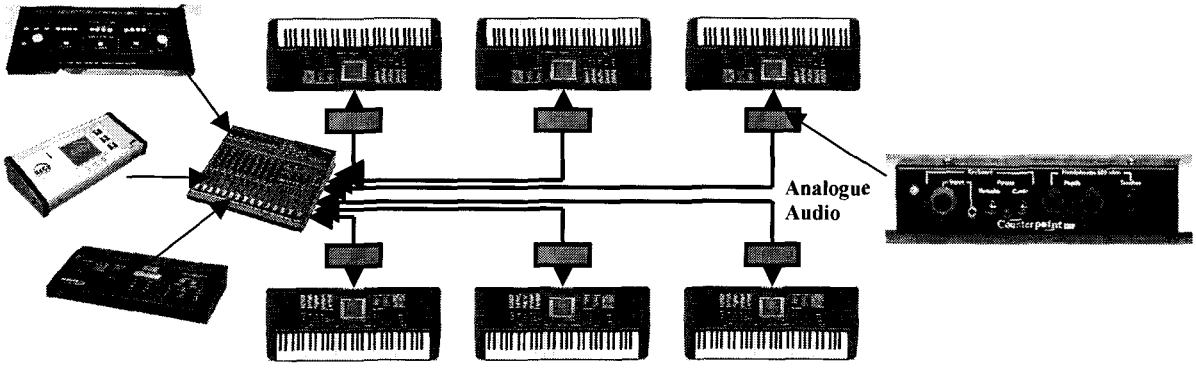


Figure 2-1: Audio music technology system (first systems type)
[Pictures taken from Counterpoint MTC Ltd (n.d.1 and n.d.2) and Nicholas Haines (n.d.)]

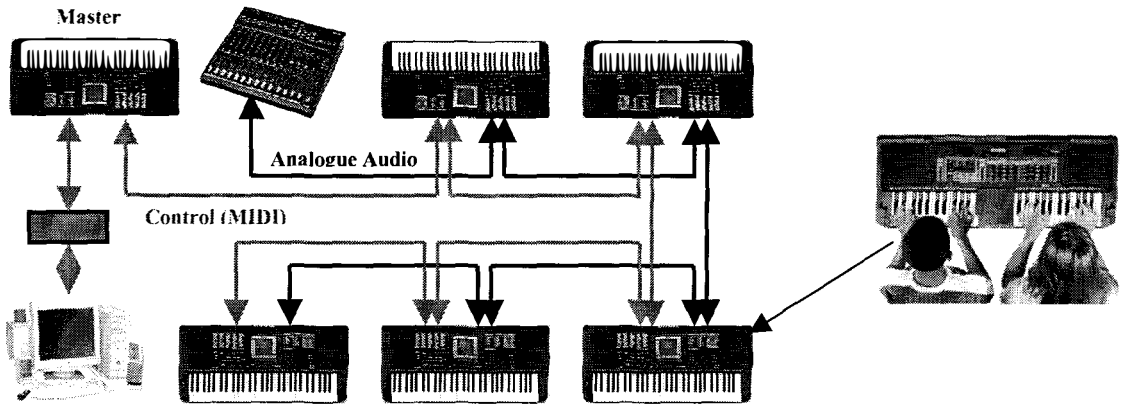


Figure 2-2: MIDI controlled music technology system (second system type)
[Pictures taken from Music Central Inc. (n.d.)]

These music technology systems offered:

- *A classroom Infrastructure:*

  Students could use keyboards individually or in pairs. The keyboards were usually installed on desks and cables were hidden in trunking. Power might also be networked and converted at each station to power up the keyboards. This was in order to reduce the number of power supplies.

- *Listen to a student from the front:*

  Teachers could easily listen or speak to any student from the front desk using an audio network. Thus the recording of students' works was centralised.

- *Grouping:*

  The teacher's console may allow mixing of audio signals from the students together to create groups.

However, these audio and MIDI controlled music technology systems were limited as they offered:

- *Inadequate Flexibility:*

  Keyboards could be used alone, but the music technology systems themselves could not be used beyond their basic functions. The music technology systems might also have required specific devices to run properly.

- *No Extra Inputs:*

  The number of extra inputs (e.g. microphone, guitar, mixer, etc.) was dictated by the number of unused sockets on the front desk.

- *Inability to share Recording:*

  Audio was recorded either in an analogue format (e.g. tape) or digitally as a MIDI file. Only MIDI files could be shared easily. However, both types of music technology system lacked real digital recording as Wave, MP3 or WMA files.

- *Poor Audio Performance:*

  The audio music technology systems required many long cables across the classroom. Such cables easily picked up interferences, which might have caused cross talk and hum.

- *Poor Reliability:*

  Music technology systems might have included electronic interfaces between the keyboards and the front desk. These interfaces needed not be complex and were not designed for rugged use.

These music technology systems used analogue and digital technologies for audio and data as well as numerous networking techniques. The following section reviews these technologies.

2.2. Hardware and Firmware

This section reviews some suitable technologies at the time of writing concerning audio signals processing and networking. Some of these technologies were used to develop the existing music technology systems described in 2.1.4, page 14.

2.2.1. Audio

Sound is nothing more than the time varying change in air pressure [Rapture (n.d.)]. Due to the laws of physics, sound is differentiable and continuous. The theoretical range of human hearing extends from 20Hz to 20kHz with over 120dB of dynamic range. Sound, due to its continuous nature, is analogue. Analogue

audio systems are capable of being at any value at any time (within the confines of the analogue audio system and not taking into account the effect of noise). Digital systems are not continuous, but rather discrete, meaning that at a given time the digital system must be at one of a possible set of numbers.

Technologies for analogue and digital systems are described in:

A) Analogue Technology.

B) Analogue Sampling.

C) Digital Technology.

D) Audio Files.

A) ANALOGUE TECHNOLOGY

Analogue audio instruments consist of three blocks [microphones, amplifier, hearing-coil, see Hearing Siemens (n.d.)], with potential additional stages:

- Filters / equalizers.
- Mixers.

Winer (1981) explains that two types of filter are high-pass (or low-cut) or low-pass (or high-cut) filters, which respectively filter out low and high frequencies from a source signal. Two other kinds of filters also exist, the *Band-pass filter* and its counterpart *Band-stop filter (or Notch filter)*. In this case, a range of frequencies is allowed (or denied) to pass through the filter.

Simple circuitry is used to mix analogue audio signals together [Gingrich (1995)], using operational amplifiers [op-amps, Liscott (n.d.)]. Figure 2-3 shows several current sources driving the negative input of an inverting amplifier. Summing the current into the node gives:

$$\frac{V1}{R1}+\frac{V2}{R2}+\frac{V3}{R3}=-\frac{Vout}{RF}$$, or if R1=R2=R3=R: $$Vout=-\frac{RF}{R}(V1+V2+V3).$$

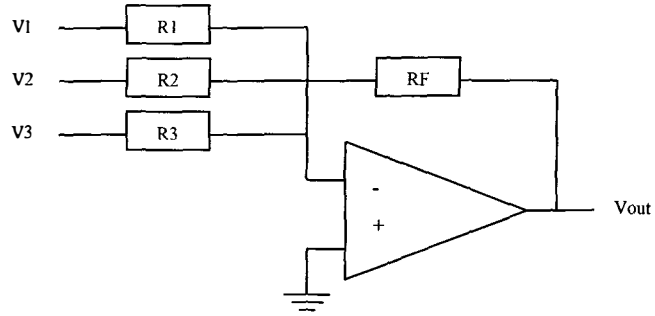


Figure 2-3: Current summing amplifier used for mixing audio signals [from Gingrich (1995)]

Analogue audio is broadly used, as analogue circuits are simple and cheap. However digital technology and analogue sampling could also be used to handle audio and bring other benefits. Quantisation (digitisation) is the process of turning a continuous signal into a discrete signal by breaking it into quanta, or blocks [Rapture (n.d.)]. Each block contains two pieces of information: time and amplitude. The way time is measured is by using a set sampling frequency. This frequency is the number of times per second that the amplitude is measured. The amplitude is measured with a resolution that is limited by the electronics. The characteristics of these two measurements determine the quality, cost, complexity and overall operation of a digital audio system.

B) ANALOGUE SAMPLING

Nyquist and Shannon tell that the sampling frequency Fs must be at least twice as high as the considered highest frequency [Rapture (n.d.)]. The Compact Disk (CD) standard sampling rate is 44.1kHz, just over twice 20kHz, which is the limit of human hearing. New standards have been developed that utilise higher sampling rates (up to 2.8MHz to decrease error at high frequencies).

Resolution is the other measure of a digital system. A digital system with 8 bits of resolution can differentiate between 2^8 = 256 different levels. This is acceptable for low quality speech but not for music. The CD standard is 16 bits per channel stereo, which can represent $2^{16} \sim 65{,}000$ levels, which gives approximately 96 dB of range [20 log(65,000)] . Recent progress in electronics has brought higher resolutions (up to 24 bits per channel).

Once a signal is in the digital domain it can be transferred between standard digital equipments. A standard has been set for this and is known as IEC-958 (or S/PDIF), and comes in two types: optical and RF. The S/PDIF digital audio system is a known standard, reliable and is compatible with the AES/EBU standard used by professional audio equipment globally [Rapture (n.d.)].

Time-Division Multiplexing is a method of putting multiple data streams in a single signal by separating the signal into many segments, each having a short duration [Whatis-Techtarget (n.d.), term: TDM]. Each individual data stream is reassembled at the receiving end based on their timing slot.

The circuit that combines signals at the source (transmitting) end of a communications link is known as a multiplexer. It accepts the input from each individual end user, breaks each signal into segments, and assigns the segments to the composite signal in a rotating, repeating sequence as shown in Figure 2-4. The composite signal thus contains data from multiple senders. At the other end of the long-distance cable, the individual signals are separated out by means of a circuit called a demultiplexer, and routed to the proper end users. A two-way communications circuit requires a multiplexer / demultiplexer at each end of high-bandwidth cable.

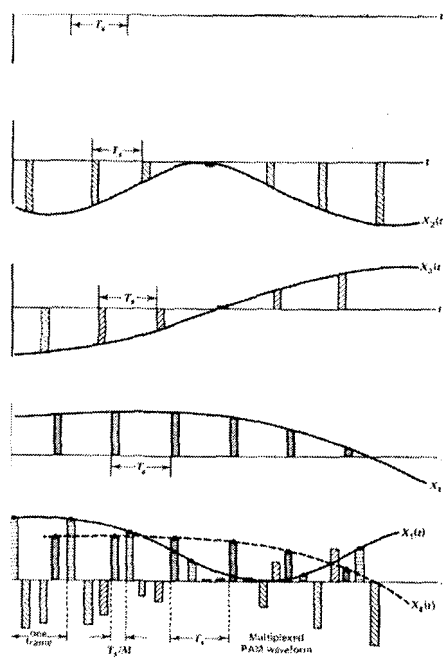


Figure 2-4: TDM techniques with 3 sampled analogue signals and the resulting signal at the bottom [Whatis-Techtarget (n.d.), term: TDM]

C) DIGITAL TECHNOLOGY

Another method to digitise analogue signals is performed by Analogue-to-Digital (A-D) converters. When data acquisition hardware receives an analogue signal it converts it to a voltage. It then digitises it with an analogue-to-digital converter, ready for transfer to a computer. Many types of specifications for A-D converters are quoted by hardware manufacturers: resolution, linearity, offset errors, sample and hold acquisition time, throughput, integration time and re-calibration.

Digital audio is less subject to noise than analogue signals as alterations in voltage of signals (noise) are dealt with differently: the amplitude of the analogue signal would differ from the original signal with the noise amplitude, whereas digital signals are composed of 'words' of bits (or byte) having only two states (0=0V, 1=5V for instance). A small noise will not change a 0 to a 1. Bytes also contain parity bits to validate the contained data bits. If a byte is not received correctly, it can be resent or re-read.

D) AUDIO FILES

Audio, if stored in audio files, can be played back later in compatible audio devices. Various audio file formats existed to store audio information. Wave format [*.wav, Webopedia (n.d.), term: wav] was developed jointly by Microsoft and IBM, and support for WAV files was built into Windows 95 making it the de facto standard for sound on Personal Computers (PCs). Audio amplitude is stored at a defined sampling rate (usually 44100 samples/s), defined resolution (8 or 16 bits) and in a set number of channels. Wave files can become large and that is why compression techniques appeared to reduce the size of audio files using psychoacoustic techniques.

A psychoacoustic model is based on studies of human perception showing that the average human does not hear all frequencies the same way [Rice – Welsh (n.d.)]. Effects due to different sounds in the environment and limitations of the human sensory system lead to facts that can be used to cut out unnecessary data in an

audio signal. The two main properties of the human auditory system that make up the psychoacoustic model are:

- *Absolute threshold of hearing* (humans can hear frequencies in the range from 20 Hz to 20,000 Hz).
- *Auditory masking* (humans do not have the ability to hear minute differences in frequency).

Mp3 is the file extension for MPEG, audio layer 3 [Webopedia (n.d.), term: mp3]. Layer 3 (for audio signals) uses perceptual audio coding and psychoacoustic compression to remove all superfluous information (by a factor of 12 from a CD). A new format created by Microsoft, Windows Media Audio (*.wma), reduces even more this ratio with an improved quality.

Moreover, audio can be streamed using the new WMA or RealAudio formats, which is the de facto standard for streaming audio data over the World Wide Web (www).

2.2.2. Computing

Digital signals are expressed in bits having two states 0 or 1 and are then described in binary or base 2. Bits are gathered in blocs of 8 bits or bytes, and a byte expresses a value between 0 and 2^8-1 = 255 in decimal (base 10). A byte can also be described by two 4-bit chunks, which are better expressed in hexadecimal (base 16, dec255 = hexFF).

Computing describes bits and byte manipulation from an Arithmetic and Logic Unit (ALU). The ALU manipulates those binary numbers and executes all the calculations allowing the program to take decisions:

- Arithmetic operators: Add, Subtract, Multiply or Divide.
- Logical comparators: AND, OR, NOT, or XOR.
- Operand comparators: greater, smaller or equal.
- Operand value: positive, negative or equal to 0.

Systems requiring computation are built with microprocessors or microcontrollers. These are described in:

A) Central Processing Units (CPU).

B) Types of computer devices.

C) Types of operating system.

A) CENTRAL PROCESSING UNITS

Three characteristics differentiate microprocessors [Webopedia (n.d.), term: microprocessor]:

- *Instruction set*: The set of instructions that a microprocessor can execute.
- *Bandwidth*: The number of bits processed in a single instruction.
- *Clock speed*: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.

In comparison a microcontroller is a highly integrated chip that typically contains a CPU, RAM, some form of ROM, I/O ports using Universal Asynchronous Receiver Transmitters (UARTs) and timers. Unlike a general-purpose computer, which also includes all of these components, a microcontroller is designed for a specific task - control.

In addition to processor and controllers, a coprocessor is a special-purpose processing unit that assists a CPU in performing certain operations. A Digital Signal Processor (DSP) is a special type of coprocessor designed for performing the mathematics involved in Digital Signal Processing. Most DSPs are programmable, which means that they can be used for manipulating different types of information, including sound, images, and video.

B) TYPES OF COMPUTER DEVICES

PCs first appeared in the late 1970s [Webopedia (n.d.), term: PC]. One of the first and most popular personal computers was the Apple II, introduced in 1977 by Apple Computers. During the late 1970s and early 1980s, new models and competing operating systems appeared. In 1981, IBM entered the fray with its

first PC, known as the IBM PC. The IBM PC quickly became the personal computer of choice. One of the few companies to survive IBM's onslaught was Apple Computer, which remains a major player in the personal computer marketplace.

Computers can be generally classified by size and power although there is considerable overlap:

- *Personal computer or Workstation:* A small, single-user computer based on a microprocessor. In addition to the microprocessor, a personal computer has a keyboard for entering data, a monitor for displaying information, and a storage device for saving data.
- *Minicomputer:* A multi-user computer capable of supporting from 10 to hundreds of users simultaneously.
- *Mainframe:* A powerful multi-user computer capable of supporting many hundreds or thousands of users simultaneously.
- *Supercomputer:* A faster computer that can perform hundreds of millions of instructions per second.

The principal characteristics of personal computers are that they are single-user computer systems based on microprocessors. Although personal computers are designed as single-user computer systems, it is common to link them together to form a network.

Other computer formats include (using a stylus and handwriting recognition):

- *Personal Digital Assistant* (PDA, as shown in Figure 2-5):
  A PDA is a handheld device that may combine computing, telephone/fax, Internet and networking features. PDAs are also called palmtops, hand-held computers and pocket computers.

Figure 2-5: PDA

[From Compaq-Hewlett Packard (n.d.)]

- *Tablet PCs* (as shown in Figure 2-6):

  A tablet PC is a type of notebook computer. Compared to a PDA, the tablet PC relies on digital ink technology, using a digitiser to capture the movement of the special-purpose pen and record the movement on the LCD screen.

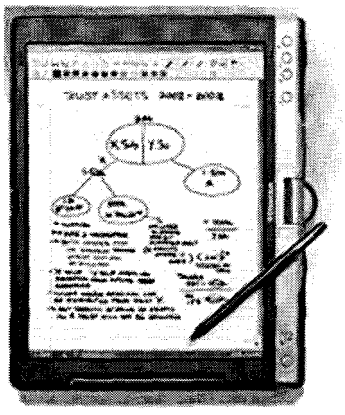


Figure 2-6: Tablet PC

[from Microsoft (n.d.3)]

C) TYPES OF OPERATING SYSTEM

The Operating System (OS) is important control software that manages all computer resources as shown in Figure 2-7 [Gupta (1996), page 95]. It allows operations such as starting the computer, locating a file or resolving conflicts. It also lets the PC recognise input from a keyboard or run different programs such as word processing or spreadsheets.

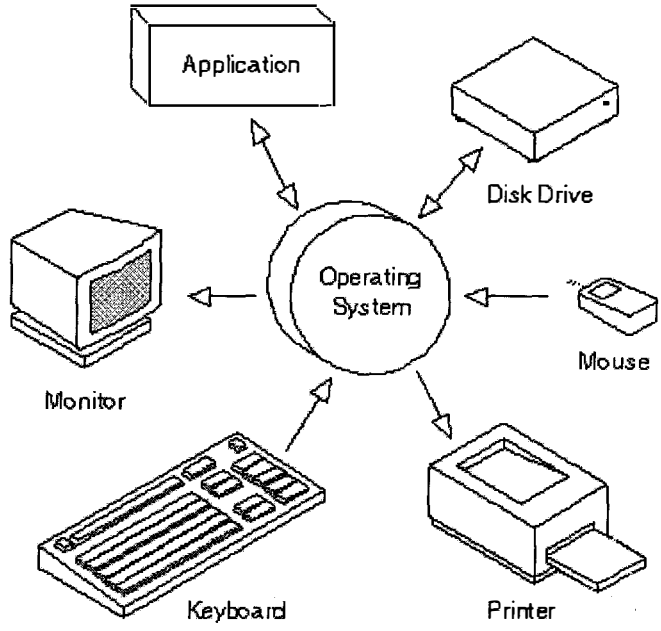


Figure 2-7: Operating system hierarchical diagram
[from Webopedia (n.d.), term: OS]

MS Windows is a common operating system for PCs. Released 32 bit versions are Win95, 98 and ME as well as WinNT, 2000 and XP, also WinXP Tablet PC edition for tablet PCs and WinCE for PDAs. Mac OS X is the current version for the Apple Macs. Other Operating Systems include:

- Unix.
- Linux.
- Aix (IBM).

Some applications (Cytrix, PC Anywhere, Windows Terminal Services, etc.) allow a user to remotely control other computers. Such applications usually lock the other running computer and display the interface of the operating system of the controlled machine as shown in Figure 2-8. Windows XP professional features this for remote control or to remotely assist and support users, using other interfacing means offered by MS Messenger (chat, audio and video).

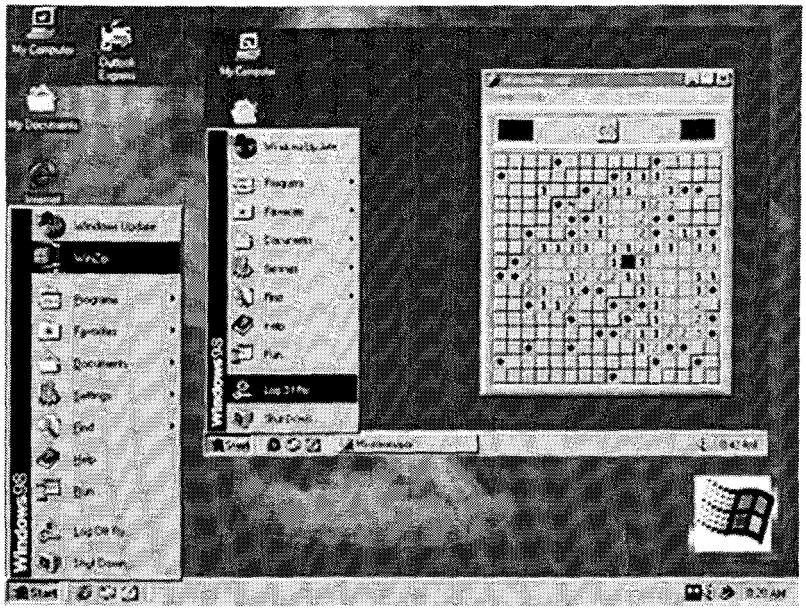


Figure 2-8: Remote control of another computer (captured in 2003)

To allow computers to control each other, they need to communicate with each other and to be connected together in a network.

2.2.3. Networking

Until the early 1980's, the benefits from having computers linked together were not understood. However, computers rapidly grew in the market place, due to new processor technology and cost, along with demand for fast and fully connected and integrated information systems with minimum maintenance [RAD (1994)].

Linking computer systems, accessing the Internet or corporate servers, and connecting communication devices into a strategic informational infrastructure is called Networking or Internetworking. When Local Area Networks (LAN) are interconnected, they are part of a Wide-Area Network (WAN). WANs can be connected through public networks, such as the telephone system, leased lines or satellites. The largest WAN in existence is the Internet [Webopedia (n.d.), term: WAN].

There are three principal network topologies, which can also be used together [Hodson (1997), page 53 and Webopedia (n.d.), term: Network]:

- *Bus topology:*

  This consists of a single communications channel. Each connected device is attached to the medium, called bus or backbone, at an interface point and has its own unique hardware address.

- *Ring topology:*

  A ring topology is configured as a ring for the data transmission. Each network interface connection has its own hardware address for identification and copies the data from the network when it identifies the packet's destination address as its own. Token ring topology is so named since a station cannot transmit until it notionally holds a token, which is circulating around the ring.

- *Star topology:*

  All devices are connected to a central switching system or hub. This topology gives better overall reliability against cable faults than the previous two approaches, as only a single node will be unavailable in the event of a cable failure. However, overflow of data through the hub can create bottlenecks.

The following are considered:

A) Data communication and transmission.

B) Communication standards.

C) MIDI standard, protocol and files.

A) DATA COMMUNICATION AND TRANSMISSION

When two devices communicate, a receiver is often set to run at the same speed as the sender (synchronisation). Consequently, each device has a clock. The clock at the sending end tells the sending device when to transmit a bit of data onto the line [Dick (2002), page 370]. Once the data has correctly been sent and received, the sending device must know that the data has been correctly

transferred. If it has not, the sender sends the data again. The whole concept of data communications is based on sending data, checking its correct receipt and confirming how successful the transfer was [Hodson (1997), page 1].

A protocol is an agreed format, for transmitting data between two devices. There are diverse standard protocols, varying in terms of simplicity, reliability, and speed. If users want to communicate with another device, they have to ensure that the device supports the correct protocols [Webopedia (n.d.), term: protocol]. Each protocol determines:

- The type of error checking to be used.
- The data compression method, if any.
- How the sending device indicates it has finished sending a message.
- How the receiving device indicates it has received a message.

Synchronous transmission uses two synchronised clocks (sending and receiving), and data is sent at known intervals [Dick (2002), page 370]. Asynchronous transmission is however the most common method of connecting PCs where the hardware has to be told when a character is about to be received, and when the transmission of the character has ceased. This implies enclosing the bits of data with extra bits known as the 'start' and 'stop' bits. The start bit takes the signal off the idle state, incoming bits can then be sampled at the clock rate and the stop bit returns the communication system back to the idle state. Conventionally, the Least Significant Bit (LSB) of the data is transmitted first.

When transmitting ASCII files, 7 bits are used to represent the range of characters. The eighth bit is known as the parity bit and can be used by the receiving end to ensure that the character transfer has been successful. Parity checking may be appropriate for small ASCII files, however, as the data size to be transmitted and the speed of data transfer increase, more automated and sophisticated approaches are needed.

If a packet is lost or corrupted during transmission, it is important that this is detected so that retransmission can take place to correct the problem. An international standard approach to error detection is to protect the data with a

checksum or Cyclic Redundancy Check (CRC), which compares a calculated remainder before and after sending the data. If the end remainder does not match the transmitted remainder, an error is detected. The device will then request that the block of data is resent by sending a Negative AcKnowledgement (NAK). If it does match, an acknowledgment signal (ACK) is sent to the transmitter.

It was previously assumed that the destination is capable of receiving data at whatever rate. In reality, this may not be true. If data is sent more quickly than the receiver can manage, including the buffering capability, data overflow occurs and data is lost. Also, in a half-duplex link (see Table 2-1), only one of the computers can transmit data at any time. To overcome these issues, handshaking methods are used:

- *Hardware Handshaking:*

  Each pin of a Recommended Standard-232 (RS232) serial port of a PC and a modem answers each other to agree on a communications protocol and to show they are ready for the next step.
- *Software Handshaking:*

  Software handshaking reduces the number of pin connections needed. When a PC needs to know when transmission can happen, the modem sends out specific non-printable ASCII numbers/codes to start and stop data transmission (Xon/Xoff method).

Simplex	Half-duplex	Full-duplex
Data is sent in one direction. This is no longer in regular use.	Data is transmitted both ways, but only one at a time. CB radio was the first to use it: while one party spoke, the other listened.	Data is transmitted at once both ways, speeding up transmissions. Most modems work in this mode.

Table 2-1: Serial connections types [from Hodson (1997), page 25, Webopedia (n.d.), term: half-duplex and Dick (2002), page 380)

Registered Jack-45 (RJ-45) are eight-wire connectors used to connect PCs onto a LAN, especially Ethernet and Token Ring networks. They plug into the standard adapter cards and hubs or access units. Each of the LANs requires a different pin connection combination within the plug.

Twisted pair cable consists of two wires that provide a signal and a return path, with the wires tightly twisted into a spiral configuration. This reduces the problems of noise, cross talk, and other interferences. Shielded twisted pair cable (STP) also provides further resilience to noise problems. Twisted pair cable is defined on a six level basis, each category being rated in megahertz.

Other media include:

- Coaxial cable.
- Fibre optics cable.
- Wireless media.

B) COMMUNICATION STANDARDS

The starting point in establishing Open Systems Interconnection (OSI) standards was that many standards already existed, and it was harder for networks that used different proprietary networking systems to communicate with each other. In 1984, the International Standard Organisation (ISO) put together the OSI Reference Model, which describes seven layers of protocols to comply with for computer communications [Gralla (1996), page 12]:

- *Application layer 7:*

  This layer is the link between two application processes by providing a range of service interfaces for application programs (e.g. Email, directory and file transfer services). This layer verifies the availability of intended communication partners, synchronises and establishes agreement on procedures for error recovery and control of data integrity.

- *Presentation layer 6:*

  This layer maps the various data formats into a common external data representation that will allow correct interpretation of the information on receipt. Encryption of data may be provided for confidentiality or security and compression of data and terminal emulation may also be implemented.

- *Session layer 5:*

  It establishes, manages and ends calls to exchange the data bit streams. It offers co-ordination between users by choosing mutually suitable protocols, and effects checkpoints for data recovery.

- *Transport layer 4:*

  This acts as the interface between the user's activities and the requirements of the data communications network. It provides two way, reliable, cost effective, end-to-end exchange of data. Flow control and sequencing of the data blocks is performed at this level.

- *Network or Internet layer 3:*

  It provides connectivity and path selection between two host networking systems that may be located on geographically separated networks. It adds unique addressing information to packets and maps addresses to network addresses.

- *Data link layer 2:*

  The data link layer is the foundation of a network. By using packet switching technology, its main task is to provide error free transmission between two end stations attached to the same physical cable or media. This allows the next higher layer to assume virtually error-free transmission over the physical link.

- *Physical layer 1:*

  This layer defines the electrical, mechanical, procedural, and functional specifications for activating, maintaining, and deactivating the physical link between two hosts. Specifications may include voltage levels, physical data rates, maximum transmission distances, or physical connectors (plugs and sockets). Ethernet and Token Ring are the two most common physical layer protocols. They run at the Media Access Control (MAC) level and move the data over the cables based on the physical address on each Network Interface Card (NIC).

TCP/IP refers to two protocols working jointly to deliver data: The Transmission Control Protocol (TCP, the transport layer 4 protocol), and the Internet Protocol (IP, the network layer 3 protocol). When information is sent across an intranet, it is broken into small packets. The packets are sent independently through a series of switches called routers. Once all the packets arrive at their destination, they are recombined into their original form. The TCP breaks the data into packets and recombines them on the receiving end. The IP handles the routing of the data and makes sure it gets sent to the proper destination [Gralla (1996), page 10].

RSxxx standards come in a variety of specifications depending on various parameters as shown in Table 2-2 from [RS485 (n.d.)].

	RS232	RS422	RS485
Cabling	single ended	single ended multi-drop	multi-drop
Number of Devices	1 transmit 1 receive	5 transmitters 10 receivers	32 transmitters 32 receivers
Communication Mode	full duplex	full duplex half duplex	half duplex
Max. Distance	50 feet at 19.2 Kbps	4000 feet at 100 Kbps	4000 feet at 100 Kbps
Max. Data Rate	19.2 Kbps for 50 feet	10 Mpbs for 50 feet	10 Mpbs for 50 feet
Signalling	unbalanced	balanced	balanced

Table 2-2: Three RSxxx standards with differentiating parameters [from RS485 (n.d.)]

The IEEE 1394 (FireWire) high-speed interconnection enables simple, low-cost, high-bandwidth real-time data connectivity between computers, peripherals and consumer electronics. Analysts predict that by 2004, 150 million consumer products—including camcorders, VCRs, printers, PCs, TVs, and digital cameras—will incorporate FireWire [Focus (n.d.)].

USB combines multiple existing interfaces into a single, easy-to-use connector. USB's plug-and-play capability ends the formerly complex process of adding computer system peripherals. USB offers three speeds, all three offering both asynchronous and isochronous (real-time) data transmission over a simple and inexpensive 4-wire cable:

- Low-speed (1.5 Mbps)
- Full-speed (12 Mbps, or USB 1.1)
- High-speed (480 Mbps, or USB 2.0)

Wireless standards include two main standards [Webopedia (n.d.), term: Wireless]:

- IEEE 802.11 (also known as Wi-Fi, short for Wireless Fidelity).
- BLUETOOTH.

Another standard, which deals with music instruments, is the MIDI standard. The next section explains it in detail as it was primarily used in this project.

C) MIDI STANDARD, PROTOCOL AND FILES

MIDI stands for Musical Instruments Digital Interface [Hinton Instruments (2001)]. Within a year of its introduction in 1983, MIDI was adopted as standard by the entire electronic musical instrument industry worldwide with a high degree of compatibility. The reason behind the overwhelming success was that the standard was carefully researched. There were no loopholes; hardware, protocol and operation were completely defined.

The MIDI hardware circuitry was designed to be cheap to implement and foolproof to connect. MIDI may be connected incorrectly without damage to itself due to the distribution of its current limiting resistors, but this also limits the maximum cable run to 15 metres which is easily exceeded in-between two rooms in a studio or on even across a medium sized stage.

The interface operates at 31.25Kbaud (+/- 1%), asynchronous, daisy chain (between chain and ring topologies), with a start bit, 8 data bits (D0 to D7), and a stop bit. Used connectors are DIN 5 pin (180 degree) female panel mount receptacle and shall be labelled "MIDI IN" and "MIDI OUT". A "MIDI THRU" output may be provided if needed, which provides a direct copy of data coming in MIDI IN.

MIDI messages comprise a STATUS byte (bit 7 = 1) followed by DATA bytes (Bit 7 = 0). Messages are divided into two main categories:

- *Channel messages* (see Table 2-3):

  Channel messages contain a four-bit channel number encoded into the Status byte, which addresses the message specifically to one of sixteen channels. For Channel messages only, the Status byte may be omitted if it would otherwise repeat the last Status byte sent.

	Channel number n = 0 to F (Channels 1 to 16)	
Note Off	8n, kk, vv	Key number, kk = 00 to 7F 3C = middle c
Note On	9n, kk, vv	Velocity, vv = 01 to 7F 40 = no velocity; 00 = note off
Polyphonic Aftertouch	An, kk, vv	
Control Change	Bn, cc, vv	Control number, cc = 00 to 65
Mode Change	Bn, cc, vv	Control number, cc = 79 to 7F
Program Change	Cn, pp	Program number, pp = 00 to 7F
Channel Aftertouch	Dn, vv	Pressure value, vv = 00 to 7F
Pitchbend	En, ll, hh	ll = lsb, hh = msb Centre = 2000h = En, 00, 40

Table 2-3: MIDI Channel messages [from Hinton Instruments (2001)]

- *System messages:*

  System messages are not encoded with channel numbers and are divided into three main types:

  - System Common.
  - System Exclusive (or Sysex, see Table 2-4).
  - System Real Time (Table 2-5).

System Exclusive Start	F0, ID, ...	table for manufacturers' IDs
Quarter Frame	F1, dd	MIDI Time Code message dd = data
Song Position Pointer	F2, ll, hh	ll = lsb, hh = msb hhll = 14 bit counter 1 count = 6 timing clocks
Song Select	F3, ss	Song number, ss = 00 to 7F
(undefined)	F4	
(undefined)	F5	
Tune Request	F6	
System Exclusive End	F7	"EOX"

Table 2-4: MIDI System Common Messages with Sysex Start/End bytes [from Hinton Instruments (2001)]

Timing Clock	F8	24 clocks = 1 crotchet
Timing Tick	F9	1 tick = 10 milliseconds
Start	FA	
Continue	FB	
Stop	FC	
(undefined)	FD	
Active Sensing	FE	
System Reset	FF	

Table 2-5: MIDI System Real Time Messages [from Hinton Instruments (2001)]

A predefined set of controllers (Bn, cc, vv) exists and manufacturers can implement the controllers they desire for each or their models. Sysexs enable manufacturers to create customised messages with their own ID, within some predefined specifications. These can be used to manipulate sequences (MIDI bulk dump) between devices. A device not recognising a message drops it and waits for another one.

MIDI files store MIDI messages and other information in a determined format so that they can be shared and played by different devices [John Stone (n.d.)]. These files are small compared to other audio file formats, as only messages of a few bytes are stored against frequency and amplitude information.

MIDI files contain one or more MIDI streams, with time information for each event. Song, sequence, and track structures, tempo and time signature information, are all supported. Track names and other descriptive information may be stored with the MIDI data. This format supports multiple tracks and multiple sequences.

Sequence files are made up of chunks. Each chunk has a 4-character type and a 32-bit chunk length, which is the number of bytes in the chunk. There are two types of chunks:

- *Header chunk* (MThd):

  A header chunk provides a minimal amount of information pertaining to the entire MIDI file. Three format exist:

  - 0: the file contains a single multi-channel track.
  - 1: the file contains one or more simultaneous tracks of a sequence.
  - 2: the file contains one or more sequentially independent single-track patterns.

- *Track chunk* (MTrk):

  A track chunk contains a sequential stream of MIDI data, which may contain information for up to 16 MIDI channels. The concepts of multiple tracks, multiple MIDI outputs, patterns, sequences, and songs may all be implemented using several track chunks.

2.2.4. Software

Software (or programs) can be launched to complete tasks for users on computers running an Operating System to interface with the hardware. Software can be separated into two categories:

- *Client Software:*

  Can be running on a stand alone computer or LAN-attached machines. All processing and memory handling is performed at the client unless a thin client structure is used where the client runs an application running on a server. It is therefore mainly used to interface with the user.

- *Server Software:*

  Servers are usually dedicated computers (Domain, WEB, application, file, printer, proxy, etc. servers) accessed only through LAN connections and are service providers. Servers allow central operations for multi-users and need to be dimensioned carefully.

Applications are designed to work on specific Operating Systems (OS) and a same program would need to be written differently to work on MS Windows or Mac OS. This is because applications may need to call or reference OS specific Application Program Interface (API) functions, stacked in libraries called Dynamic Link Libraries (Dlls).

Applications are designed and compiled using various development tools. In early computer programming ages, programs used to be written and compiled in punched cards (used as a storing media). Then, programs were designed to work as a sequence of actions, where variables could be manipulated through subroutines and functions. Media changed, but the programming method remained and development tools such as (Quick) Basic, Pascal, C (++) appeared so that programmers could develop applications more easily. Recently arrived is a new concept with MS Windows 95, where programming changed from being sequence centred to object centred.

New development tools appeared such as Delphi (visual Pascal), Visual C++, Visual Basic or Java. The 'Visual' concept is important as objects could have a graphical interface (and were then called controls). Graphical User Interfaces (GUIs) were designed more easily and Human-Computer Interaction (HCI) improved. The latest technology from Microsoft is called .Net and still uses Object Oriented Programming (OOP), but has merged many existing development languages into a unique language. This new technology also emphasises the use of the Internet and the use of web browsers as thin client running applications sitting in web servers (ASP .NET technology).

The following are considered:

A) Object Oriented Programming.

B) Unified Modelling Language.

C) Graphical User Interfaces, Human-Computer Interaction.

D) Databases.

A) OBJECT ORIENTED PROGRAMMING

Object Oriented Programming (OOP) is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure [Webopedia (n.d.), term: OOP]. In this way, the data structure becomes an object that includes both data (known as properties) and functions (known as methods). In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects and communicate with each other through events.

One of the principal advantages of OOP techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify and transport to other applications.

The software industry has considerable interest in reducing the time and effort required to custom applications [Mkpe (1999)]. In recent years the introduction of 'middleware' has proved popular. Component technology, a type of middleware product, was created as a solution for reusing software in a wide variety of applications. Center stage in the middleware lineup are Distributed Common Object Models (DCOM, from Microsoft) and Common Object Request Broker Architectures (CORBA, from Object Management Group). Both technologies, while not directly compatible, are designed to accomplish similar goals, namely platform independent object communication over networks. Because a component can 'plug-in' to many different development environments, its flexibility is inherent.

Some COM and DCOM objects are also known as Active X (OCX), which differ from Dynamic Link Libraries (Dlls) only because they may offer a visual interface. Dlls offer objects and functions (such as APIs), whereas OCX offer objects with a visual interface (or control).

B) UNIFIED MODELLING LANGUAGE

UML is Short for Unified Modelling Language, a general-purpose notational language for specifying and visualizing complex software, especially large, object-oriented projects [Webopedia (n.d.), term: UML]. It is developed under the auspices of the OMG and is the industry-standard language for specifying, visualising, constructing, and documenting the artefacts of software systems.

UML simplifies the complex process of software design, making a 'blueprint' for construction [Rational (n.d.)]. Good models are essential for communication among project teams and to assure architectural soundness. As the complexity of software systems increase, so does the importance of good modelling techniques.

Yeates *et al.* (1994) state that UML is structured around:

- System Functions.
- Concepts and their relationships.
- Use Cases:
    - Built with smaller functions.
    - Processed between 'Actors' and the software system.

System Functions describe the specifications of the software system and can be grouped together. They are the primary processes the software system has to perform. Concepts are abstract entities describing parts of the software system that need to interact with each other. These interactions are described in a Conceptual Model. Concepts or groups of concepts may become objects in the application. Use Cases are the features the application can perform and that the user can call from the Graphical User Interface. Use Cases are then described as a sequence of smaller functions offered by objects and 'Actors' (entities outside the software system interacting with this software system, such as a user, a database, etc.) interacting with each other through events.

C) GRAPHICAL USER INTERFACES, HUMAN-COMPUTER INTERACTION

HCI is short for Human-Computer Interaction, a discipline concerned with the study, design, construction and implementation of human-centric interactive computer systems [Webopedia (n.d.), term: HCI]. A Graphical User Interface (GUI) is a means for a human to interact with a computer. HCI goes beyond designing screens and menus that are easier to use and studies the reasoning behind building specific functionality into computers and the long-term effects that computer systems will have on humans. HCI theories and applications could also be found in Tomeski and Lazarus (1975), Norman and Drapers (1986), Baecker and Buxton (1987), Maddix (1990), Thimbledy (1990), Gorayska *et al.* (1997), Dix *et al.* (1998), and Schneiderman (1998).

HCI is a broad discipline that encompasses different specialties with different concerns regarding computer development:

- Computer science is concerned with the application design and engineering of human interfaces.
- Sociology and anthropology are concerned with the interactions between technology, work and organisation and the way that human systems and technical systems mutually adapt to each other.
- Ergonomics is concerned with the safety of computer systems and the safe limits of human cognition and sensation.
- Psychology is concerned with the cognitive processes of humans and the behaviour of users.
- Linguistics is concerned with the development of human and machine languages and the relationship between the two.

As computers become more pervasive in culture, designers are increasingly looking for ways to make interfacing with devices easier, safer and more efficient. Yvonne Rogers in Preece *et al.* (1994) states that "*we don't use computers because we want to use them. We use them because we want to do things, and computer is an enabling device that can help us to do them.*"

Ravden and Johnson (1989) state that Usability concerns the extent to which an end-user is able to carry out required tasks successfully and without difficulty, using a computer application system. Metaxas (1996), Najjar (June 1998), Squires and Preece (1999) highly recommend to have multimedia applications for Education reviewed and assessed on their usability before being released because end users are students using the computer system to learn.

Tips for good GUIs include:

- Visual clarity.
- Consistency.
- Compatibility.
- Informative feedback.
- Explicitness.
- Appropriate functionality.
- Flexibility and control.
- Error prevention and correction.
- User guidance and support.
- Learnability.
- Reliability.

Good software is not software that offers a broad selection of functions, it is software users will use because they can achieve tasks easily, and learn how to achieve other tasks implicitly without thinking and worrying. Excellent software would not need any 'Help' at all.

A GUI is the visible part of a software system. The GUI enables users to communicate directly with the computer system through peripherals:

- *User ⇒ Program:*
    - Mouse, Keyboard, Touch-screen, Scanner.
    - Data in.
- *Program ⇒ User:*
    - Visual aids: Screen (image, text, video, etc.), Printer.
    - Audio Aids: Speakers (noise, sound, music, etc.)
    - Data out.

The study of the user interface of existing software in Appendix 8.3.1 (page 266) reveals some ideas:

- Ease of use is inversely proportional to the number of available functions of a software system. Time is needed for the user to learn how to use the software efficiently when there are a high number of functions.
- Learning and training methods also depend on the users.
- Functions are usually sorted by types and accessed hierarchically in drop down menus. The most frequently used can be accessed with icons or key shortcuts. The way to access them depends on the user:
    - Some prefer to access them from the menus.
    - Some try to add icons in the toolbar to have them all in sight.
    - Others prefer not to use the mouse at all and to access them with key shortcuts.

  Functions need to be accessed from all three methods to satisfy all kinds of users.
- When a newer version is installed, interfacing habits may have to be changed. This may disturb the user, who may prefer to use the older version. Continuity has to be followed by designers through different versions even if greater functionality means changes. Software systems present usability levels to be used by any type of users.

Users may require training or a help section within the software system. A help file contains information (description, examples, etc.) on subjects or functions that can cause problems to users. Usually 'help' is accessed on the right side of a menu, or by pressing F1 key.

Wizards are special help instances enabling users to go through a long or difficult process by presenting questions or sequences, while checking and choosing options.

The following techniques could be applied to reduce the users' need for help:

- Anticipate a problem or a misuse and:
    - Tell the user or,
    - act directly to avoid the problem without disturbing the user.

- Try to help the user sequentially with wizards when multiple tasks have to be performed.
- Use context sensitive menus and controls, which are offered according to a context.
- Reduce mouse movements or have good accessibility of function through the keyboard or other means.
- Have intuitive means of accessing and performing a task.

A few comparable control systems were found that interfaced with a non-computer network. Most control systems were aimed at controlling flexible manufacturing systems as described in Ou *et al.* (June 2000).

The method used by Ou *et al.* (June 2000) was to unify the hardware interfacing, then the (distributed) software interfacing, and to define hierarchical software architecture. They stated that the use of Object Oriented Programming (see 2.2.4, page 38) was best for this type of control system software because nodes (for them machine tools) could be viewed as a collection of control objects and the integration became transparent.

The main differences between these solutions and new music technology control software were:

- *The environment* — Education / Industry.
- *The end user* — Music teacher / Engineers or other machines.
- *The controlled system* — Music technology system with Electronic Music Interfaces and new protocol / Machine tools with hardware and software interfacing and using TCP/IP for networking (see 2.2.3, page 32).

D) DATABASES

Databases (often abbreviated DB) are collections of information organised in such a way that a computer program can quickly select desired pieces of data [Webopedia (n.d.), term: Database]. Databases can be considered as electronic filing systems. Traditional databases are organised by fields, records, and files. A

field is a single piece of information; a record is one complete set of fields; and a file is a collection of records (also known as recordset or dataset).

To access information from a database, a DataBase Management System (DBMS) is needed. A DBMS is a collection of programs that enables a user to enter, organise, and select data in a database. A DBMS provides:

- Data Definition Language (DDL), permitting users to define the database.
- Data Manipulation Language (DML), letting users insert, update, delete and retrieve data from the database.

Following are the principal constituents of a DBMS [Webopedia (n.d.), term: DBMS and Whatis-Techtarget (n.d.), term: DBMS]:

- *Data dictionary*: holds the information related to the database.
- *Schema manager*: allows the design and maintenance of table structures.
- *Database engine*: is the essence of the data storing and accessing system and is hidden from normal user view. Its main role is to maintain and use indexes created for the tables.
- *Forms generator*: creates forms to interface with the tables.
- *Query system*: extracts data from tables defined by selection criteria, using query languages such as SQL (Structured Query Language), or form driven systems such as QBE (Query By Example).
- *Report generator*: produces printed or on screen reports.
- *Menu generator*: provides user-interfacing tools linking applications together.
- *Procedural language facility*: permits the execution of arbitrary procedures expressed in a high-level language for more complex process. For example, MS Access uses VBA (Visual Basic for Applications).

Entity-Relationship (E-R) modelling, or top-down approach, is an approach to semantic modelling for relationship databases, originally defined by Chen in 1976 [referenced in Dickman (1995)]. It first identifies large-scale objects in the application domain and the relationships between these objects before analysing their attributes. An E-R model contains 3 main components:

- *Entities* (will define tables).
- *Attributes* (will define fields).
- *Relationships* (between tables). Relationships must be resolved, or reduced to 'One to Many' and/or 'One to One' meaning that only one table is required. 'Many to Many' requires the creation of an intermediate table to create two 'One to Many' relationships.

The two-tier client / server architecture is a network architecture in which each computer or process on the network is either a client or a server. The three-tier architecture introduces a middle tier for the application logic. It is a special type of client / server architecture consisting of three well defined and separate processes, each running on a different platform:

- *The presentation*: runs on the user's computer or client and provides a user interface to the database.
- *The Task and rules*: govern the process of data. This middle tier may run on a server called the application server and may be multithreaded to be accessed by multiple clients.
- *The DBMS*: stores the data required by the middle tier. This tier may run on a second server (database server).

Some schools in the UK use the Student Information Management System (SIMS) database to manage school related data [students, classes, staff, etc., Capita Education Services (n.d.)]. As information stored by databases can be important and confidential, the Data Protection Act [UK government (1998)] has been put in place by the UK government to protect data from misuse.

2.2.5. Discussion

This section reviewed some suitable technologies at the time of writing concerning audio signals processing and networking. Some of these technologies were used to develop the existing music technology systems described in 2.1.4 (page 14). Audio, computing and networking technologies were used to create the new music technology system described in Chapter 3. Software technologies were used to

create user-systems interfaces, along with information systems. These later systems are detailed in Chapter 4.

Software may perform complex tasks, however programs perform predefined sequences of tasks and sub-tasks and therefore cannot be interpreted as intelligent. Artificial Intelligence (AI) may assist 'non-intelligent programs' to perform better.

2.3. Artificial Intelligence

This section refers to both Russel and Norvig (1995) and Luger and Stubblefield (2002).

2.3.1. Definition

The field of Artificial Intelligence (AI) attempts to build and understand intelligent entities (also called Agents). Definitions of AI vary along two main dimensions:

- Some are concerned with thought process and reasoning.
- Others address behaviour.

Also, these two dimensions may measure success against:

- Human performance.
- An ideal concept of intelligence.

Historically, all four approaches have been followed, using two approaches centred on humans or rationality. A human-centred approach is an empirical science, involving hypothesis and experimental confirmation. A rationalist approach involves a combination of mathematics and engineering.

2.3.2. History

AI is one of the newest disciplines, its name being defined only in 1956. However the study of intelligence is also one of the oldest disciplines:

- For over 2000 years, philosophers have tried to understand how seeing, learning, remembering and reasoning could, or should, be performed.
- From over 400 years of mathematics theories of logic, probability, decision-making, and computations have been created.
- From psychology, tools to investigate the human mind, and a scientific language to express the resulting theories have been produced.
- From linguistics, theories of the structure and meaning of language have been formed.

The advent of usable computers in the early 1950s turned the above speculations into a real experimental and theoretical discipline. The first work now recognised as AI was performed by McCulloch and Pitts in 1943 [Russel and Norvig (1995)]. Other AI works drew on three sources:

- Knowledge of the basic physiology and function of neurons in the brain
- Formal analysis of propositional logic (Russell and Whitehead).
- Turing's theory of computation.

In 1958 LISP high-level language was defined by McCarthy and was to become the dominant AI programming language. Developments in neural networks flourished in the 1960s. But, AI turned out to be more difficult than many at first imagined, and modern ideas are now richer, subtler and more interesting as a result.

From this came expert systems, which became available commercially in the early 1980s, and neural networks were revisited using new learning algorithms. It is now more common to build on existing theories than to propose brand new ones, to base claims on rigorous theorems or hard experimental evidence rather than on intuition, and to show relevance to real-world applications.

2.3.3. Fields and Applications

Different fields and applications of intelligence covered by AI were [Russel and Norvig (1995) and Luger and Stubblefield (2002)]:

- Agents.
- Knowledge and reasoning, case-base reasoning.
- Use of uncertainty:
    - Belief networks.
    - Fuzzy logic.
    - Decision networks.
- Learning:
    - Decision tree.
    - Neural networks.
    - Genetic algorithms.
    - Belief network systems.
- Communicating, Perceiving and Acting:
    - Natural language.
    - Perception.

AI covered various fields but intelligence was usually restricted to its expertise field (learn or decide, etc.) and to its expertise domain {business [Vraneš *et al.* (1996)], Korean idioms recognition [Lee and Lim (1994) and Kim *et al.* (1996)], soccer [Veloso *et al.* (1999)], sludge process [Lai and Berthouex (1992)], etc.}. Therefore, AI solutions had to be applied separately for each issue.

In the scope of this research, an appropriate research route to assist teachers with intelligent systems was considered. Two sub-projects were considered and compared:

- An intelligent assistant could help perform tasks for the teacher so that the teacher could stay focused on the lesson.
- Teaching tools could help students gain appropriate keyboard skills to follow the lessons more easily.

The applications of AI that were of interest for the research were then:

A) Agents to complete tasks intelligently.

B) Intelligent Tutoring Systems.

Both applications of AI are reviewed here and further researched in Chapter 5, after gaining feedback from teachers using the new music technology systems.

A) AGENTS TO COMPLETE TASKS INTELLIGENTLY

Intelligent agents aimed at simplifying the work of users by:

- Customising applications, which only offer or highlight the most commonly used functions or information by modelling users' actions [Tewkesbury (1994), Shapira *et al.* (1997), Kremer *et al.* (2000)].
- Performing routine, tedious and time-consuming tasks [Terveen and Murray (1996), Zalila *et al.* (1998)].
- Resolving problems and helping the user make decisions [Vivancos and Serres (1999), Volberda and Rutged (1999)].
- Anticipating problems to avoid them [Altman Klein *et al.* (1998), Corker and Pisanich (1998), or Niessen *et al.* (1999)].
- Anticipating tasks to perform them earlier or better from real time events [Wing and Flanagan (1998), Heinze *et al.* (1999)], or from users' habits [Orwant (1996), Wathieu (1997)].

The first agent function could be observed on recent Windows Operating Systems and was of little use in the new system software as functions were limited in number and already quickly accessible or always in sight.

The second function was interesting as a use was easily found. Teachers did not use the keyboard sequencer facilities as often as possible as the sequence download and upload between each keyboard and the computer took as long as one minute. An agent could achieve the task sequentially for all required stations, either on its own (feasible with keyboards with sequence request MIDI messages) or by telling the students on each station what to do to perform the required tasks from the keyboard panel. The speech to students could be performed with

available speech engines (one was included with Windows OS for its accessories / accessibility / narrator tool). Another similar task was to automatically manage emailed workpieces from students similar to the intelligent system explained in Segal and Kephart (1999). No other tasks of this kind were identified.

The third and fourth options were not considered as the tools to debug a control system crash were implemented and because the control system appeared to operate correctly from maintenance and support reports.

The last agent aim (*Anticipating tasks to perform them earlier or better from real time events*) could be of some use as observation of teaching suggested habits that could be anticipated. The sequence manager proposed in the second option was considered, as it appeared to be a sequential feature and therefore simple to implement. The emailed workpieces manager was not considered for implementation at this point, as Internet features had not yet been created.

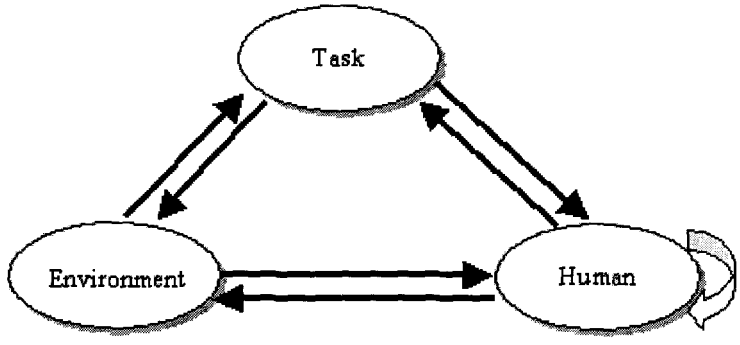


Figure 2-9: Human – Task – Environment Coupling
[from Moray (Nov 1998)]

Human - Task - Environment interactions are shown in Figure 2-9. Each of the interactions is reciprocal, with varying impact depending on the context. The human's 'self loop' indicates that the (human) teacher is able to think about the task according to the environment [Moray (Nov 1998)].

An assistant can be included to decrease the intensity of each interaction by bypassing the human (Figure 2-10). An assistant can act freely or under the supervision of the human. The more intelligent and trustful an assistant becomes, the more complex the tasks it can be given to complete.

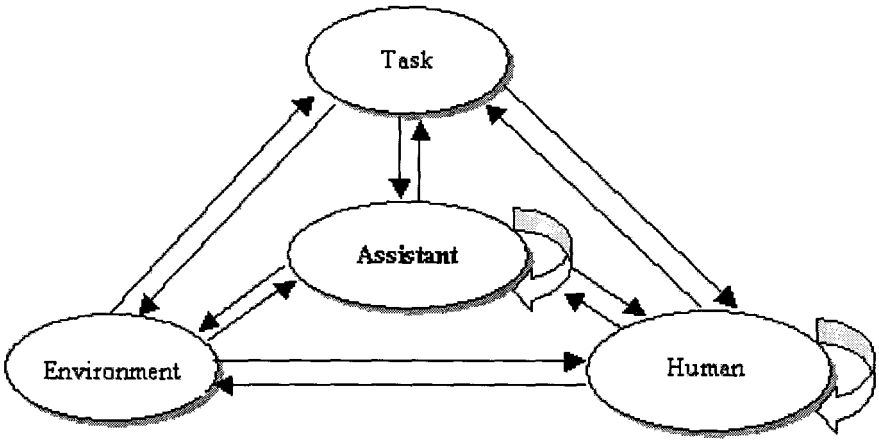


Figure 2-10: Human – Task – Environment Coupling with an Assistant [from Moray (Nov 1998)]

The assistant's actions would be limited to the Task-Human coupling as the assistant may have little interaction with the environment. This interaction could be assisted through an intelligent agent observing and learning the teacher's behaviour. After the agent had gained sufficient confidence, it would be able to anticipate the teacher's actions and offer some help.

Anticipation is the completion of a task (or a set of tasks) after a pattern of events or tasks have been detected and recognised. Recognised patterns of events' can be summarised as 'Habits'. From the user's habits a behaviour model could be defined. Different models could be set depending on habits and the context or application currently being used. For example the teacher would have to teach the same classes every day of the week. From another point of view, teachers would teach in their own style, with a habitual sequence of action. Both of these could be acquired using different methods. There were different methods for an agent to acquire a user model [Terveen and Murray (1996)]:

- End user programming (The user describes the rules for the model).
- Learning (The agent detects patterns and regularities).
- Programming by demonstration ('Macro' recording).

The first method interacted with the user and could be simple to implement and accurate if the user expresses fully the rules to be acquired for the model. The third method also interacted with the user as the user had to 'record' sequences of actions, for example macros. However the intelligent system would not be capable of capturing events to automatically launch the defined 'macros'. The second method had less interaction with the user and a slow learning curve. This could be improved with the use of stereotypical modelling [Raskutti *et al.* (1997), Shapira *et al.* (1997)], where the agent used common statements about the user's type to make quick assumptions. The means of acquiring the data and patterns, and recognising the patterns with certainty could be a problem. Research has been conducted to weigh the most recent data to calculate the degree of confidence of captured habits in order to overcome the possibility of a change of the user's habits. Other research has been carried out to handle noisy and disordered captured patterns. This has been achieved through the use of positive and negative modelling, use of inferences for collected data and context, and Neural Network learning facilities [Chan *et al.* (1995), Debar *et al.* (1992)].

A question that arose was the level of interaction the agent must have with the user. First, the agent must not disturb the user with intrusive messages or incorrect information. Second, should the user have knowledge of the model and the ability to change it? Third, which methods were best suited for acquiring and recognising specific types of habits?

The interaction with the user was important, as a way to gain knowledge about the accuracy of the predictions. The agent should give advice and information to the user, in the background, and ask question about an action, however the user should be allowed to decline answering. Information should be obvious so that the user does not need to look for it and filtered so that only necessary and useful information is given. Furthermore, different options should be offered to the user in order to improve the learning curve and usefulness of the agent [Shapira *et al.* (1997), Ruvini and Fagot (1998), Doughty *et al.* (1998), Segal and Kephart (1999)].

B) INTELLIGENT TUTORING SYSTEMS

The second sub-project dealt with teaching tools that could help students gain appropriate keyboard skills to complete the lessons more easily. Existing tools could be separated into four groups:

- Teacher lessons, where a teacher gave himself instructions.
- Keyboard Aided Instruction (KAI), where commercialised portable keyboards had embedded instructors [Casio Corporation (n.d.) and Yamaha Corporation (n.d.2)].
- Computer Aided Instruction (CAI), where software presented the theories of music and the user could interact with a MIDI keyboard connected to the computer [MiBAC (n.d.), Rising Software (n.d.1 and n.d.2)].
- Intelligent Computer Aided Instruction (ICAI) or Intelligent Tutoring System (ITS), similar to CAI but with embedded intelligence, usually to model the knowledge of the student to provide customised teaching [Stankov (1996), Molnar (1997), Lelouche (1998)].

The teaching methods used by teachers were targeted towards a specification within the National Curriculum [British Department of Education and Employment (n.d.)]. Heinemann Educational, with their New Music Matters for each key stage, became a reference for providing templates, tips and ideas for teachers to use in their lessons [Heinemann Educational (n.d.)]. The document was separated in projects, such as 'Teaching notes' or 'The music of Java and Bali'. Each project was described in detail along with its aims, resources to use for demonstration (provided with the documentation and assessments). Teachers used their own teaching method according to the National Curriculum specifications, the school resources and especially according to the students' ability and their social background. The way to deliver the knowledge and to assess students' knowledge was different in each case.

The teaching method used in KAI used pre-recorded sequences played back to the keyboard user to deliver keyboard skills. The keyboard had a broad collection of songs used for demonstrations and for the teaching.

CAI teaching methods were performed through computer software. The interaction was usually:

- *Output:*
  - Visual (Screen).
  - Sound (from the sound card and speakers).
- *Input (from the learner):*
  - Computer keyboard.
  - Mouse.
  - MIDI keyboard connected to the sound card.
  - Microphone connected to the sound card.

Available CAI software for music usually included sole assessment tools for different subjects, where the student had to answer questions (note playing, multiple answer questions, rhythm tapping, etc.) or show and practice – assess knowledge.

From Gerlič (1998), CAI test systems' strategy was one of the oldest in the use of computers in education. It was a lower strategy because it just posed questions taken from a limited pool of pre-defined questions (selection, association, sort). Traditional test systems were classical forms of tests (informal or standardised), which evaluated students' knowledge by giving test questions that were verified and evaluated. Students could practice as long as they wanted and could go to the next level if the rate of correct answers among the tests was sufficient. CAI systems could produce reports for each student and keep them in registry at the current attained level.

In CAI systems, no student model was produced to customise the teaching or for any other purposes. A student model could be of some use to deliver knowledge in a customised way to a student having variable prior knowledge and learning abilities. Student modelling was the core of ITS systems. The teaching fields delivered by existing ITS systems were:

- Languages [Kaicheng and Kekang (1997)].
- Science [Butz (1999 and 2000)].
- Process training / simulation [Andre (1997), Atolagbe and Hlupic (1997)].

Little research existed concerning music, so the research in this dissertation had to focus on customising existing techniques or on the creation of new techniques. The teaching techniques used were of several types [Self (1988)]:

- Task to be executed through a set of rules: compare and critique student knowledge against Expert knowledge (Priest T. and Young R.M).
- Use learning history (Weber G., Waloszek G. and Friedrich Wender K.):
  - Get long-term memory to compare with episodic memory.
  - Generalisation and building of analogies.
- Use student interaction [learning by doing, (Costa E., Duchenoy S and Kodratoff Y.)]:
  - Gain semantic memory (previous knowledge).
  - Get working memory (immediate understanding).
- Assess student level by assessing the knowledge transfer between the tutor and the student and check the knowledge is fully integrated to allow next step teaching. Get prerequisites to teach new knowledge (*IMPART*, Elsom-Cook M.).
- Attempt to have a student model build concepts at the same time as a student and collaborate to commonly build Knowledge (Gilmore D. and Self J.).
- Compare an expert model with a student model in order to decrease differences. Focus on learning to learn (*HUNKS*, Todd R.R.).
- Predict student actions and anticipate bugs (*FITS-2*, Woodroffe M.R.).
- Knowledge to teach defined by using scripting methods (*SCALD*, Nicolson R.).
- Define misconception and try to get number down to 0: work on remediation of bugs. Detect patterns and then try to help student (O'Shea T., Evertsz R., Hennessy S., Floyd A., Fox M. and Elsom-Cook M.).
- Diagnose errors and train to solve errors against misconceptions (*ELECTRE*, Caillot M.).
- Use experience (actions history) and record frequency of use of functions, errors detected, and number of use of help (Cooper M.).

The Knowledge assimilation checking process was common, in general, to all above methods. This process consisted of the feedback loop between the student

and the teacher/assessor, which built up the student's model. However, the checking method was different.

All above methods were implemented and tested individually or together using small applications. It appeared that Knowledge in short term memory could be managed 'On-line' as the ITS runs, with Knowledge data kept in RAM. The management of this recent or instantaneous Knowledge is performed by assessing or monitoring actions from the student, with a given task, in a determined and known context and an expected outcome. This was achieved either on the last action only or on a collection of actions showing a pattern. To manage longer term Knowledge, data was required to be stored in files or databases. Patterns for misconceptions could be retrieved 'on-line' or 'off-line' depending on required processing power.

Hunks system created by Todd R.R. [Self (1988)], was based on the ITS structure shown in Figure 2-11. This structure was described in Mc Taggart (2001):

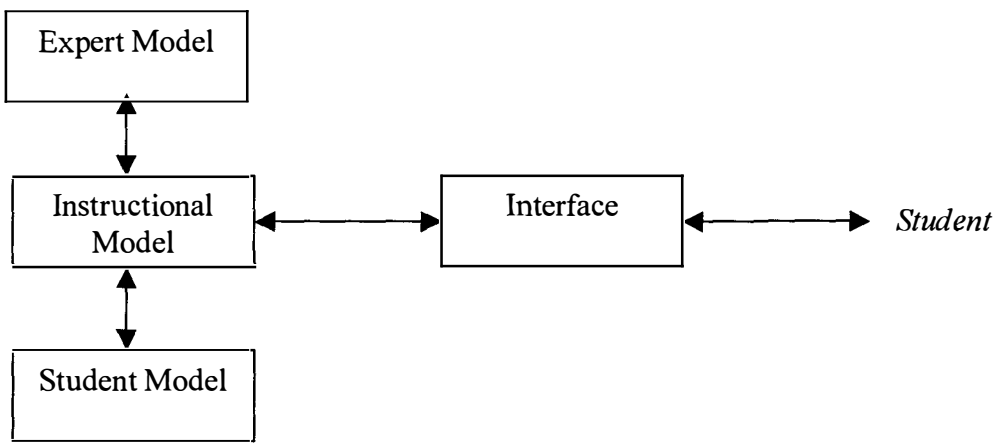


Figure 2-11: Student / Expert comparison ITS structure
[from McTaggart (2001)]

Work from Gerlič (1998) showed other interesting general templates to develop new ITS, in any field. The next three figures show three application types with increasing complexity of teaching techniques:

- An *Expert shell* interfaced with *Users* and delivered knowledge contained in a Knowledge base (see Figure 2-12). This was a CAI system with no student modelling.

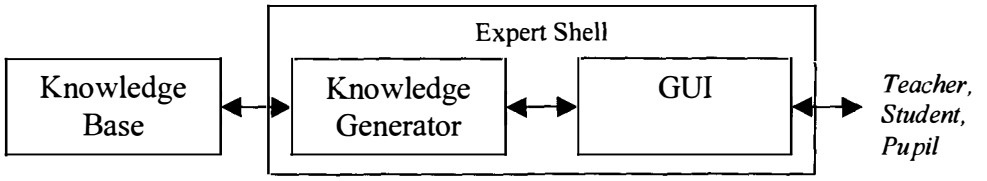


Figure 2-12: First Application Type
[from Gerlič (1998)]

- A Teaching module interfaced with a student and delivered Knowledge against defined teaching strategies (see Figure 2-13). An Expert system interfaced between the Teaching module and the Student model to build the model and enabled the customised delivery of the knowledge. An Error analysis was placed in parallel with the Teaching module and the Expert system for test assessment and bug capturing. This was a first type of ICAI system with student modelling for customised teaching.
-

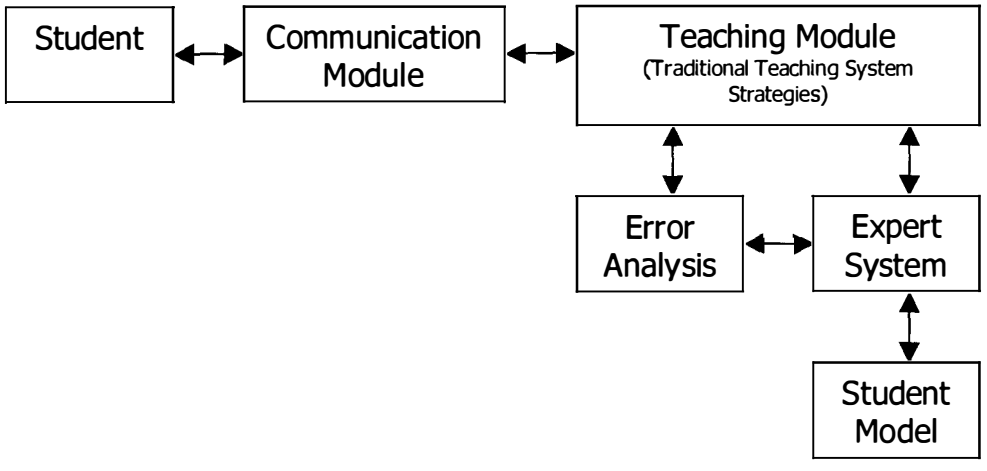


Figure 2-13: Second Application Type
[from Gerlič (1998)]

- A Student Interface module stored all information from the student in a Student Model module ('on-line' as answers and errors or 'off-line' as student's details, see Figure 2-14).

The data was then processed and used to deliver the Knowledge to the student either directly (lessons) or through an expert system for tests. This was a second type of ICAI system with student modelling for customised teaching.

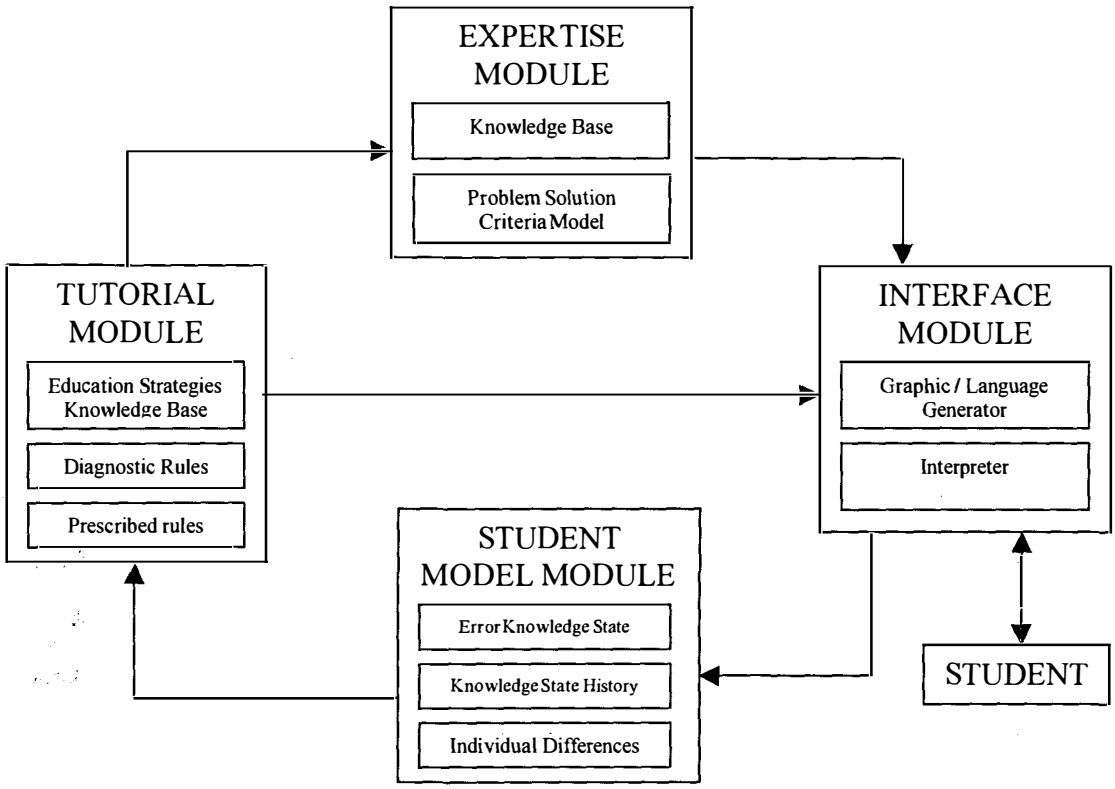


Figure 2-14: Third Application Type

[from Gerlič (1998)]

It was interesting that feedback information tended more towards the way future knowledge was to be delivered rather than the knowledge itself. The teaching strategy had thus an important role in the teaching. Furthermore, student learning abilities had to be considered as well as their misconceptions. The structure demonstrated a 'pull' teaching method, where knowledge was delivered and misconceptions checked for and 'debugged'. Such ICAI systems were 'issue-centred' and called 'diagnostic ICAI system' [AECT (2001)].

"*When assessment is at its best, it can be motivating and productive for students, helping them know how well they are doing and what else they need to do*" [Brown (1999)]. The direct relationship and inseparability of assessment, feedback and learning was highlighted in the QAA principle "*Institutions should ensure that appropriate feedback was provided to students on assessed work in a way that promoted learning and facilitated improvement*" [QAA, General principle 12, in Rust (2002)]. In order for feedback to promote learning, students had to actively engage with the feedback [MacLellan (2001), Rust (2002), Taras (2002)].

The engineering of feedback to engage students in learning (assessment for learning, as separate from assessment of learning [Elwood and Klenowski (2002)] should be included in lesson design.

Regular and integrated use of assessment as part of teaching also reduced the threat of assessment. It was particularly important, in terms of retention of students from non-traditional backgrounds, that the assessment system used in their first years should be non-threatening [Rust (2002)]. If the strategic goal was to use feedback as part of teaching to promote learning, then it followed that the feedback should be related, in some way, to module assessment. Interfacing with the student had to be carefully considered as to keep motivation and self esteem high to get better results. A distressed student would never learn, would build misconceptions or would just not want to use the ICAI system.

This teaching method is also described by Wilson and Cole (1996), where assessment used for 'learning by doing' small chunks of knowledge could help assimilate the knowledge. This 'Cognitive Load Theory' enables the working (short term) memory to transfer better to medium term memory. This 'push' method (compared to the 'pull' method used by diagnostic ITSs) emphasises the knowledge understanding rather than focus on misconceptions. Students are asked here to think and try to make relationships between these small chunks of knowledge.

Cavallo et al (n.d.1) describe that teachers at 1-teacher schools can have stronger, fuller, and deeper relationships with their students, and the students can be more autonomous and in control of their own learning while still more supportive of and cooperative with their colleagues. Technology can provide connectivity, thereby eliminating some of the resource constraints. Cavallo et al (n.d.2) also describe "Constructionism" as a theory of learning and education. Constructionism is based on two different senses of "construction." It is grounded in the idea that people learn by actively constructing new knowledge, rather than by having information "poured" into their heads.

In fact, Mergel (1998) details three teaching theories. These are:

- *Behaviourism:* Based on observable changes in behaviour. Behaviourism focuses on a new behavioural pattern being repeated until it becomes automatic.
- *Cognitivism:* Based on the thought process behind the behaviour. Changes in behaviour are observed, and used as indicators as to what is happening inside the learner's mind.
- *Constructivism (Constructionism):* Based on the premise that we all construct our own perspective of the world, through individual experiences and schema. Constructivism focuses on preparing the learner to problem solve in ambiguous situations.

A 'Push' method based on the 'Cognitive Load' and 'Constructionism' theories was used to construct the new ITS as it was thought as better suited for the students. This method emphasised on the potential of teaching/learning interactions rather than issues used by other diagnostic ITSs. Furthermore, as shown in Figure 2-15, this method related to the O'Shea et al five ring model, which could be implemented on existing CAI systems. Another ITS structure than Gerlič's (shown in Figure 2-14 on page 58) was required to represent this method. However, Gerlič's structure was used to compare and contrast both methods. Jones (1992), describes the spectrum of methodologies used in existing ITSs. This is shown in Figure 2-15, with the description of the methods.

Exploratory learning environments: requires the student to move around knowledge bases of the knowledge domain and are best suited for teaching abstract and general concepts such as use of analogies and model building.
ACT principle: Its main concern is deviation of the student from the ideal student model. With a secondary concern of looking out for common known mistakes that a novice is likely to make. After this it will then try to provide corrective information to steer the student back on to the ideal path.
Hartley and Sleeman method: it does not give misconceptions in the domain (the bug catalogue) primary importance, but instead it uses the student model as the primary component. This student model is a model of the student's performance throughout the tutorial and possibly in other tutorials. It is more tutoring oriented. The interaction provided by the user interface is controlled by the ITS. The MEGRs use information held about the student in the student model to choose which of the possible teaching operations the ITS should present the student with next.
O'Shea et al five ring model: it gives primary importance to the student model. It also gives much importance to the teaching strategy. It is also possible to buy a tool kit for building ITSs that use the five ring model in much the same way that you can use authoring systems to develop traditional CAI.
Traditional CAI: emphasises on teaching strategy rather than representation of the knowledge

Figure 2-15: ITS methodologies spectrum [from Jones (1992)]

Finally Du Boulay and Luckin (2001) emphasise the use of motivation in the teaching if ITSs are to be useful (conclusion drawn after the discussion made by Ohlsson in 1987 on the use of early ITSs). Buxton *et Al.* (n.d.) also stress the need to use motivation as used in their ActIPret project at COGS (University of Sussex, UK). The ICAI system provides feedback to motivate the trainee and to enhance the training effect. This results in a superior training effect compared to repetition without feedback.

2.4. Chapter Discussion

This chapter reviewed:

- How music education was performed in the UK.
- Technologies that were useful in the creation of the new music technology system.
- Application of Artificial Intelligence (AI) to non-intelligent systems, in the scope of two sub-projects that could further assist music teachers .

The new music technology systems described in Chapters 3 and 4 implemented some of the reviewed technologies. Feedback on the impact of the new systems showed that AI was required to assist music teachers with the new systems. The implementation of AI is described in Chapters 5 and 6.

CHAPTER 3. HARDWARE AND FIRMWARE

Music technology systems offered an infrastructure where audio and music devices could be monitored and controlled (see 2.1.4, page 14). However, the available music technology systems were limited in their scope as they were inflexible, difficult to upgrade and sometimes of little help to their users. Furthermore, linking new technologies and digital devices to these music technology systems might have been difficult.

A new music technology system was created with the help of music teachers. This new music technology system, called a Keyboard And Audio Network (KAAN) was created in collaboration with the University of Portsmouth and Counterpoint MTC Ltd based in Worthing (West Sussex, UK). Industrial design methods and standards were used to create this new music technology system. Because the collaborating company supplied music and IT equipment to Education and the company dealt solely with the Education market, it was easy to meet with teachers / customers. These were interviewed and observed to design and test the new music technology systems.

This chapter describes the creation of the KAAN system:

- The specifications for this new music technology system were derived from requirements gathered from teachers, students, music advisors, and the collaborating company.
- Methods to network audio signals from students' stations for grouping and recording were investigated. This needed to be defined before the rest of the music technology system design because audio management depended on the selected method. New electronic interfaces were then considered.
- The creation of new Electronic Music Interfaces (EMIs), are described.

- Ways to control the new EMIs and the creation of a new communication protocol is discussed.
- Music technology system installation and maintenance is discussed.
- A chapter conclusion confirms that specification criteria were met and discusses potential future work for this new music technology system.

Note that the author graduated in mechanical engineering and assistance to gain knowledge on electronics was needed. This help was provided by Dr Giles Tewkesbury with design, test and debugging of some of the sub-systems described in this chapter. Dr Giles Tewkesbury also coded the entire source of the microcontroller used in the music technology system. All work that was performed by Dr Giles Tewkesbury is indicated as such.

3.1. Specifications

Specifications for a new music technology system had to be defined. Requirements were gathered from different sources. The sources were:

- *End User*: Teachers.
- *Market Research*: Collaborative company marketing department.
- *Journals:* Music Advisors

Requirements were then grouped together to form specifications.

3.1.1. Requirements

The requirements of teachers were first gathered from papers written by English Music Advisors (as seen in 2.1.3, page 12). These requirements were then confirmed and refined with information from local music teachers. This was gathered by the collaborative company marketing and support departments (one member of staff was also a Music Teaching Advisor). The requirements are shown in Table 3-1.

After comparing the music technology systems available (see 2.1.4, page 14), some features were found to be useful and others limiting. These features were:

- *Useful:*
  - Teacher / Students interaction (listen, speak, play or record) from the teacher's desk.
  - Students could be grouped for collaborative work.
- *Limiting:*
  - Inadequate Flexibility and No Extra Inputs. Music technology systems only use specific equipment or in pre-determined configurations. The new music technology system had to allow for any configuration and equipment.
  - Inability to share Recording in multiple formats between audio equipments. Some music technology systems just allowed recording on tape or MIDI files, unless a computer was connected as a separate element to the music technology system. In that case, work could be recorded digitally as MIDI, wave, mp3 or wma files (see pages 20 and 33).
  - Poor Audio Performance (hiss, interferences and cross-talk). This was disturbing and tiring after some time. The new music technology system had to manage high audio quality.

R1: Two students playing independently on each keyboard without hearing each other (and thus not being disturbed).

R2: A fully controllable student station to improve discipline for:
- **R2a**: both stand-alone play (disabling the keyboard's panel, so that students can focus on the lesson).
- **R2b:** and remote usage (such as muting, listening to students, etc. from the teaching position).

R3: Any MIDI Keyboard so that teachers can reuse their keyboards. The selected MIDI portable keyboards were to feature full size keys (on 5 octaves), feature 'touch sensitive' and have a built-in sequencer (to allow at least 4 track sequencing). These keyboards were inexpensive. Methods to disable the 'Demo' function had to be investigated for each of them.

R4: Good audio quality for comfort and recording (e.g., no recognizable hiss, interferences or cross-talk).

R5: Easy shareable recording material on tape or any digital medium (especially with computers as MIDI, wave, mp3 or wma files).

R6: Extra stereo audio inputs for microphones or other instruments.

R7: Stations could be grouped in multiple groups (pairs up to the whole system).

R8: A flexible and upgradeable system. The system could be mounted under desks or on the wall, in a main classroom or practice rooms. The system could also be installed in a mobile way with 'pods' easy-to-connect to the main system.

R9: More advanced control from the teaching position using a control device (such as a PC) to enhance system functionality.

Table 3-1: Hardware Requirements gathered by the author in 1998 from multiple sources (Teachers, journals, company)

3.1.2. Specifications

After testing existing music technology systems, it was found that audio quality problems (R4) occurred because analogue audio signals were processed at the teaching position. Audio signals were capturing noise between the students' stations and the teacher's console. Furthermore, certain music technology systems used a power network to reduce the number of power supplies required to power up students' keyboard. Power was converted in nodes and used for attached devices and internal circuitry. In such music technology systems, audio signals 'jumped' on the power network, resulting in students hearing other students.

Either digital technology could be used to overcome this issue or analogue signals could be processed locally. In both cases, electronic interfaces (or nodes) at each desk were required to either convert analogue to digital and vice-versa at the station or to procure the audio processing facility. A power network could be considered if the power conversion circuitry was carefully designed. Also, these nodes required a microcontroller (see 2.2.2, page 22) for control, communication and potential setting storage.

Specifications could be defined from the requirements stated earlier and these last comments. These specifications would set the work to be achieved and would need to be verified at the end of the design. Specifications are shown in Table 3-2.

S1: To process MIDI messages from a portable keyboard in real time, to split it and to control it. (**R1, 2a, 3**). The split modes (as shown in Figure 3-1) would consists of:

- ***Solo***: The keyboard was set to use all 5 octaves.
- ***Split***: The keyboard was split into two 2-octave keyboards, with each student only listening to what he was playing.
- ***Duet***: Similar to split mode, but each student could listen to the other student but at a lower volume.

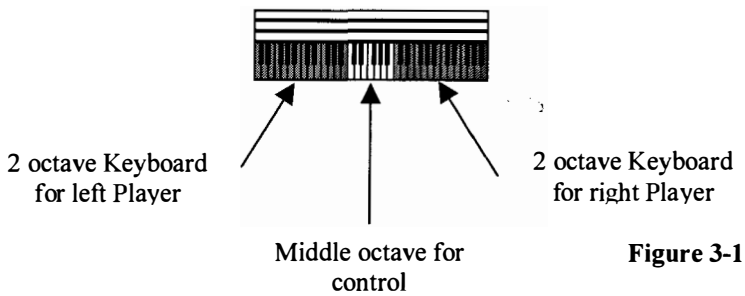


Figure 3-1: Keyboard Splitting Features

S2: To process audio locally to improve audio quality and to dispatch the resulting audio signal to the appropriate student. **(R1, 2a, 3, 4)**

S3: Transmit and receive audio (in an analogue or digital form) to or from an audio network. To allow stations to be grouped together (from a central console or locally). Nodes to be used stand alone or networked and remotely controllable. **(R2, 3, 5, 6, 7, 8, 9, 10)**

S4: To be controllable remotely (from a computer or another controller) using the existing network(s) or another digital network. **(R2b, 5, 8, 9)**

S5: To supply power to peripheral devices (portable keyboard and / or mixer) to reduce the number of power supplies. A power network was considered. A clean power supply without 'cross talk' was essential. **(R3, 4)**

S6: To use a daisy-chained structure (like MIDI networking, see 2.2.3, page 32) to facilitate installation and maintenance. The comparison of the different network structures and the music systems detailed in 2.1.4 (page 14) showed that many long cables were difficult to install and maintain, and could pick up noise. Maintenance with shorter cables was made easier if a node did not work properly, as the nodes after the faulty node would not work properly either. **(R8, 9)**

S7: To network 16, or more, stations as classes of approximately thirty students could be asked to play together on the same keyboard(s). **(R1, 8)**

Table 3-2: Hardware Specifications defined from Requirements (1998)

From the specifications, a basic design of a new music technology system was created as shown in Figure 3-2. Stations were composed of nodes and keyboards (or other instruments) for two students, which were connected together on an audio and control network. The network was linked with a control device (for example a computer), on which a teacher could interact with students.

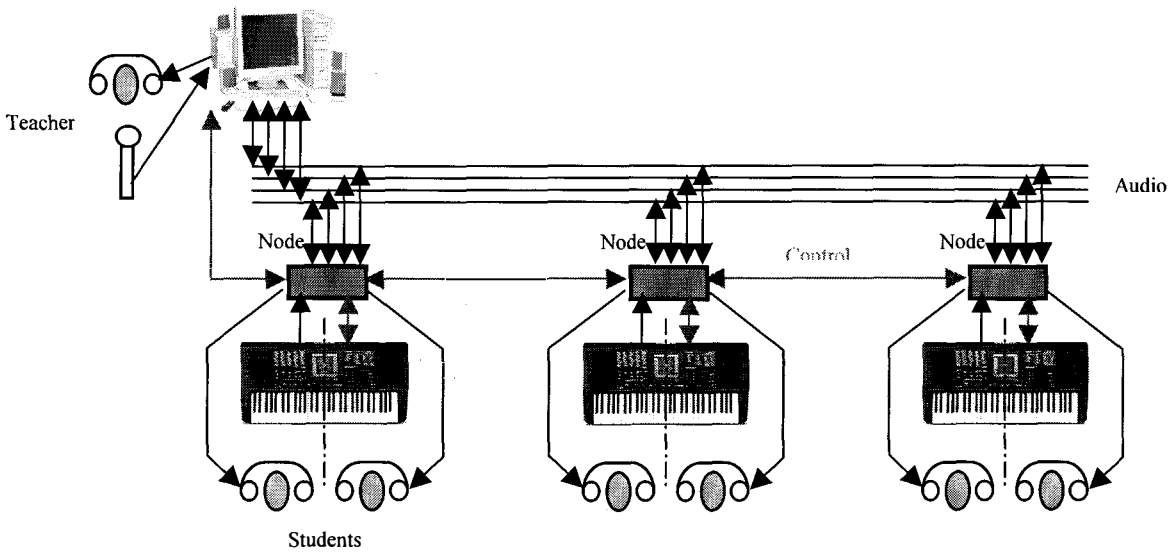


Figure 3-2: New Music Technology System Basic Design (1998)

Specification **S3** determined how the music technology system structure had to be defined. From 2.2.1 (page 18), many technologies existed to network and process audio. The control device had to be incorporated on the network as well. The next section discusses four different designs that could have been used for the new music technology system.

3.2. Design investigation

Audio network design and audio transmission and processing was investigated and tested. Audio signals had to be used for grouping, and for a teacher to listen, speak or record any student. Additionally, the control network and methods to remotely control the nodes were researched.

3.2.1. Overview of other music systems

As seen in 2.1.4 (page 14), both analogue and digital technologies (also see 2.2.1, page 16) were used in the several music systems described. For all analogue and some digital systems the grouping and control was achieved at the front console to reduce the number of student electronic interfaces. The other digital systems used the functionality of their audio protocol (such as IEEE 1394 (Firewire),) to achieve the grouping.

The aim of this investigation was to compare different technologies that could be used for the new music technology system against the following three criteria:

- Simplicity.
- Cost.
- Development time.

Four designs were considered and some tested:

- Parallel Network Buses.
- Analogue Audio Sampling.
- Networked Digital Audio.
- Analogue Audio Driving / Receiving.

The following aspects were considered to compare the four designs:

A) Audio processing.

B) Control.

C) Features.

Features (simplicity, cost, development time) were graded by the author, from his design experience and results from prototyping. Grades varied between 1 and 4, 1 being minimum and 4 maximum, without medium grade to show a preference.

3.2.2. Design 1: Parallel Network Buses

The sub-system was tested successfully. Audio quality could be an issue as long ribbon cables might act as an antenna.

A) AUDIO PROCESSING (ANALOGUE)

As shown in Figure 3-3, a 32-way bus was used with cross point switches to allow players to listen to each other. The node picked up and continuously transmitted mixed audio signals to and from a 32-bus network. Any node could listen to any of the other nodes independently.

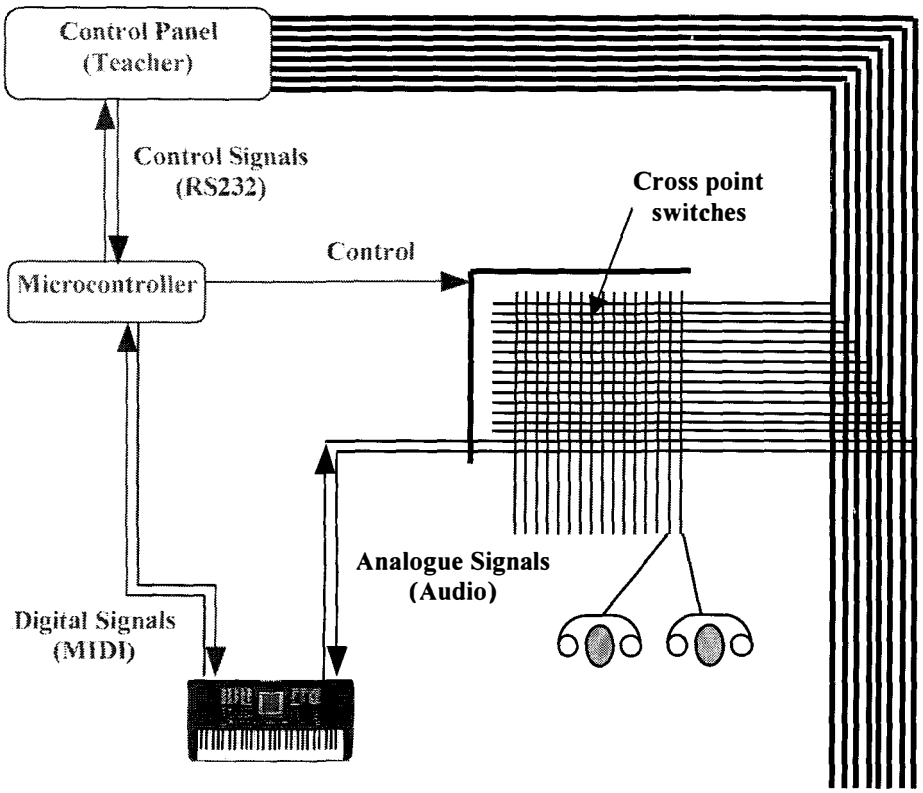


Figure 3-3: Design 1: Parallel Network Buses (Tewkesbury, 1999)

B) CONTROL

The digital control data could be transmitted via a RS 232 (serial) port or on one of the unused ribbon cable buses.

C) FEATURES

Design 1 was simple and quick to develop. Also many groups could be created locally. Cost would be reduced depending on the cost of the ribbon cable and the cross point switches. Audio quality could be an issue in noisy environments as long ribbon cables acted as an antenna. Two networks for audio and control were required. This was because control data carried with audio on ribbon cables would have created a lot of interferences. Two networks would make the cabling cumbersome.

The features are summarised in Table 3-3.

	Simplicity	Cost	Development Time
Design 1	4	1, depended on the ribbon cable and cross point switches	1

Table 3-3: Design features summary for Design 1 (1999)

3.2.3. Design 2: Analogue Audio Sampling

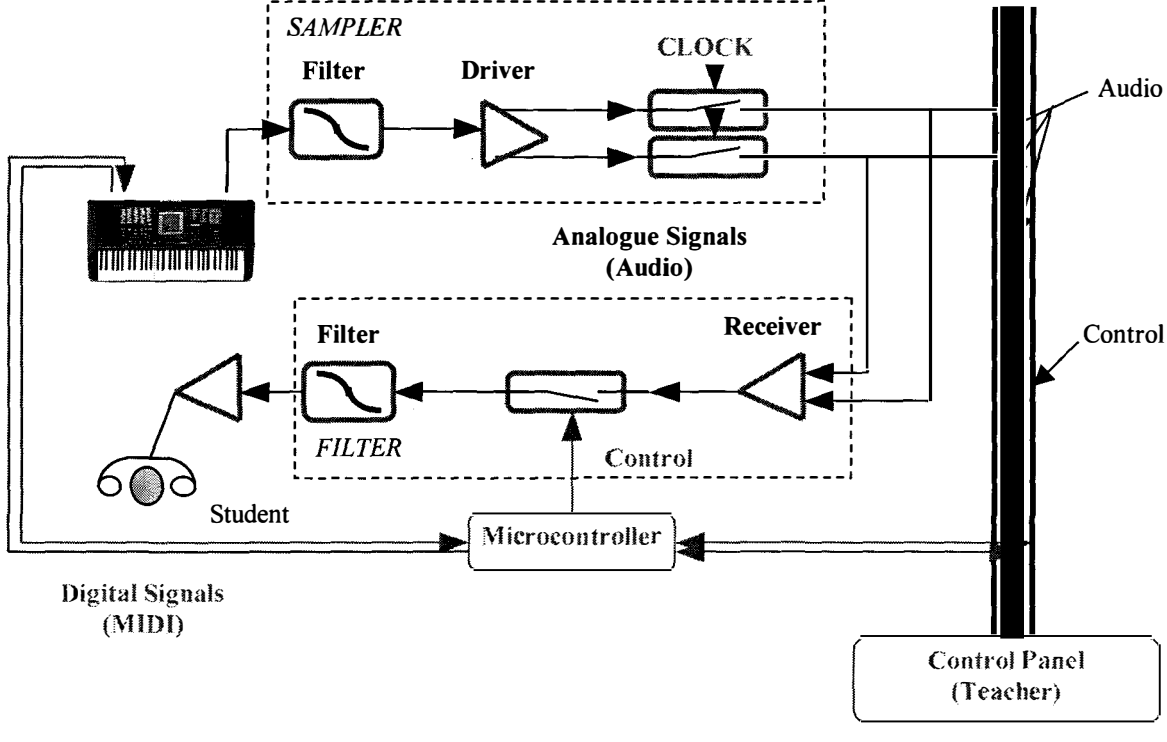


Figure 3-4: Design 2: Analogue Audio Sampling (Tewkesbury, 1999)

The method was tested to check its feasibility. The test consisted of sampling first a sine wave then a mono audio signal. The signal was sampled ($1/32^{nd}$) and filtered at the other end to retrieve the correct signal. This signal was checked on an oscilloscope or audibly for distortion and clipping. The test was successful but the testing system was found to be complicated and expensive. Sending MIDI messages using another time slot was considered in order to record students' work in a MIDI file.

A) AUDIO (SAMPLED ANALOGUE)

The sub-system shown in Figure 3-4 used an analogue sampling method for the audio signals called Time Division Multiplexing. This was where each node was given an address and a defined time slot to transmit its analogue audio signals onto the network (see 2.2.1, page 18). To listen to a determined node, the listening node polled the network during the time slot of the transmitting node. The resulting analogue signal passed through a low-pass filter to be 'smoothed'. To group stations together, the grouped nodes needed to listen to the network during the time slots of all the grouped nodes, resulting in a 'virtual' group. The audio network used standard 4 pair twisted cables (see 2.2.3, page 30).

B) CONTROL

The digital control data could be transmitted via a RS 232 (serial) port or one pair of the 4 pair twisted cable.

C) FEATURES

Design 2 is more difficult and required more time than design 1 as it requires more components and sampling technology. Required components could be expensive and were estimated at two or three fold the cost of the Design 1. The sampling of one signal was successful and the sampling of more signals was expected to work. Audio quality needed to be checked with different types of signal and with long cables. For audio and control networking two media could be used. However, 4 pair twisted cables could theoretically bring audio and data on

different pairs without causing interferences. This would incur less cabling than Design 1. Many groups were available.

The features are summarised in Table 3-4.

	Simplicity	Cost	Development Time
Design 2	2	3	3

Table 3-4: Design features summary for Design 2 (1999)

3.2.4. Design 3: Networked Digital Audio

This design was not tested because of the potential cost for building and testing a prototype.

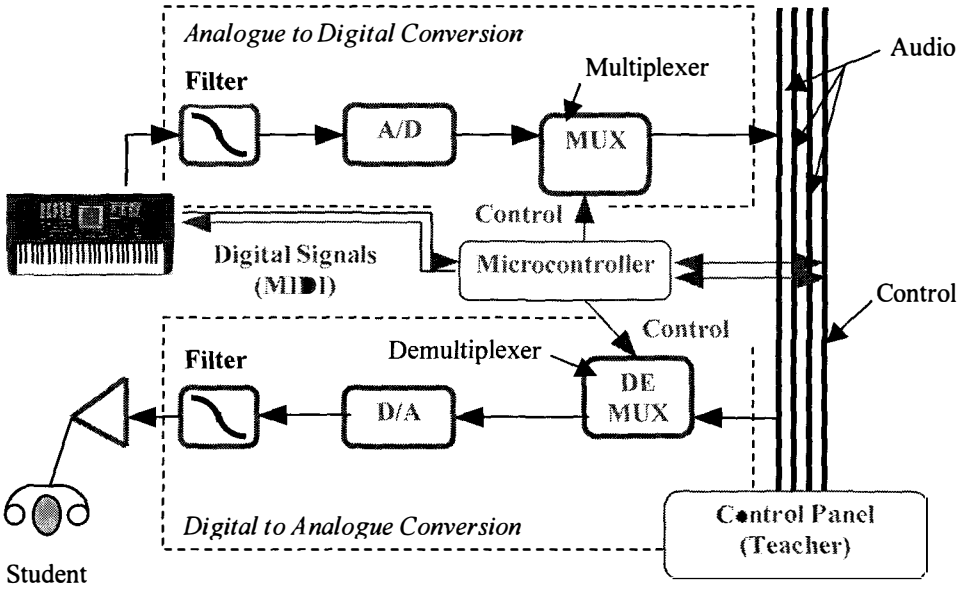


Figure 3-5:Design 3: Networked Digital Audio (Tewkesbury, 1999)

A) AUDIO (DIGITAL)

As shown in Figure 3-5, analogue audio signals were converted into digital audio signals. Digital audio and control signals could be transmitted using standard 4 pair twisted cables and a new protocol. Grouped stations would mix re-converted analogue signals from all grouped stations (see 2.2.1, page 20).

B) CONTROL

The digital control data could be transmitted via an RS 232 (serial) port or one pair of a 4 pair twisted cable.

C) FEATURES

Design 3 was more difficult than Design 1, but easier than Design 2 as standard Analogue to Digital (A/D) and Digital to Analogue (D/A) converters could be used. However, it would still take time to develop. Digital converters and components were expensive and the cost of this design was estimated at three or four fold the cost of Design 1. Audio quality would be excellent as digital audio was processed. Like Design 2, two or just one network could be envisaged, as there were only digital data. This would lead to less cabling than Design 1. Like Design 2, many groups ('virtual') were available.

The features are summarised in Table 3-5.

	Simplicity	Cost	Development Time
Design 3	1	4	4

Table 3-5: Design features summary for Design 3 (1999)

3.2.5. Design 4: Analogue Audio Driving/Receiving

The method was tested using two Jack2Net interfaces as shown in Figure 3-6. Sinusoidal and mono audio signals were input on 1 side and then balanced and driven to one bus of a standard 4 twisted pair cable. The signal was then received by another interface and mixed with another signal (sinusoidal or mono audio signals). This enabled to check that audio could be networked and processed at the other end with local audio signals. The resulting signal was checked with an oscilloscope and audibly for distortion and clipping. The test was successful.

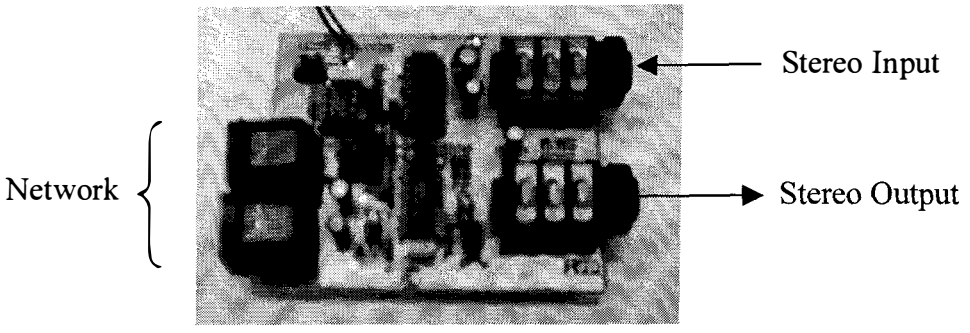


Figure 3-6: Jack2Net Interface (Tewkesbury, 1999)

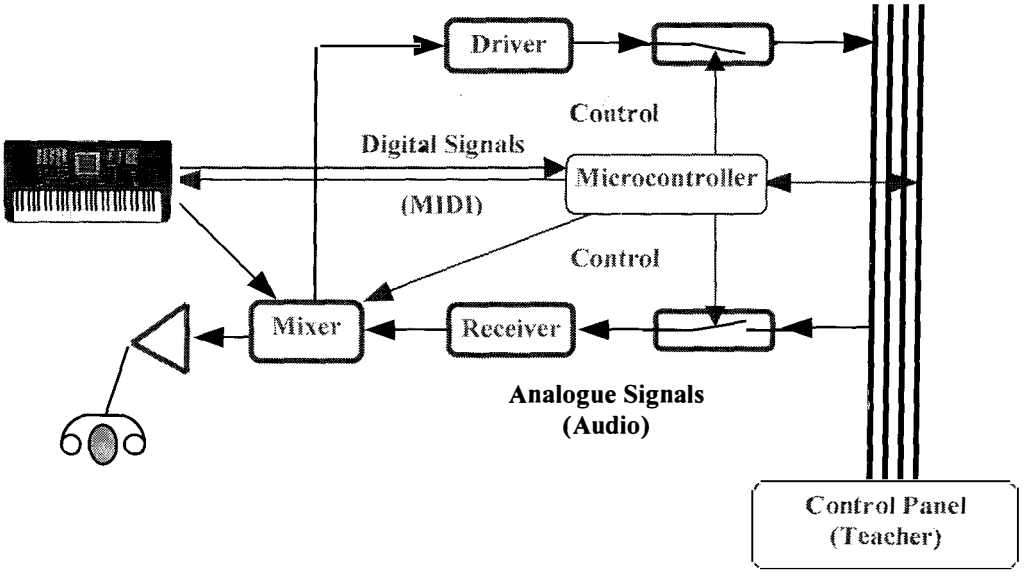


Figure 3-7: Design 4: Analogue Audio Driving/Receiving (Tewkesbury, 1999)

A) AUDIO (ANALOGUE)

As shown in Figure 3-7, mixed audio signals from the keyboard, auxiliary inputs and received networked audio signals could be transmitted onto a maximum of four buses using audio line drivers / receivers and standard 4 pair twisted cables. To listen to another node, both nodes were 'attached' to a bus and would drive / receive analogue audio signals onto or from it. The networked signal was then mixed with local inputs before being dispatched to the correct outputs (local or networked).

B) CONTROL

The digital control data could be transmitted via an RS 232 (serial) port or one pair of a 4 pair twisted cable.

C) FEATURES

Design 4 was simple and could be developed quickly. Required components were relatively cheap, in the same order as the Design 1. Audio quality needed to be checked with multiple mixed signals in a noisy environment. Like Designs 2 and 3 two or just one network was needed, which will lead to a small amount of cabling. However, only three or four groups were available and a method had to be found to produce more groups.

The features are summarised in Table 3-6.

	Simplicity	Cost	Development Time
Design 4	3	2	2

Table 3-6: Design features summary for Design 4 (1999)

3.2.6. Comparison and Discussion

Four different types of audio networking designs were envisaged and some tested. These were considered against the three criteria of:

- Simplicity.
- Cost.
- Development Time.

Table 3-7 outlines the results of this consideration (Grades set between 1 and 4, 1 being minimum and 4 maximum).

Design 4 was selected as it offered the best features (cheap and simplicity) and fewest drawbacks (bus number). Design 3 could be reconsidered for future work if new protocols using the IEEE 1394 (Firewire) standard emerged.

The new KAAN system was created based on design 4 with analogue audio drivers and receivers to provide networked audio signals. Remote control was achieved by using a new customised communication protocol. Both audio and control networks used standard 4 way twisted pair cables. The creation of the new nodes consisted of two phases:

- Hardware (Electronic circuitry and metal enclosure designs).
- Firmware (Control protocol definition and microcontroller program development).

	Simplicity	**Cost**	**Development Time**
Design 1	4	1, depended on the ribbon cable and cross point switches	1
Design 2	2	3	3
Design 3	1	4	4
Design 4	3	2	2
	Advantages	**Drawbacks**	**Equipment**
Design 1	Many groups Cheap Simple	Wiring / Cabling ⇒ Maintenance Audio Quality?	Cabling, cross point array, microcontroller, amplifier/filter
Design 2	Less cabling	Difficult ⇒ needed experiments Many components ⇒ expensive larger nodes? Audio Quality?	Microcontroller, sampling, filter, amplifier
Design 3	Less cabling Digital Audio	Difficult ⇒ needed experiments Many components ⇒ expensive big nodes?	microcontroller, sampling, filter, DAC/CAD, amplifier
Design 4	Less cabling Simple Cheap	Audio Quality? Grouping?	microcontroller, amplifier/filter

Table 3-7: Designs comparison (1999)

3.3. Hardware

Electronic interfaces were required to control and manage audio in the new Kaan system as the system had to allow stand-alone operation without a control device (**specifications S2 and S3**). The new electronic interfaces used Design 4 for the audio and control network side of the music technology system.

3.3.1. Electronic Music Interface (EMI)

The new KAAN system consisted of a network of new electronic interfaces called Electronic Music Interfaces [EMIs, as described in Lassauniere *et al.* (1999a)]. EMIs could be connected to a MIDI portable keyboard and/or any other instrument. Each EMI could work stand-alone or be remotely controlled from a control device at the teacher's desk.

The role of the EMI was to:

- Supply power. (**Specifications S5, S6 and S7**)
- Process Audio. (**Specifications S2 and S3**)
- Process MIDI. (**Specification S1**)
- Communicate via networks. (**Specifications S4, S6 and S7**)

These roles could be described as function blocks, which could interact with each other as shown in Figure 3-8:

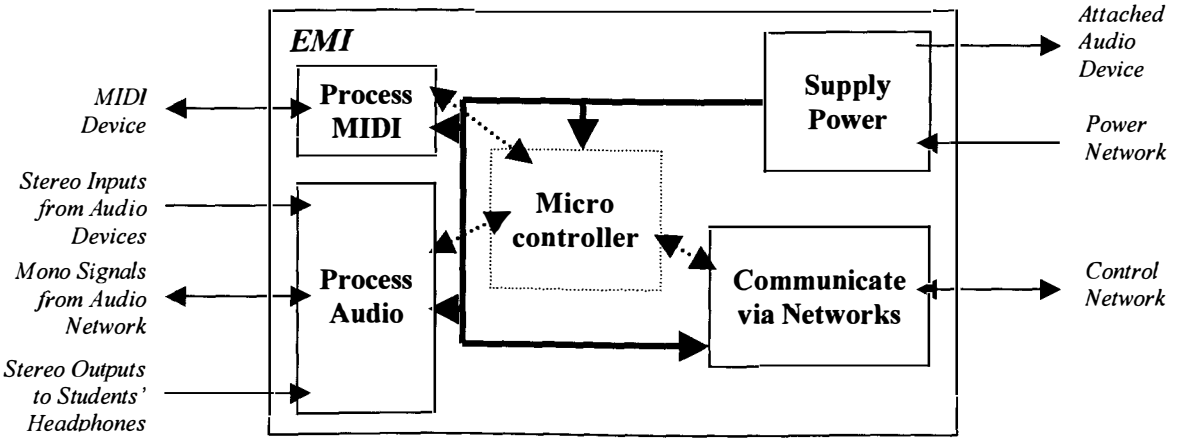


Figure 3-8: EMI Block diagram (with blocks interactions, 1999)

Having defined the entire specification of the new KAAN system and the system EMIs, the creation of the new hardware could be considered. The new hardware was split into three main function blocks:

- Power Supply.
- Audio (networking and processing).
- Microcontroller (control and communication).

The three main blocks were tested separately and then together. All calculations can be found in Appendices 8.2, page 237. The design was performed on a Ranger XL CAD software package from Seetrax (n.d.). This software allowed designing of the circuit in order to place and check components and tracks on a Printed Circuit Board (PCB). The software could then export the design in any standard format to build and populate the PCB.

3.3.2. Power Supply for the EMIs

Power was networked using a single dc power supply and a daisy-chained network (Specification **S5**). First, the voltage supplied had to be defined against the capability of dc converters needed to convert this input voltage to appropriate levels. It was required that the power network could withstand a high current as, in a daisy-chained network, the current drawn by each EMI would be added (Kirchoff's law).

All portable keyboards could be powered with 10V dc. The network voltage needed to be above this value because the dc converters required a minimal voltage of 12V and a maximum of 30V. The default voltage was defined at 13.8V dc (standard power supply voltage) or 13.8V dc. The development of the power supply block required the transformation of the networked 15V dc into different voltages (as shown in Figure 3-9):

- +10V (1A max) to power up a portable keyboard and/or a mixer.
- +5V for the internal circuitry.
- 0V Ground.

- Clean ground for audio. In general a clean reference for the audio side was required as cross talk found on other audio systems could be generated by audio signal travelling on the power lines.
- -5V for audio circuitry.

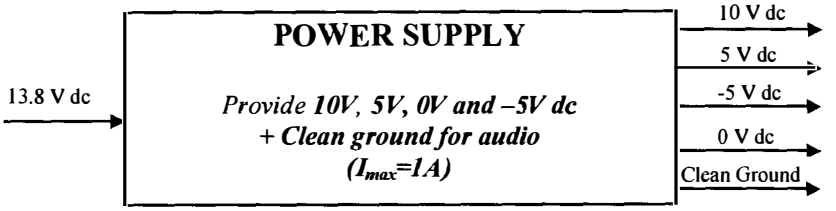


Figure 3-9: Power Supply Functional Diagram (1999)

POWER TO AUDIO DEVICES: To convert to +10V dc for the Keyboard, a LM2940T10 (10V dc regulator, 1A) was used since a keyboard could draw as much as 750mA. A diode prevented inverting polarities and a 1A fuse was used to avoid overloading. This part was kept independent to the internal power supply to the internal circuitry, in order to avoid cross talk. Two capacitors were added to filter any noise from the networked power.

POWER FOR INTERNAL CIRCUITRY: To produce a high current negative 5V for the audio, a virtual ground of +5V was used. Thus:

- 0V was +5V.
- +5V was +10V.
- –5V was 0V.

For this a LM2940T10 dc converter was used to convert the input network voltage to 10V and a 7905 dc converter to convert to +5V (The 7905 was a -5V regulator). A diode prevented inverting polarities and a 500mA anti surge-fuse was used to avoid overloading (chosen from observed values). As another security, a 5V Zener diode was added between the virtual ground and +5V to control the difference of potential between these two lines. Two capacitors were added to filter any noise from the networked power. Every Integrated Circuit (IC) was powered between the virtual ground and +5V (real +5V and +10V) with appropriate decoupling capacitors.

CLEAN GROUND FOR AUDIO: Two types of op-amps were used in the circuitry: standard and headphones op-amps. Standard ones were used for power and headphones for audio only. A clean ground was required for audio to reduce noise. This clean ground was generated from the unused +5V phantom voltage level generated by the selected headphones op-amp.

HEAT DISSIPATION: A heat sink dissipated heat from the three voltage regulators (2 * LM2940T10 + 1 * 7905). The heat sink dimensioning is outlined in Appendix 8.2.1.

SPECIAL CONNECTORS FOR NETWORKED POWER (LOW VOLTAGE – HIGH CURRENT): Special connectors were needed for the networked power leads to withstand the high ampere rate driven on the power network. This was caused by the addition of the current drawn by each EMI on the daisy chained power network. Portable keyboards powered from an EMI could draw up to 750mA. If a value of 1A was drawn by each EMI, a network of 32 EMIs could draw as much as 32A. Amp's Mate n lock connectors (PCB socket and cable plug with crimps) were found to be the best:

- As the only connector to withstand up to 19A.
- As being indexed not to invert polarity.
- For safety as no contact was visible (particularity interesting in a classroom environment).

Constraint 1: Max current in power network = 19A, with safety coefficient of 1.5: reduced to **13A.**

The EMI had two PCB connectors to allow the daisy chaining of the power network. A reinforced power track between the two connectors was designed to reduce the resistance of this track, thus reducing heat dissipation. The track was tested on one EMI with low resistance power resistors (dimensioned in Appendix 8.2.1, page 237) and a high capacity power supply.

The test was carried out twice for 8 hours with the test equipment shown in Figure 3-10. The voltage at a resistor was measured to determine the real ampere rate, 4.18V: 2 * 4.18 / 0.47 = 17.8A

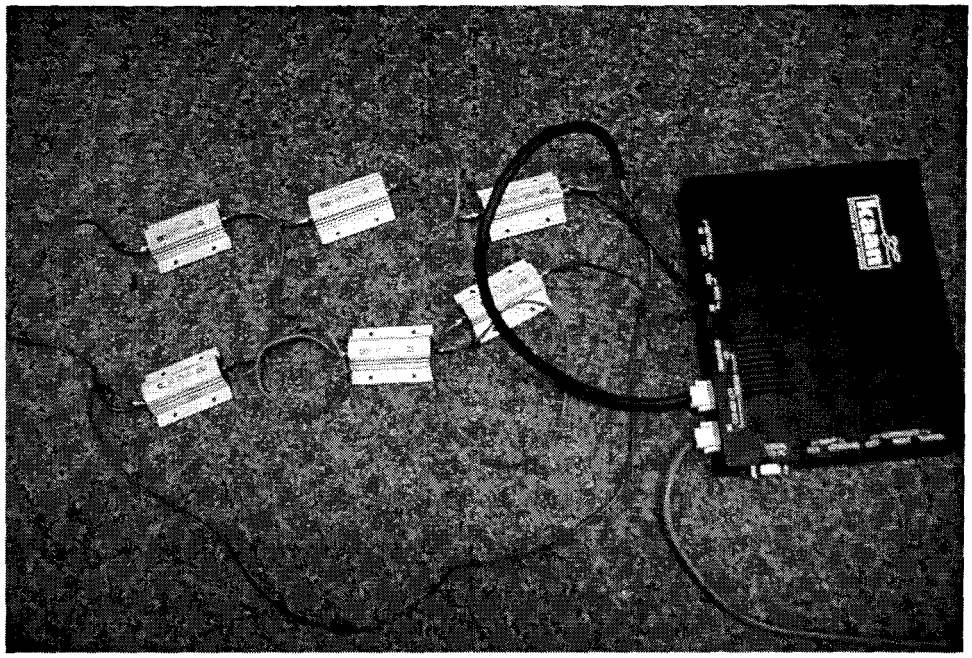


Figure 3-10: Power Testing (2000)

3.3.3. Audio

The audio block development needed to consider the:

- Type and number of inputs and outputs.
- Audio management:
  - Mixing and dispatching.
  - Volume levels and muting.
  - Network driving and receiving.
  - Grouping.

As the music technology system was to be used stand-alone as well as in a network, it was decided to have at least two stereo inputs per station for a keyboard or other instruments **(Requirement R6).** If this number was two, for a greater period one of these inputs would be connected to a portable keyboard and the other input could be connected to more than one instrument at the same time. An external mixer would then be required and therefore both stereo inputs

were set as line-in level. Networked audio could come from up to four buses (if the control bus is unused). A total of eight mono signals (Keyboard L and R, Auxiliary L and R, Bus 1 to 4) were available per EMI.

For the outputs, the signals could be sent to any of 4 network buses. Headphones outputs were also required locally. Three stereo outputs were required (for up to 2 students per keyboard and the teacher) connected to 2 mono channels called Left and Right. Each output would need to be controlled so that each could be set in different modes:

- **Left Student Output:** Left + Right (stereo, in Solo mode) or,
  Left + Left (can only hear himself in Split mode).

- **Right Student Output:** Left + Right (stereo, in Solo mode) or,
  Right + Right (can only hear himself in Split mode).

- **Teacher Output:** Left + Right (stereo, in Solo mode) or,
  Left + Left (Left student only) or,
  Right + Right (Right student only) or,
  Left and Right + Left and Right (Left and Right students).

This gave six mono signals to output (Left, Right and Bus 1 to 4).

The eight mono input signals had to be mixed together in a set pattern and then dispatched to any of the six mono outputs. The headphones outputs would eventually be set to match any of the above configurations. These headphones outputs would also require standard audio management such as volume setting and muting. Networked signals were eventually used to group EMIs together on any of the four buses. Four buses limited the music technology system to four groups at most. A method to handle more groups had to be investigated.

Constraint 2: Only four buses, a method to use them the most efficiently was required.

Figure 3-11 shows the final audio block function diagram.

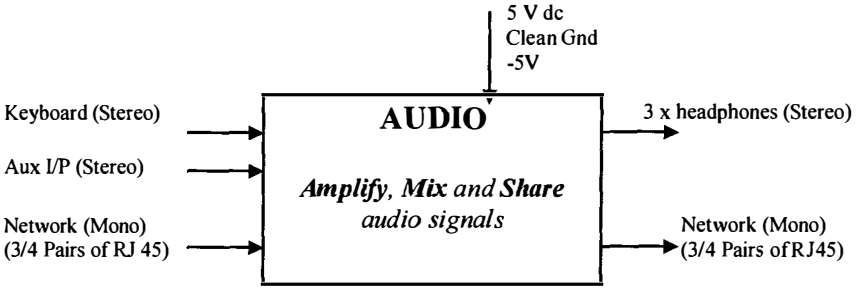


Figure 3-11: Audio Functional Diagram (1999)

Audio management was achieved using standard components (see Appendix 8.2.2 page 238, for audio designs):

- Headphones Op Amp for the clean ground referencing, mixing and output driving stages.
- Eight way switch ICs (Max395) for the mixing stage. One switch was used per output/channel and one for the headphones output setting. Each switch could be controlled separately from the microcontroller using the Serial Peripheral Interface (SPI) standard.

However, three problems occurred in the design stage:

PROBLEM 1: Audio signals follow a logarithmic scale. However, only linear digital potentiometers were found to produce controllable amplification (volume control). A conversion table had to be calculated to emulate a logarithmic law using a linear potentiometer (see Appendix 8.2.2, page 238). A test consisting of hearing the increase in volume of a given audio signal was carried out to check that the volume control was performed smoothly using that assumed law. The test was successful, which validated the law, and 32 volume levels were achieved.

PROBLEM 2: Audio network drivers and receivers working in this application could not be found. It required the design of switchable audio line drivers and receivers. The driver had to be switchable in order to drive audio to a

bus only when required. Appropriate components were found, however received signals decreased with the number of EMIs on the bus. It was found that the selected chip could be considered in operation as a resistor. Therefore, Ohm's law had to be applied for each EMI added on the bus and the corrected 'loading' of a bus had to be calculated to compensate the resistance effect of the chip. It was decided to control the volume of the received signal coming from a bus as well as the loading. Therefore, a corrected table needed to be calculated from Problem 1 law to account for the number of EMIs on the bus (see Appendix 8.2.2, page 238).

PROBLEM 3: Grouping was limited using only four buses. A method to get more buses or sub-buses had to be considered. This was achieved by adding a relay controlled by the microcontroller on one of the buses (bus 2). This relay would divide a bus into sub-buses where only adjacent EMIs could talk to each other. This meant that only contiguous EMIs could be grouped together, which was satisfactory as teachers could group stations close together so that students can see and interact with each others.

The addition of this relay required that the net sockets had to be labelled 'IN' and 'OUT', (as the network was daisy chained) and in order to have the relay always on the same side for better control. For grouping, an EMI was represented as Figure 3-12:

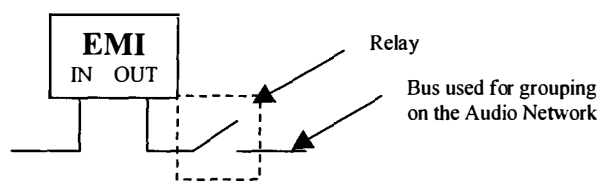


Figure 3-12: EMI representation for grouping (1999)

Constraint 2 (changed): Only contiguous stations could be grouped. This required the labelling of network connectors.

3.3.4. Microcontroller

As described in 2.2.2 (page 22), the microcontroller chosen was a highly integrated chip that typically contained a Central Processing Unit (CPU), memory as RAM and some form of ROM (PROM, EPROM, EEPROM), I/O ports, and timers. Unlike a general-purpose computer, which also included all of these components, the microcontroller was designed for a specific task - control. As a result, the parts could be simplified and reduced.

The requirements for the EMI microcontroller were (as shown in Figure 3-13):

- To control the internal circuitry (mixer switches, digital potentiometers, etc.) using a SPI bus.
- To communicate with the external world:
  - MIDI with a MIDI device.
  - RS232 with a computer.
  - Another protocol to be defined between EMIs.

  And eventually process MIDI signals in real time.
- To be easy to program and reprogram for upgrades.

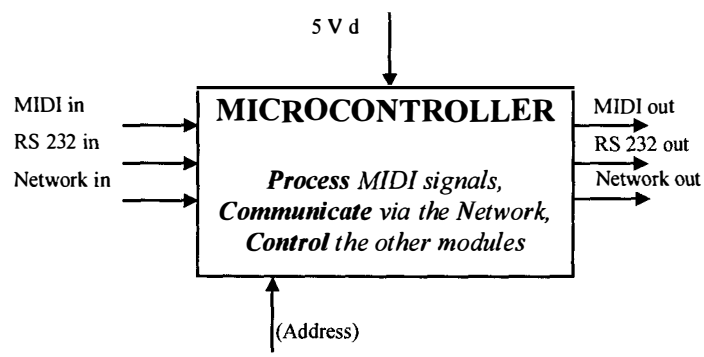


Figure 3-13: Microcontroller Functional Diagram (1999)

INTERNAL CIRCUITRY CONTROL: An SPI bus was used to communicate with digital components (compatible with SPI). To communicate with an SPI compatible chip, the microcontroller needed to make the selected chip listen to the SPI bus before talking to it.

Chip Select pins to allow Other 'On/Off' function devices (such as the relay for bus splitting) were controlled using mosfets (a type of transistor) as controllable switches.

COMMUNICATION WITH MIDI KEYBOARDS (MIDI PROTOCOL): An EMI had to communicate with any connected MIDI portable keyboard (**Specification S1**) to control it. The MIDI hardware design was available along the MIDI specification and was copied.

COMMUNICATION WITH PC (RS232 PROTOCOL): If an EMI or the Kaan system were to be controlled by a computer, at least one EMI would be connected on one of the available I/O ports of that computer, as other peripherals could be. Available I/O ports included (see 2.2.3, pages 29 or 32):

- Serial port.
- Parallel port.
- Joystick/MIDI port (from available sound card).
- USB port.
- Firewire port.
- Wireless (Wifi or Bluetooth).

The communication standard requirements were:

- Baud rate slightly higher than MIDI baud rate (31250 bauds) to allow the propagation in both directions of MIDI messages in real time for MIDI recording.
- Full duplex (Transmission and reception could be processed in parallel).

This standard did not need to be fast or complicated. The RS232 communication standard using the computer's serial port was chosen as the easiest option. The USB or Firewire standards could be used for future work if serial ports became unavailable. Only Tx and Rx of the RS232 protocol were needed for communication with a PC.

The RS232 communication was specified with:

- 38400 baud (bits/sec) to be slightly higher than MIDI baud rate (31250 Bauds). A value of 38400 baud was the next one available in the RS232 specification.
- Tx-Rx full duplex.
- 10 bit words (start bit = 0, Least Significant Bit sent first).

The RS232 circuitry was tested using the test rig shown in Figure 3-14.

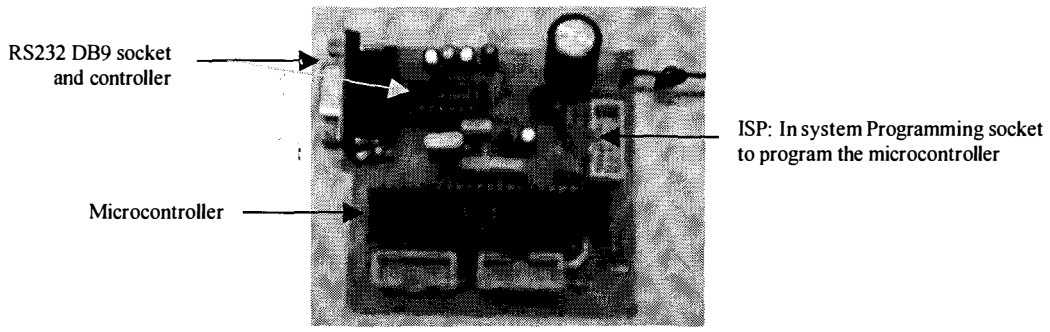


Figure 3-14: RS232 test rig (Tewkesbury, 2000)

As the EMI used a virtual ground of +5V internally, 'shift down' and 'shift up' facilities were required to transmit and receive from a PC, as a PC uses standard voltage levels (ground of 0V).

COMMUNICATION BETWEEN EMIS (NETWORK PROTOCOL): If a control device fired a command, the command had to be sent to the KAAN network to reach the EMIs to set. A network communication standard for remote control had to be investigated and the choice depended upon:

- Number of EMIs able to communicate.
- Maximum Baud rate.
- Maximum distance between two EMIs.

Among the available standards (some are shown in Table 3-8), the RS485 standard was found to be suitable as it allowed up to 32 EMIs to be used. However, the RS485 standard was half duplex, meaning that an EMI could not 'speak' and 'listen' on the control bus at the same time.

	RS232	RS422	RS485
Cabling	single ended	single ended / multi-drop	multi-drop
Number of Devices	1 transmit 1 receive	5 transmitters 10 receivers	32 transmitters 32 receivers
Communication Mode	full duplex	full duplex half duplex	half duplex
Max. Distance	50 feet at 19.2 Kbps	4000 feet at 100 Kbps	4000 feet at 100 Kbps
Max. Data Rate	19.2 Kbps for 50 feet	10 Mpbs for 50 feet	10 Mpbs for 50 feet
Signaling	unbalanced	balanced	balanced
Input Level Min.	+/- 3 V	0.2 V difference	0.2 V difference
Output Current	500 mA (Note that the driver ICs normally used in PCs are limited to 10 mA)	150 mA	250 mA

Table 3-8: Communication Standards Specifications [from RS485 (n.d.)]

The RS485 communication was specified as:

- Around 38400 baud (bits/sec) to be slightly higher than MIDI baud rate (31250 Bauds) to allow the propagation of MIDI messages in real time, like the RS232 standard. The baud rate for MIDI and RS232 needed to be within a certain frequency tolerance as they were interfacing with standard devices. However, the baud rate for RS485 did not need to be standard as it was used only between EMIs.
- Tx-Rx .Half duplex.
- 10 bit words (start bit = 0, Least Significant Bit sent first).

The RS485 circuitry was tested using the test rig shown in Figure 3-15. A computer was able to send RS232 characters and messages to an RS232 console through two of these test rigs.

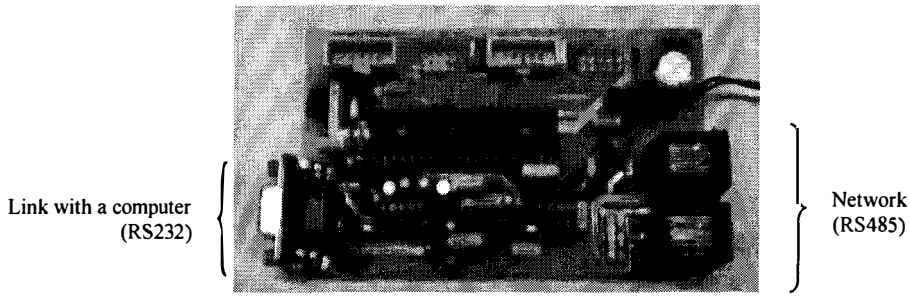


Figure 3-15: RS232 to Net test rig (Tewkesbury, 2000)

COMMUNICATION (Final Design): The microcontroller needed to handle at least two communications at the same time. A control device needed to be connected to an EMI MIDI port (if the control device was a keyboard) or an EMI Serial port (if the control device was a computer). This EMI then needed to pass the commands through the network. The other EMIs needed to 'listen' to the network for commands and to eventually process MIDI messages in real time if a keyboard was connected to them.

Communication handling was performed via Universal Asynchronous Receiver Transmitters (UART). The new Atmel Mega161 featured two built-in UARTs and enough memory (16Kb) and clock rate (up to 8MHz). It could also be self re-programmed and therefore could be programmed via the network through one of its UARTs **(Requirement R8).**

Calculations (see Appendix 8.2.2, page 242) were made to determine the best crystal to use to clock the microcontroller as the microcontroller had to handle three different protocols running at different baud rates:

- MIDI at 31250 bauds ±5%.
- RS232 at 38400 bauds ±5%.
- RS485 at around 38400 bauds

The crystal frequency was selected as 8MHz with minimum error for both RS232 and MIDI AND best processing speed (for real time MIDI processing). At that stage, the microcontroller was still being developed by the manufacturer. The main PCB was designed for this chip using RangerXL CAE software [from Seetrax (n.d.)]. A daughter board, using another microcontroller and an external SPI UART, was designed to emulate the final microcontroller. This was inserted on the main PCB in place of the final microcontroller. In System Programming (ISP) circuitry allowed programming of the microcontroller, as it did not feature programming through its UART. The daughter board can be seen in Figure 3-16.

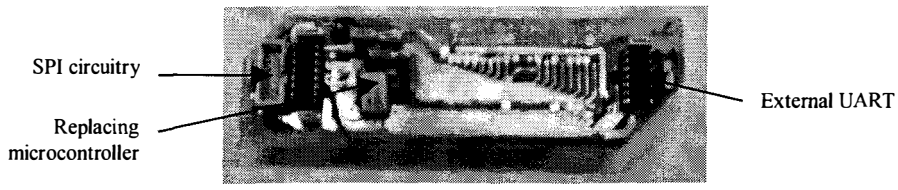


Figure 3-16: ATMega161 emulation daughter board (Tewkesbury, 2000)

3.3.5. EMI Production

PCB design went through 8 iterations before production. Improvements were:

- *Digital noise*: ground planes were added and digital tracks routed further away from sensitive audio tracks to avoid digital noise.
- *Input levels*: it was found that it was difficult to balance the levels of the sound from EMIs connected either to a keyboard, a mixer or PC sound cards. The pre-amplification conducted in the input stage had to be redefined and customised against the use of the connection.
- *Communication with PC (RS232)*: communication problems occurred with the PC due to digital level latency.

Figure 3-17: RS232 Signals causing communication problems (2000)

The network state had to be defined for the instance when nobody 'talked' to prevent level stabilisation latency and therefore communication problems. This needed to be set only in one box, with 'pull up' and 'pull down' resistors. The value of the pull up and down resistors of the network terminator had to be calculated carefully as some EMIs would not communicate properly. The calculations can be found in Appendix 8.2.2 on page 243.

The two following figures show the EMI electronic circuitry:

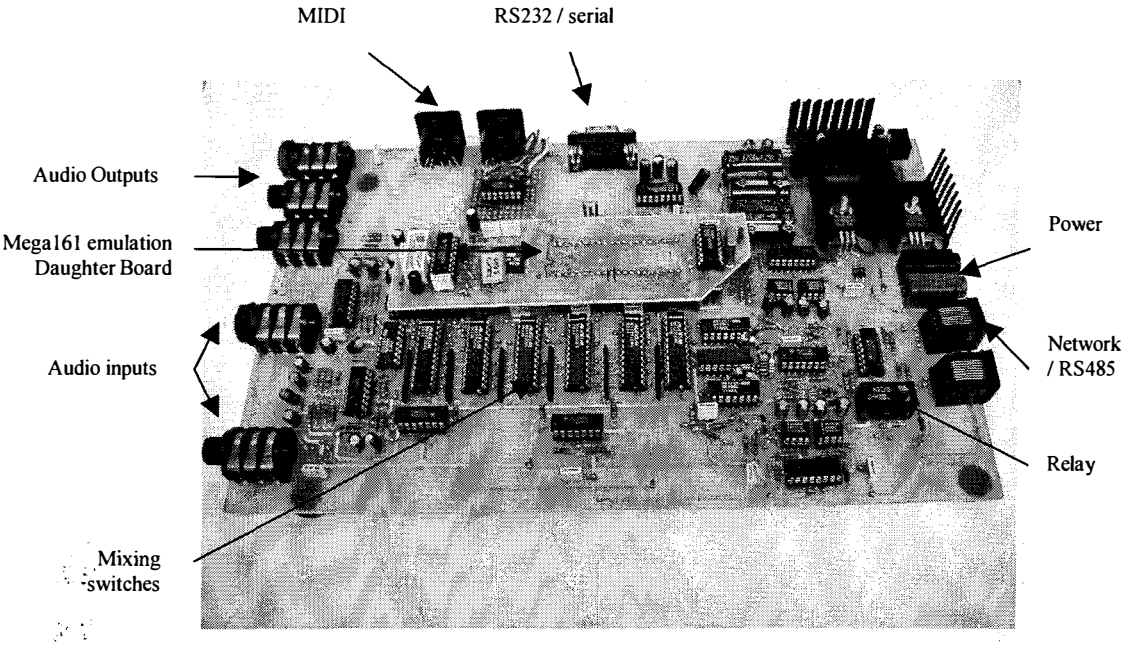


Figure 3-18: First Prototype (2000)

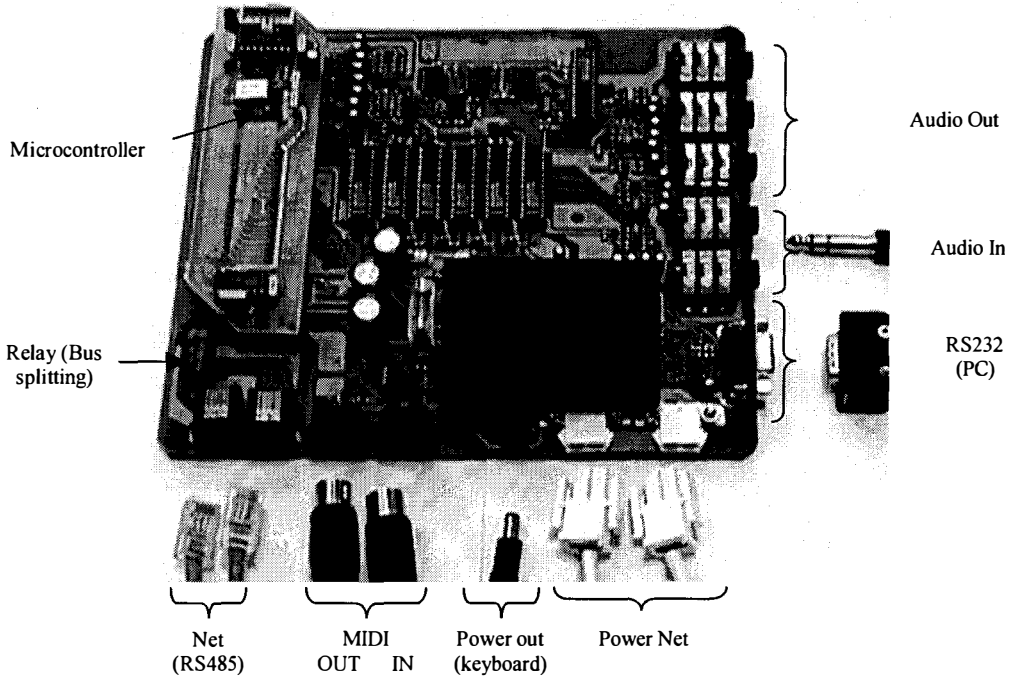
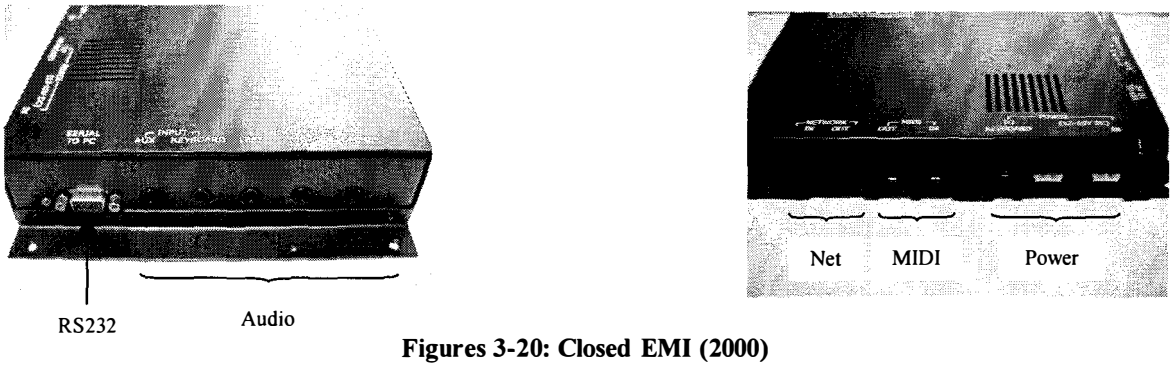


Figure 3-19: EMI Main PCB with daughter board (2000)

A metal enclosure for the EMI was designed to enclose the EMI PCB. The requirements for the metal enclosure were to enclose the PCB and allow connectors to protrude, be secure (so that students could not disconnect cables and change the setup) and safe [(so that students could not reach dangerous cables, grounded, Electro-Magnetically Compliant and prevented Short circuits (i.e.

from things poked into the box by students)]. The metal enclosure could be installed under a desk, on a wall or mobile, installed on pods, non-attached, etc.

Designs steps can be found in Appendix 8.2.3 (page 244). The metal enclosure was designed by hand, as the design was simple. The plans are the propriety of the collaborative company and are not enclosed in the appendices. The following figures show the final EMIs alone and installed in diverse configurations:



Figures 3-20: Closed EMI (2000)

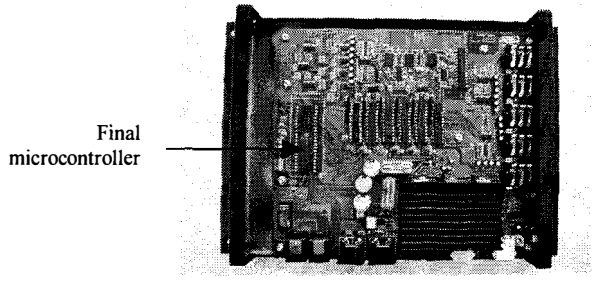
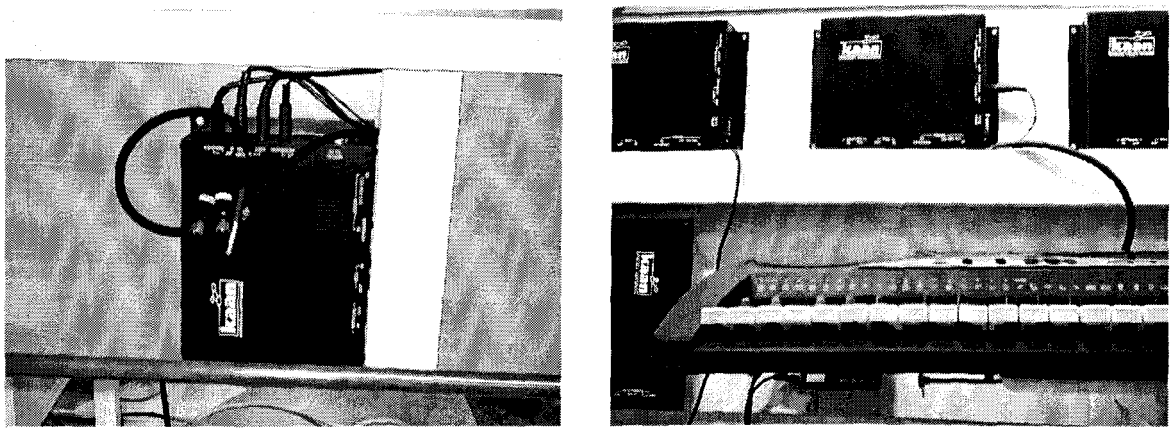


Figure 3-21: Open EMI (2000)



Figures 3-22: EMI under a desk or on a wall with trunking to hide / protect cables (2001)

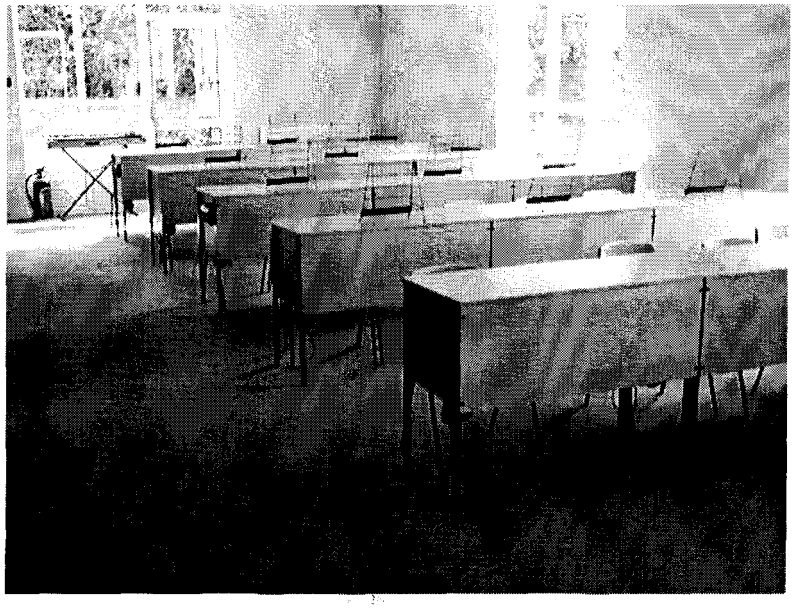


Figure 3-23: Kaan System (Perins School, Hampshire, UK, 2002)

3.3.6. Discussion on the Hardware section

The new music technology system called KAAN consisted of networked EMIs. The music technology system could be controlled by any control device either using MIDI protocol or protocols using the RS232 communication standard. EMIs could be connected to any instrument and could process MIDI messages from a portable keyboard in real time.

An EMI consisted of 4 blocks designed using defined specifications.

Supply Power: A single Power Supply Unit (PSU) was needed and a networked DC power line was connected to every keyboard instead of multiple power supplies and power sockets.

Process Audio: The EMIs had to provide on board controllable audio mixing and amplification for 2 direct stereo inputs (keyboard and auxiliary), and up to four mono networked inputs. The resulting signals were sent either to three controllable stereo outputs or back to any of the network buses. The distribution of the audio processing down to each EMI allowed better audio quality and less cabling.

Process MIDI: A 5 octave portable keyboard could be split into two separate two octave keyboards, with the middle octave being used to control transpose, pan, voice and volume, by processing MIDI data in real time. Three modes could be set: Solo, Split or Duet.

Communication via Networks: Three configurations were defined. The first two used a daisy-chained network carrying audio and data signals. The third configuration used an additional network for data communications to the daisy-chained network, which was then used purely for audio. Each station could be remotely configured and controlled from a central controlling device.

A *first configuration* consisted of a network of daisy chained EMIs. The teacher's EMI was connected via MIDI to a portable keyboard (see Figure 3-24). Keys on the keyboard allowed the teacher to listen to an individual player or a group of students. The teacher was able to remotely control and set up every keyboard on the network through the "master" keyboard.

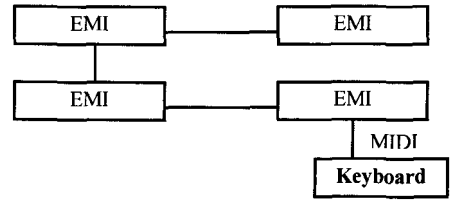


Figure 3-24: First Network Configuration (with a control keyboard, 1999)

The teacher could speak or play, to any individual student or a group of students from the front of the class. Furthermore, individual students or a group of students could perform to the class.

A *second configuration* allowed every function available using configuration 1, but a computer was used to control the music technology system instead of a keyboard (see Figure 3-25). This computer also enabled the teacher to record student work in MIDI or Wave files. Music sequencing and scoring software could be installed in order to use students' work. The PC was linked to the network via its serial port using the RS232 standard.

A user interface allowed the teacher to control the music system nodes and the keyboards, to store students' marks and details, and to perform other administration functions.

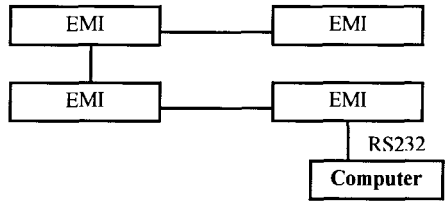


Figure 3-25: Second Network Configuration (with a control computer, 1999)

The Dual Network System was an extension of the two systems described above. It used the daisy-chained network with computers (PCs) linked together in a separate data network and plugged into the EMIs' RS232 (serial) port. The daisy-chained network was then used purely for audio. The firmware had to be modified to handle communication from MIDI and RS232 instead of MIDI and RS485. The PC included music software for the students. This music technology system configuration is shown in Figure 3-26.

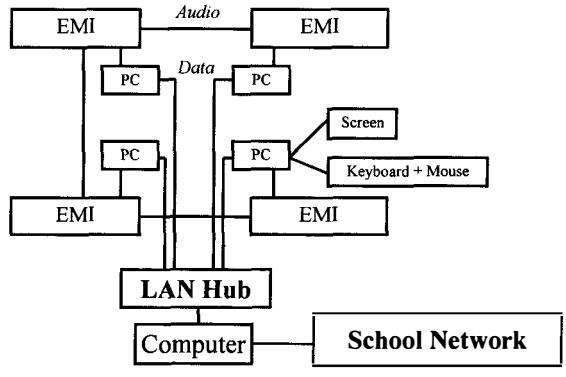


Figure 3-26: Third Network Configuration (Dual Network System, 1999)

This configuration was never installed as a simpler alternative was selected by connecting computers on students' desk to the EMI differently. EMIs were linked to their computer via the computers' sound card only. Control was still performed via one of the four network buses. MIDI messages from the keyboard were sent to the EMI for local processing as well as the computer sound card for sequencing with a MIDI splitter. The sound card audio (stereo) output was linked to the EMI's auxiliary input. This alternative configuration was installed at Davisons High School in Worthing, West Sussex, UK.

The final EMI block diagram could be fully described in Figure 3-27.

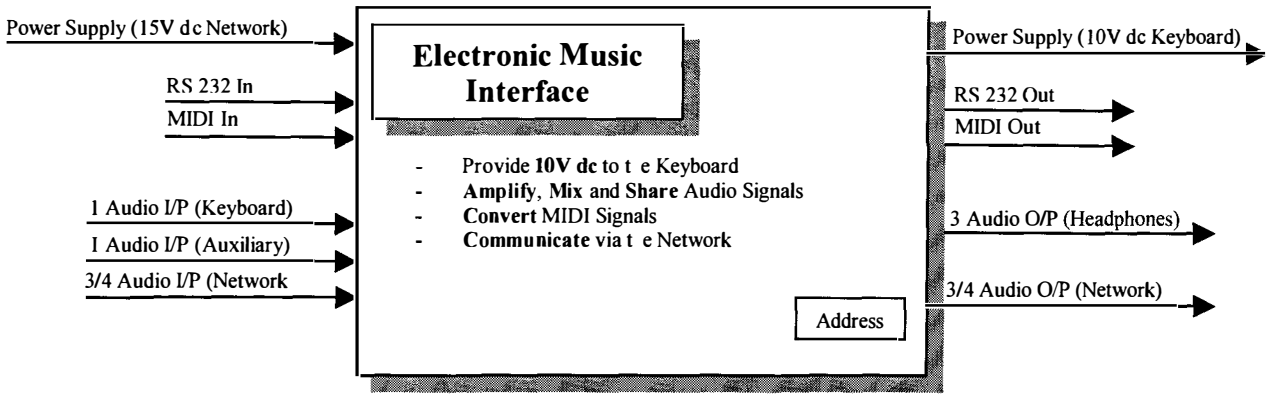


Figure 3-27: Electronic Music Interface Functional Diagram (1999)

EMI hardware comprised a PCB and a metal enclosure (design considerations are described in Appendix 8.2.3, page 244) and was designed by the author with assistance from the University of Portsmouth and manufactured by a local company. The designs are the property of a collaborating company and are not included in the appendices.

A microcontroller (or an emulation daughter board) was among the electronic parts of an EMI to control the circuitry and to communicate with the external world.

3.4. Firmware

The firmware consisted of a new protocol based on the RS385 standard and of microcontroller code implementing functions complying with the new protocol. The protocol was proposed by the author and accepted for implementation by Dr Giles Tewkesbury. Source code was fully written by Dr Giles Tewkesbury at the University of Portsmouth in pseudo C++ (from ATMEL). The source code was tested by Dr Giles Tewkesbury with the author using smaller VB6 application written by the author. The work of Dr Giles Tewkesbury encompassed the source code and the commands shown in appendix 8.2.4 from page 247 to 253.

The microcontroller controlled the internal circuitry of the EMI. Control functions required to control the internal circuitry were:

AUDIO FUNCTIONS:

- Switching/mixing of up to eight mono inputs (keyboard left and right, auxiliary left and right, bus 1 to 4) and dispatching into six output channels (left, right, bus 1 to 4).
- Control of left and right outputs volume, and input level of the buses, as well as muting.
- Output switching (for the students and teacher left and right (pseudo stereo) or all left/all right).
- Attachment to talk to a bus or split the splitable bus.

MIDI FUNCTIONS (WITH A MIDI PORTABLE KEYBOARD):

- Send / Receive MIDI data from the MIDI port or the network (record into a MIDI file, or down / upload keyboard's memory).
- Filter / process MIDI messages to change mode, transpose, etc. Any model of keyboard had to be controlled by the EMI. To change the 'split mode' of a station, MIDI messages had to be sent to the keyboard to set it in an appropriate state. However, all keyboard models implemented the MIDI specification in different ways. Therefore, EMIs had to be 'customised' to send the correct MIDI messages to its attached keyboard. This was achieved using a chunk of the microcontroller EEPROM. In order to control the keyboard panel and set it as required, this part of memory of the microcontroller was programmed and used to keep customary information on the portable keyboard linked to the EMI.

OTHER FUNCTIONS:

- Send / receive data from the RS232 (serial) port.
- Send / receive networked data.
- Receive debug information (displayed with a tricolour LED), reset EMIs, etc.
- Write / read EEPROM (see MIDI Functions above).
- Reprogram the microcontroller.

The new communication protocol to remotely control EMIs was specified against the RS485 communication standard and the above functions.

3.4.1. Choice of communication standard

The RS485 communication standard allowed up to thirty-two EMIs to communicate on the network (see Table 3-8, page 88). However, as this standard was only half-duplex, only one EMI could 'speak' on the network at a time. It was decided that only one EMI would initiate the communication in order to simplify the protocol. This EMI was the one linked to the control device (for instance a PC) and was called the **Master EMI**. The other thirty-one potential students' EMIs were then called **Slave EMIs**.

Slave EMIs could only answer a Master's request for data or status. Each EMI needed to be given a unique address (0 for Master, 1 → 31 for slaves) so that each EMI could respond to commands. These commands could be either addressed to 1 EMI (**local command**) or to all of them (**global command**). The commands could require a number of parameters.

3.4.2. Definition of custom protocol

Having defined the way the EMIs would respond to commands from the PC, the type of commands needed to be investigated:

- For local (addressed) commands, the address of the addressed EMI needed to be included.
- Local and global commands needed to be considered.
- An understandable command number had to be sent.
- A way to check if the message was correctly received had to be added.
- The length of the commands needed to be considered:
    - Either of fixed length = maximum length (according to the maximum number of parameters to be sent with a command).
    - Or of variable length:
        - Like MIDI, the first byte tells the number of bytes to follow (if 9n hex then 2 bytes to follow, if Bn hex then 1 byte to follow).
        - Indicate the number of bytes to follow.

To check if a command was received correctly, an extra byte, called a Checksum byte, was sent at the end of each command. This checksum byte was calculated as the Xor-ed value (Exclusive Or logical operator) of all the bytes of the command and a known value. This method was usually used for a small number of bytes.

It was decided to have a first byte indicating the nature of the message, like MIDI status byte (See 2.2.3, page 33), because this prepared the communication controller with what to expect from the following bytes. The nature of the message was of the sort:

- Local / Global: 1 bit required (say 1 = to 1 EMI = local, 0 = to All = Global).
- Address: 32 EMIs = 2^5 ➔ 5 bits.
- Bytes to follow (databyte): Unknown at the moment, but 2 bits left = up to 3 bytes to follow.

After considering the different kinds of messages that would be sent, three parameters were considered sufficient for any command:

- 0 databyte commands: Ping an EMI to check it responded.
- 1 databyte commands: Mute both sides, Read mute status.
- 2 databyte commands: Volume V for side (left or right).
- 3 databyte commands: Other special commands?

Local commands could include a first byte with an EMI address and number of bytes to follow, a command byte and, up to two parameter bytes. For a global command, no address was needed and therefore a command number could be set instead of the address. Global messages could then include a first byte with a command number and number of bytes to follow and, up to three parameter bytes. As no explicit commands were found to require more than three bytes, it was decided to proceed with this solution.

In summary, only one Master EMI (address 0) could be present on the network. The Master EMI was the only device in the music technology system to be allowed to initialise communication. The Slave EMIs could answer a Master request for

data or status. A message comprised of one to four bytes (of 8 bits), followed by a checksum byte, which validated the message:

- 1 first byte which defined:
  - The type of command (local / global).
  - Who the message was addressed to (if local).
  - The number of following bytes after this first byte.
- Up to three following bytes (the number was defined by the first byte).
- A Checksum Byte (every byte was Xor-ed with each other and a determined initial value).

From the protocol definition stated above, a message was composed of a First Byte and of following bytes. The 8 bits of a first byte were defined as follows:

G	x	x	x	x	x	F	F

With:

FF: Bytes to follow.

xxxxx: Address or Command.

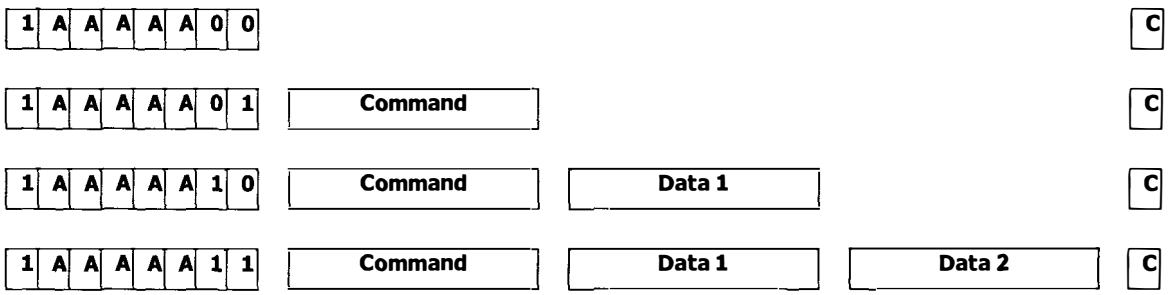
G: Addressable Mode.

FF	Description	Notes
00	0 Byte to follow	
01	1 byte to follow	
10	2 bytes to follow	
11	3 bytes to follow	

G	Description	Notes
0	Message addressed to every EMI	First Byte called First Command Byte
1	Message addressed to only 1 EMI	First Byte called Address Byte

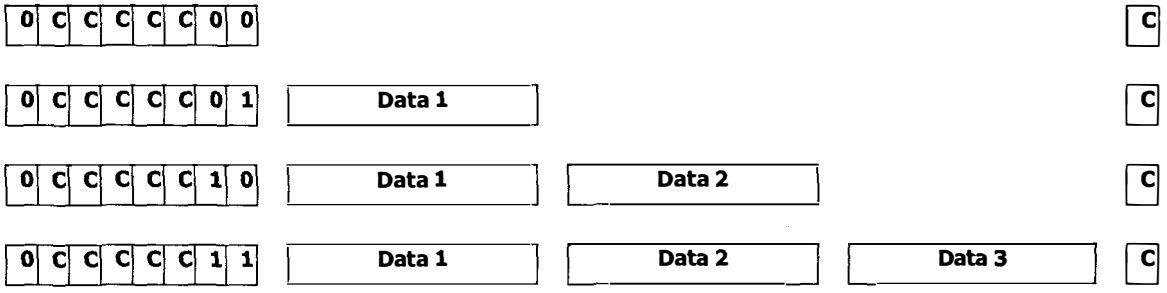
Tables 3-9: First Byte description (2000)

From the first byte and especially G and FF bits, messages were constructed as follows for all the alternatives for a first byte. C represents the checksum byte, at the end of every message to validate it.



With:

AAAAA: Address of the EMI.



With:

CCCCC: Command.

C: Checksum Byte.

There were 2^5 = 32 different addresses AAAAA. Master EMI's address = 00000. There were also 4 sets of 2^5 = 32 instructions available for a First Command Byte, each set having a different number of following bytes. There were 3 sets of 2^8 = 256 instructions available for a following command byte, each set having a different number of following bytes. Finally, there was a range of 2^8 = 256 [0 - 255] for data. For G=0 and FF=00, the byte was called Stand Alone Command; for G=1 and FF=00, the byte was called Enquire Byte.

3.4.3. Coding and testing

The source code of the microcontroller had to be changed after the introduction of the new ATMEGA161 and the use of its internal second UART.

The full protocol is documented in Appendix 8.2.4 (page 246). The documentation shows the available commands and methods to use them.

The protocol was tested using small utilities written in Visual Basic. Visual Basic was selected as it allowed:

- The user-friendly design of effective Graphical User Interfaces (GUI).
- Object Oriented Programming (OOP) (see 2.2.4, page 38), which in this case allowed the creation of a single portable object called Transceiver to handle communication between the PC and the Master EMI through a serial port. This Transceiver was then used in the different small utilities.

Additionally, the development software was already used within the research laboratory with available expertise.

The following shows how each byte coming from the communication port was handled to get messages. As the communication was asynchronous (no timing involved), to read the status of a Slave, a read command was sent to the Slave and then a message was awaited within a determined time out. The 'Got Message' routine confirmed if the bytes received constituted a valid message.

COMMUNICATION HANDLING ALGORITHM: An OnComm event was raised from Mscomm object (interacts with the communication Port) every time a defined number of bytes were stored in the serial port input buffer. This was set to 1 to raise the event at every received byte. An array was used as a rolling buffer to store the received bytes. Two modes were set to handle these incoming bytes differently:

- *In Synch* (the first byte to come was expected to be a first protocol byte, which will define the number of bytes to follow and the checksum byte). Every time a message was validated the 'Got Message' routine was launched.

- *Not In Synch* (each incoming byte was expected to be the checksum byte that was to validate a message composed of previously received bytes). Set at the beginning and if an expected checksum byte in 'In Synch' mode failed to validate a message.

- GotMessage: handled incoming messages and stored messages for use with 'Chained' messages (messages sent to every EMI in turn to respond to get an overall state of the music technology system). Raised a 'RecMsg' event on reception.

- ReceiveMessage: handled incoming messages that were waited for, such as requests for slave's status. A timer was set to handle time outs in case of no response. This time out was set to 200 ms.

The Transceiver main core routines were to:

- Send local addressed commands to the network.
- Send global commands to the network.
- Receive (in an asynchronous way) answers back from Slaves EMI.
- Handle communication errors and re-synchronising.
- Maintain a collection of the entire catalogue of commands set in the protocol.
- Handle raw MIDI data.

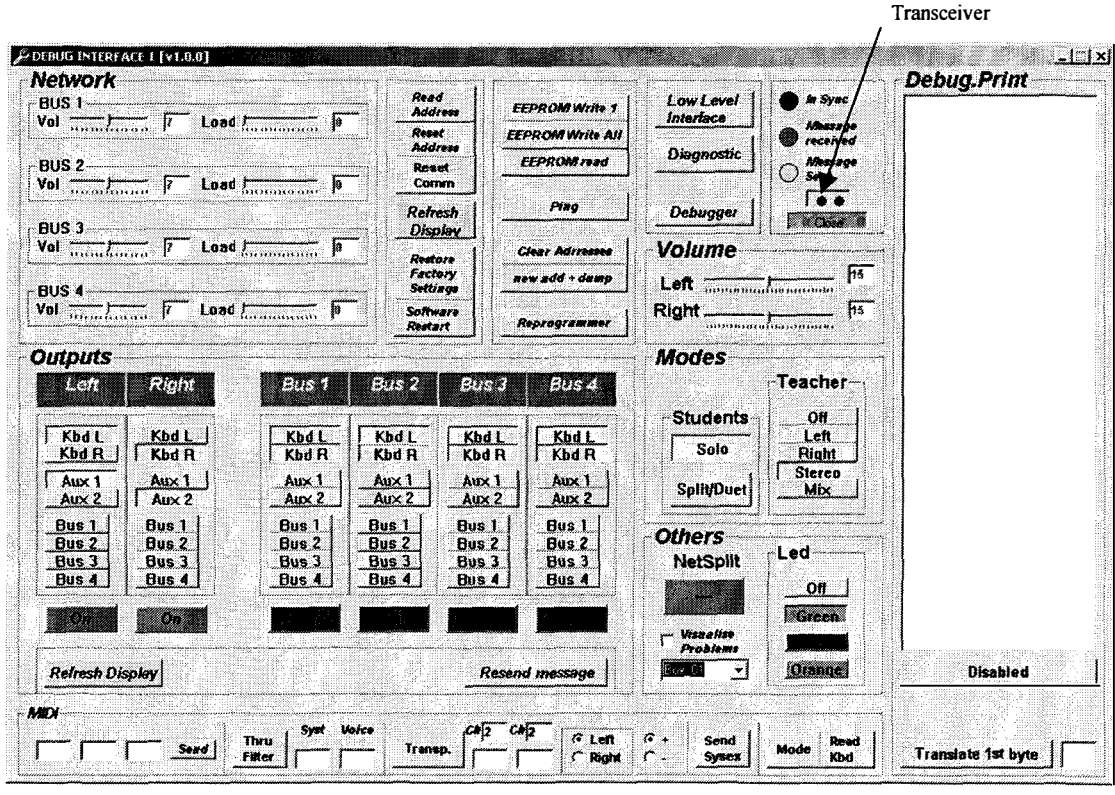


Figure 3-28: Debug Interface to read and set every low level parameter of a determined EMI (2000)

The main utility designed for the purpose of debugging the protocol was a debug interface (as shown in Figure 3-28). All the low level functions were interfaced or accessible through this program. Any EMI could be called to give the interface the information on its status to refresh the display. Any action on the interface would set the addressed EMI with the new required status. This utility was also used on site after an installation or a maintenance action to test the music technology system.

The music technology system debugger shown in Figure 3-29 tested communication between the PC and any EMI by trying to 'refresh' 100 times the status of each of them.

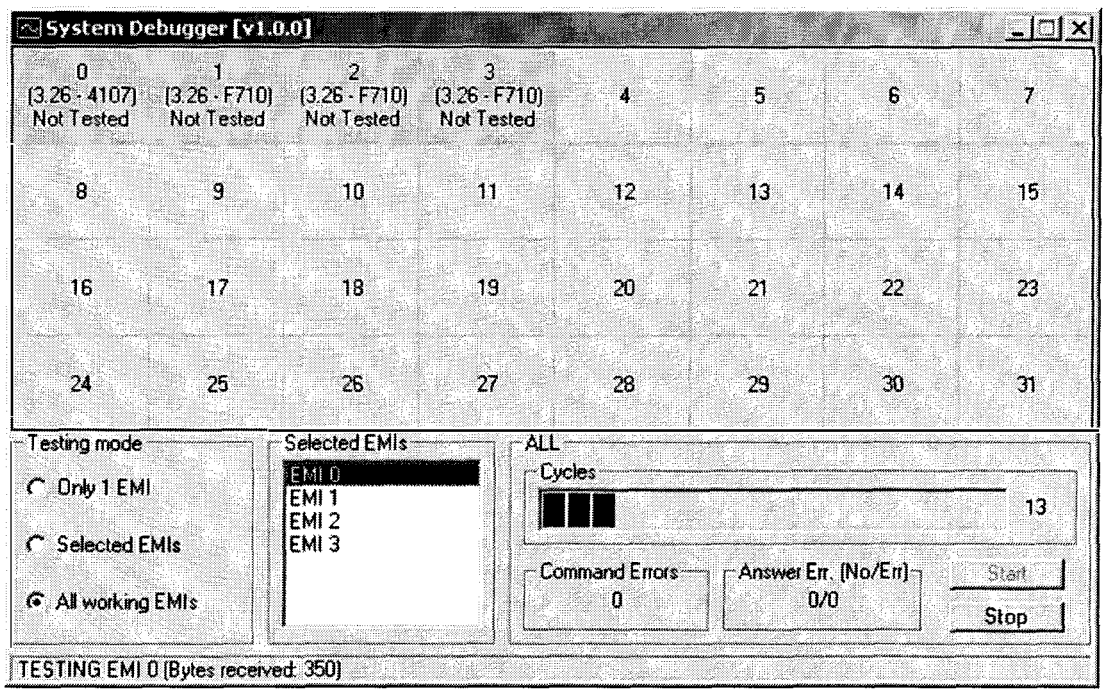


Figure 3-29: System Debugger to assess communication problems (Quality Control tool, 2001)

Other utilities were created to test protocol commands and test system features. These can be found in Appendix 8.2.4 (page 246) with an explanation on the use of the commands involved. However, these utilities are presented below as they were mainly used for testing, installation and maintenance.

EMI ADDRESSING: The utility shown in Figure 3-30 allowed an address to be set to any unallocated EMI in the network. It first pinged all EMIs from 1 to 31 and then allocated the last address where no EMI pinged back. It then pinged

back this address to check if the new address was set correctly. If not, the background turned red.

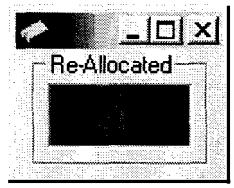


Figure 3-30: New unallocated address (2000)

MIDI: The utility shown in Figure 3-31 was used to interface with a MIDI keyboard to capture key presses and to send MIDI messages.

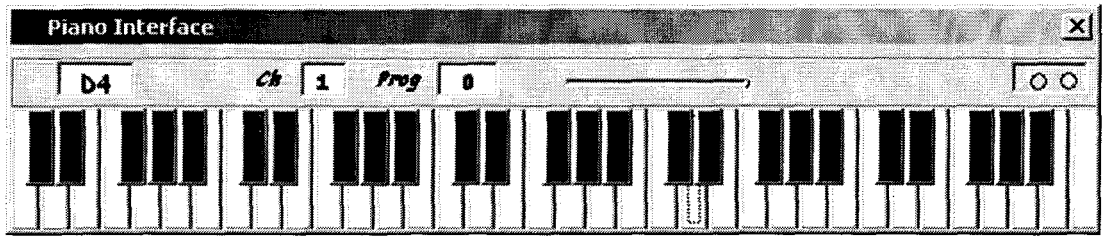


Figure 3-31: Piano interface (2000)

The utility shown in Figure 3-32 tested the MIDI Sysex Dump handling (import / export) from different models of keyboards.

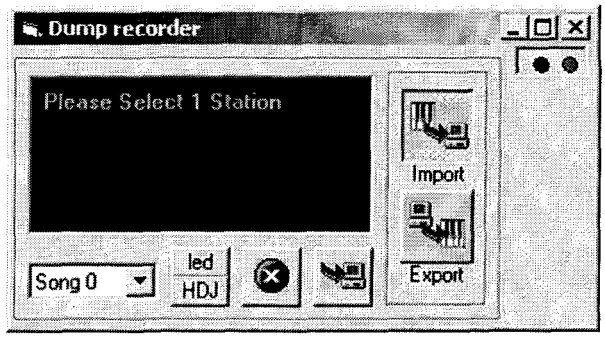


Figure 3-32: MIDI Dump handler (2001)

The utility shown in Figure 3-33 was a MIDI recorder used to test first the MIDI transfer between one EMI and the PC and second to test the creation of understandable MIDI files.

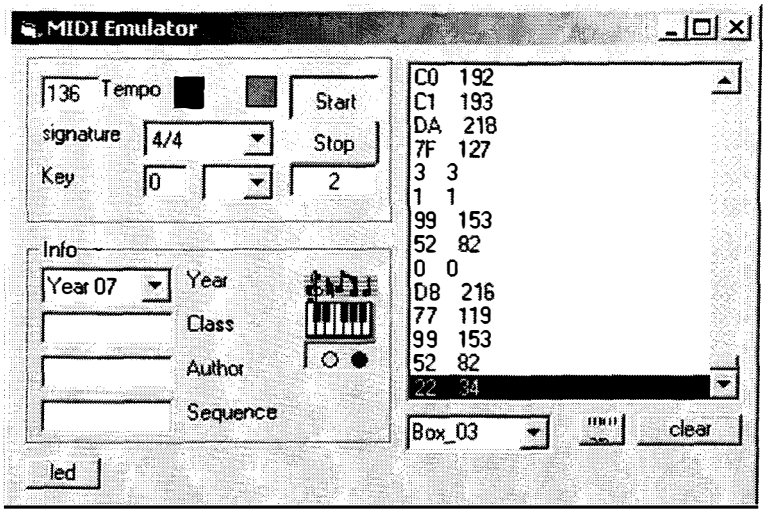


Figure 3-33: MIDI Recorder (2001)

MIDI files were created from the received raw MIDI bytes stored in a temporary file and then filtered. The receipt of notes timestamps was compulsory to determine the length and position of each played note. The receipt of F8 timestamps allowed determination of the tempo of the song played.

EEPROM PROGRAMMING TO CUSTOMISE KEYBOARD CONTROL: EMIs could process MIDI messages from an attached keyboard in real time. This allowed them to record MIDI and to control the keyboard for the split and duet modes. The KAAN specification allowed the use of any keyboard model with the music technology system. As all keyboard models operated differently, part of the EEPROM memory was used to store information relating to the attached keyboard to customise its control. The EEPROM could be programmed from a simple file called EEPROM.txt. EMIs controlled their attached MIDI keyboard in Split and Duet modes by capturing MIDI events sent after a panel button was pressed. Reset patterns were stored in the EEPROM and compared with the incoming MIDI messages. If a reset pattern was recognised within the message, the EMI status was reset to its stored status. Some of the keyboard panel buttons (such as the intrusive Demo button) did not send any message and had to be disabled definitively within the hardware. The utility shown in Figure 3-34 was used to control the custom area of the EEPROM. The EEPROM was written from an EEPROM file (see Appendix 8.2.5, page 262) where details on a MIDI keyboard were stored.

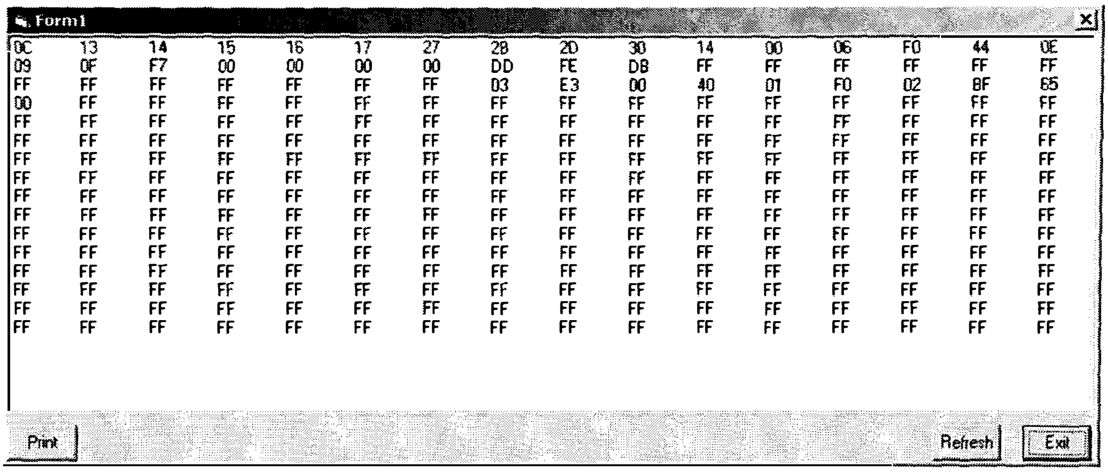


Figure 3-34: EEPROM Reader (2000)

MICROCONTROLLER PROGRAM UPGRADING: The final microcontroller could be reprogrammed from its ISP bus or through one of its two UARTs. This meant that each EMI could be reprogrammed remotely from a computer via the KAAN network.

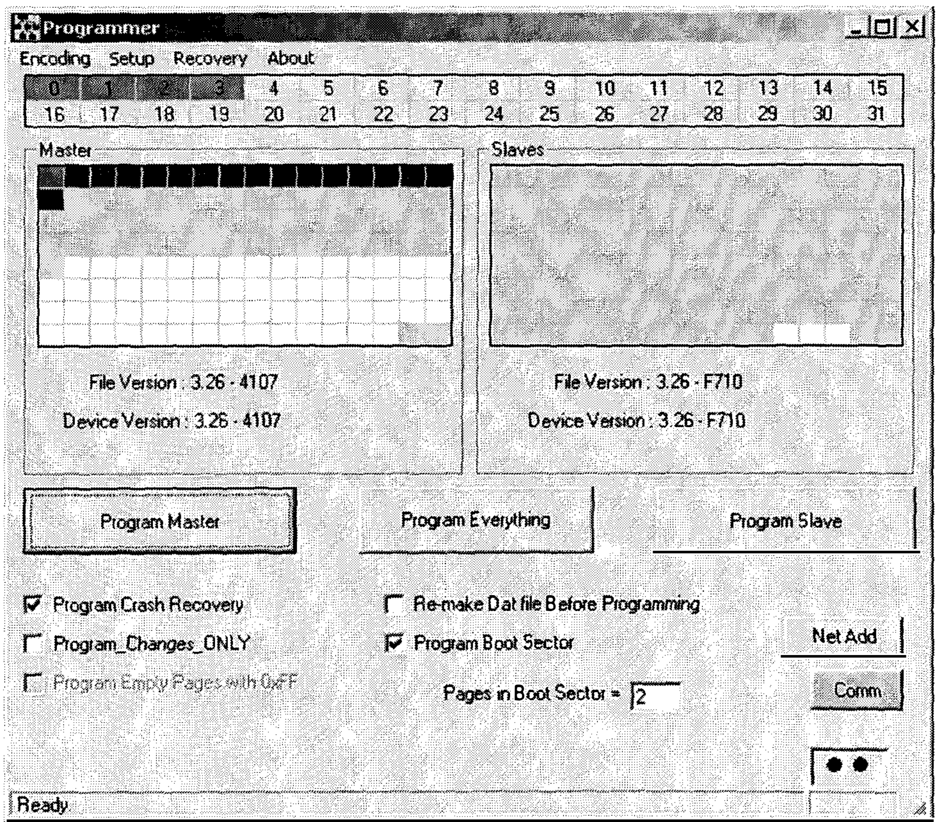


Figure 3-35: EMI re-programmer (2001)

The last utility shown in Figure 3-35 allowed the encoding and encryption of an Intel hex file generated from the microcontroller source code compiler. Then it could check the current version of the firmware of the Master and all the slaves against the last encoded version. Finally, it managed to reprogram any EMI with different flag settings.

3.5. Music Technology System Installation and Maintenance

EMIs were contracted to a local contractor, who populated the PCBs and assembled the PCBs in metal enclosures. The microcontrollers were dispatched ready-programmed with security lock bits set to secure the source code. The contractor was asked to test each EMI (100% testing) to enable them to fix bugs after a test failure. A software tool was produced to test the EMIs as shown in Figure 3-36.

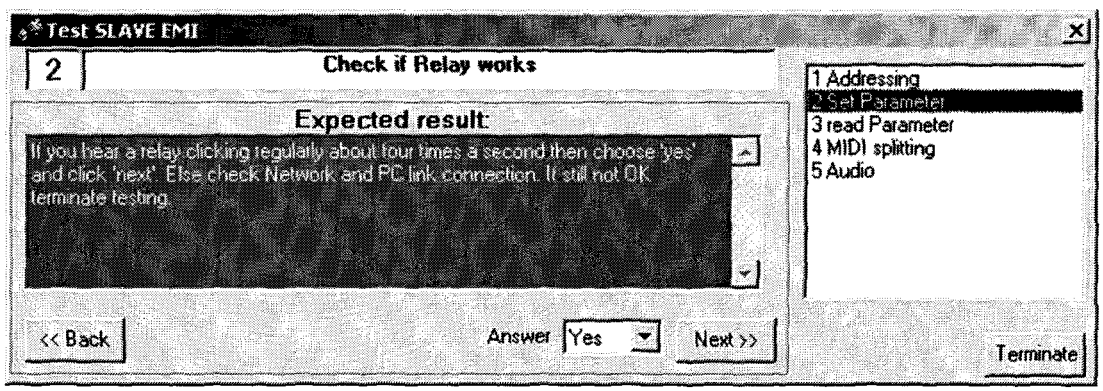


Figure 3-36: Contractor's test program (2001)

Before each music technology system was installed, a networking test was performed 'in-house' with the required number of EMIs plus 4 spares and 2 masters using the debugger utility. EMIs were also separately addressed and labelled to allow the execution of this test on site as soon as possible.

A plan of each classroom was needed prior to an installation to agree the music technology system display. EMIs had to be installed in sequence for the grouping (relay splitting).

A special power supply was designed by another contractor to supply 13.8V dc at a high current. This contractor installed this power supply and installed the power network. The power supply had four way 13.8V dc (controllable) output at 5A max. The power supply had a switch on the front panel and usually installed next to the teacher's desk. The power supply could also be switched on remotely using a relay and commands from a serial port.

Every EMI was installed where needed (in the main classroom, under a desk, or on a wall, or in practice rooms). The two networks were installed and the power net was tested as well as the communication. Then keyboards, mixers as inputs, and headphones (with an extension lead for a teacher headphone) were fitted and connected to the EMI using specially made cables (power, audio and MIDI). Then, each station controlled:

- Keyboard power.
- Audio outputs to headphones.
- MIDI splitting.

Cables were hidden inside trunkings to tidy up each station.

If maintenance was required:

- Network:
    - The debugger could determine if EMIs were not responding well (one EMI or all EMIs after the failed one) to eventually replace it or check the network cables.
    - The music technology system could be readdressed using either the debug interface or the address assigner.
    - The music technology system could be upgraded using the EMI re-programmer.
- Stations:
    - If a problem occurred on the power output to the keyboard or a fuse blew, the cover could be removed without unplugging any cable to replace the 1A fuse.
    - If another keyboard was to be connected to an EMI, the EMIs EEPROM could be re-programmed from the debug interface.

3.6. Chapter Discussion

This chapter explained how the main components of the new KAAN system called EMIs were designed, tested, manufactured, installed and maintained. The creation of a new communication protocol to control every EMI is also documented.

The specifications were reviewed as follows:

S1: To process MIDI messages from a portable keyboard in real time, to split it and to control it. **(R1, 2a, 3)**

Specification **S1** was fully implemented as a microcontroller processed MIDI messages in real-time. MIDI real-time processing enabled to split a keyboard into two as messages coming from the keyboard were filtered and dispatched to two distinct channels for left and right students. In firmware, a special command 'Keyboard Mode' was created to change to solo, split or duet modes. EEPROM programming allowed the use of any MIDI compatible keyboard (6 at the time of writing) by customising the MIDI filtering. Any 'reset' MIDI messages indicating an action on the keyboard panel while in split or duet modes, made the microcontroller reset itself as well as the attached keyboard to control this keyboard.

S2: To process audio locally to improve audio quality and to dispatch the resulting audio signal to the appropriate student. **(R1, 2a, 4)**

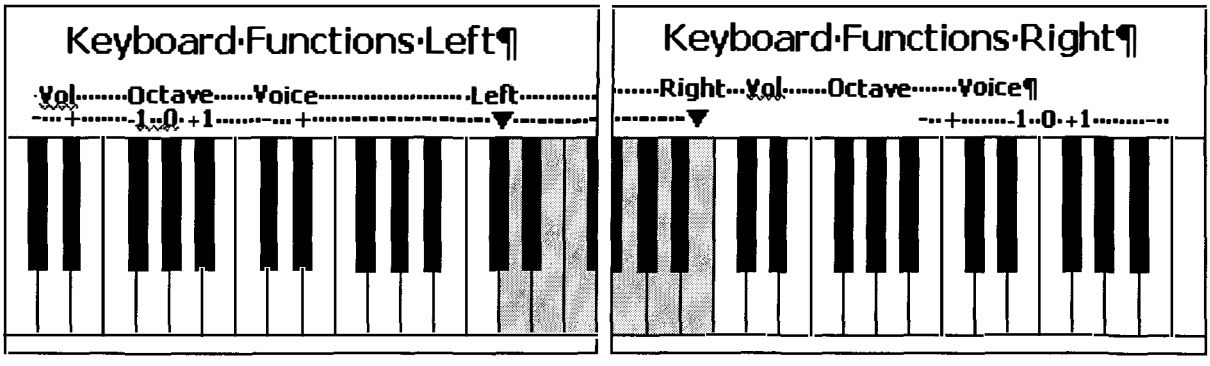
Specification **S2** was fully implemented using Design 4 after comparing four possible designs. Local audio processing consisted of local audio mixing and dispatching of eight mono inputs to six mono outputs. Audio Quality was assessed as good by audio experts at the collaborating company as well as by teachers and audio presented neither hiss nor cross talk. In firmware various commands were created to control the audio processing, such as Mute, Mix, Volume, etc. commands.

S3: Transmit and receive audio to or from an audio network to allow stations to be grouped together. Nodes to be used stand alone or networked and remotely controllable. **(R2, 3, 5, 6, 7, 8, 9, 10)**

S4: To be controllable remotely (from a computer or another controller) using the existing network(s) or another digital network. **(R2b, 5, 8, 9)**

Specifications **S3** and **S4** were implemented using Design 4 to network analogue audio signals. Four pair twisted cables were used and offered three or four buses for audio depending on the network configuration. One bus could be split for grouping to allow for more groups. However, this method had a constraint and only contiguous stations could be grouped together.

Stations could be used stand alone or networked. Stand alone control was performed with the use of local keyboard functions. Teachers or students could (if enabled) change their keyboard mode, instrument, volume and transpose level using key combinations captured by the EMI. Key combinations are shown on Figures 3-37.



Figures 3-37: Keyboard functions (2001)

A new communication protocol defined around RS485 standard was created to remotely control each EMI separately or the entire music technology system at once. The remote control could be performed from a portable keyboard or a computer using one out of the four available buses (Network configurations 1 and 2). However, an existing LAN could be used in parallel to KAAN network to link

CHAPTER 4. SYSTEM SOFTWARE

To help music teachers, a new music technology system called KAAN was created. The hardware and firmware parts of the new KAAN system were described in Chapter 3. This new music technology system was created to be used stand-alone and could control a MIDI keyboard locally or to be controlled remotely using the teacher's keyboard or a computer as control device.

If the new KAAN system were controlled by a portable keyboard, key combinations would allow the teacher to setup every station. The Master and Slave EMIs handled MIDI and RS485 protocols in this case. To control the music technology system by a computer, a user interface had to be designed to interface with the music technology system. In that case, the Master EMI was connected to the computer through a serial communication link and communicating using a RS232 interface standard. If no control device was present, the music technology system still could be used and teachers could set EMI parameters locally with a set of predefined key combinations (see 3.6, page 111).

This chapter explains why the computer solution was selected. Then, the chapter describes the creation of new ergonomic system software to remotely control the system hardware described in Chapter 3. It also describes the creation of a second new system software called ESAAMS. All source code is propriety of the collaborating company and is not in the appendices.

The music technology system commands offered by the new KAAN protocol were Audio, MIDI (with a MIDI portable keyboard) and Other Functions.

If the control device was a portable keyboard, then most of the audio functions could be controlled remotely. The music technology system would be controlled through the detection of the key pressed on the portable keyboard. A standard for the use of the portable keyboard keys was defined to access these functions, and ergonomic ways of displaying these functions to the teacher were considered. This method of controlling the music technology system proved not to be user-friendly because no visual aid helped the human-machine interaction. The method was also limited because it only offered control functions. Moreover, students' work was still recorded onto tape, unless a computer was connected to the music technology system to record work digitally (as MIDI, wave, mp3 or wma files).

In the case of a computer-controlled music technology system, a new user interface had to be designed to interface with the system. Students' work was then stored digitally on the computer.

The computer approach was selected and created as it offered more flexibility and was more user-friendly. Moreover, other software and administrative utilities could be installed on the computer for the teacher.

Microsoft Windows Operating Systems (OS) was selected because of the number of computer users utilising this OS. Microsoft (MS) Visual Basic 6 (VB6) was chosen to write the base software as it offered:

- Microsoft Windows Graphical User Interface (GUI) facilities.
- Object Oriented Programming (OOP, see 2.2.4 page 38).
- Easier to use interface than MS Visual C++, and more focused on the GUI aspect.

It was decided to continue the research with the creation of new computer control software to interface with the new KAAN system, using VB6 development software for Microsoft Windows. Note that at this time (year 2000) VB.NET did not exist yet, which would have been chosen otherwise.

Specifications for the music technology system were considered to define the interaction to be used to interface with the new KAAN system.

4.1. Specifications

The music technology system and the control interface were going to be used by music teachers. These music teachers were to use the new music technology system every day all day long. They may also have little time to train and practice on the new music technology system.

Also the new music technology system would have to be stable not to disrupt a lesson, and easy to use and debug by the teacher in case of a failure.

From initial discussions and thoughts specifications were defined like in Chapter 3. However, in this case just a few requirements were asked from teachers, which were to get a user-friendly system software with which they can work with other music software or management and assessment tools. After observing how teachers worked, other specifications were recognised as being relevant for the design of the software. These specifications were (as shown in Table 4-1):

S1: **Control the new KAAN system** by sending commands from the computer to the KAAN system via a serial port. The *Transceiver* control described in 3.4.3 (page 98) was to be used to perform this task.

S2: **Easy to learn and to use:** Teachers would have little time to learn how the system worked. The interface had to capitalise on the teacher's pre-existing knowledge of interfaces in order to be used as naively as possible.

S3: **Attractive and Meaningful:** Teachers would have to use the GUI everyday (and sometimes all day long). The GUI had to be simple, and non-disruptive with the whole set of functions in sight for quick interaction. Prototype GUIs were reviewed and tested on site with teachers to determine issues and improvements so that other teachers, when evaluating the product, got an appropriate feeling of its usability.

S4:	**Adaptable and Upgradeable:** Other functions could be added as software upgrades and updates.
S5:	**Fast Interaction:** Teachers needed to interact quickly with a student or a group of students using the interface. As lessons did not last a long time, teachers could not afford to wait for the system to complete a required task while lecturing.
S6:	**Administration and Preparation:** Teachers wanted to keep records of individuals (for example work, marks, comments, photograph, or for administrative purposes), and to retrieve the information quickly and easily. Teachers also needed to use the system outside of lessons to prepare it.
S7:	**Accurate System Control and Status Display:** The GUI had to show the state of the system and handle communication problems.
S8:	**Mobility:** Teachers needed to move around and still be able to control the system locally and / or remotely.

Table 4-1: Software Specifications (2000)

4.2. KAAN Interfacing Philosophy

A typical KAAN system user was to be a music teacher in a school. This type of user might not be computer literate and might have little knowledge of the basic MS Windows interaction philosophy, so:

- The system software had to be reliable and ready-to-use. In case of failure, appropriate interfacing was required to show and solve a fault. This was partly addressed as the system hardware had to be tested on site for quality control.
- The interfacing had to be easy and explicit, with everything in sight for a quick interaction. From the Human Computer Interaction (HCI) tools described in 2.2.4 (page 40), the following were selected:

- Iconic interactions (for a more pleasing and interactive GUI).
- Visual representation of the music technology system.
- Extensive use of tooltip texts (small descriptions popping up when the mouse pointer stops on an object) for help.
- Wizards for complicated processes.

A challenging idea was to build the application without a help section. The application had to be explicit and easy to learn and it was decided to produce a user manual and to provide training. Tooltip texts popping up when the mouse pointers hovered on controls were broadly used for all controls to explain their purpose as an alternative.

Another method was tested to provide longer help tips using a 'help label' as shown on Figure 4-1.

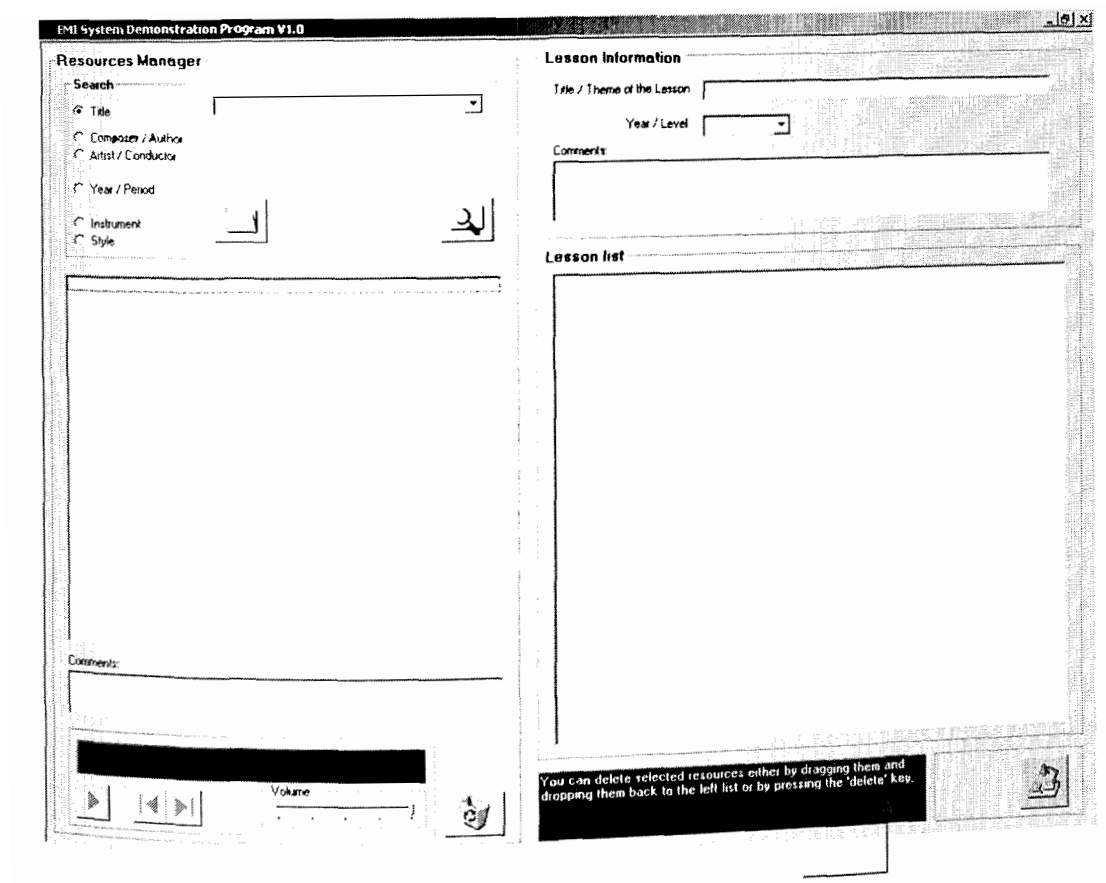


Figure 4-1: Dynamic visible help (2000)

It was decided to offer access to functions in multiple ways because different users would interact differently with the interface. User and context sensitive

control management was used to offer only useful or authorised functions to the user, according to a user profile. Functions were accessible from:

- A toolbar.
- Menus.
- 'Right-click' popup menus.

4.2.1. Selection

Teachers had to interact quickly with the computer at the teaching position while lecturing in order to stay focused on the lesson. This interaction had to be bi-directional, the teacher being able to listen to a student or a group of students, to speak or play audio to them and to record their work.

A quick way was to represent the students or their stations and to have means of offering the available functions. A method was station centred, where functions were applied to stations. Other methods might be student or class centred.

The selected method was both student and station centred. Students were placed on stations representing the music technology system layout. Students or stations were selected to receive determined functions, as for instance:

Function Mute and Select this Student,
or Select this Student and Function Mute. } **Mute this Student.**

This meant 'select Station 13 then Mute it', or 'select John on the left side of station 2 then set the player for him'. This interaction philosophy implied that:

- Students and/or stations could be accessed.
- Functions could be found.
- The status of selected stations was shown to confirm a function had been applied successfully.

4.2.2. Classroom layout

A 'Main Display Space' used most of the space in the main background window to interface with the music technology system. Toolbars, menus and status bars were used on the sides to offer functions.

A method to interact with the music technology system was to display it as it was, as shown in Figure 4-2. Student station representations (desk + EMI + keyboard) had to be placed on the screen to represent the physical layout in the classroom. The teacher station representation (desk + EMI + computer + keyboard) was placed relative to the student stations to give a visual reference. This reference was usually at the bottom of the Main Display Space.

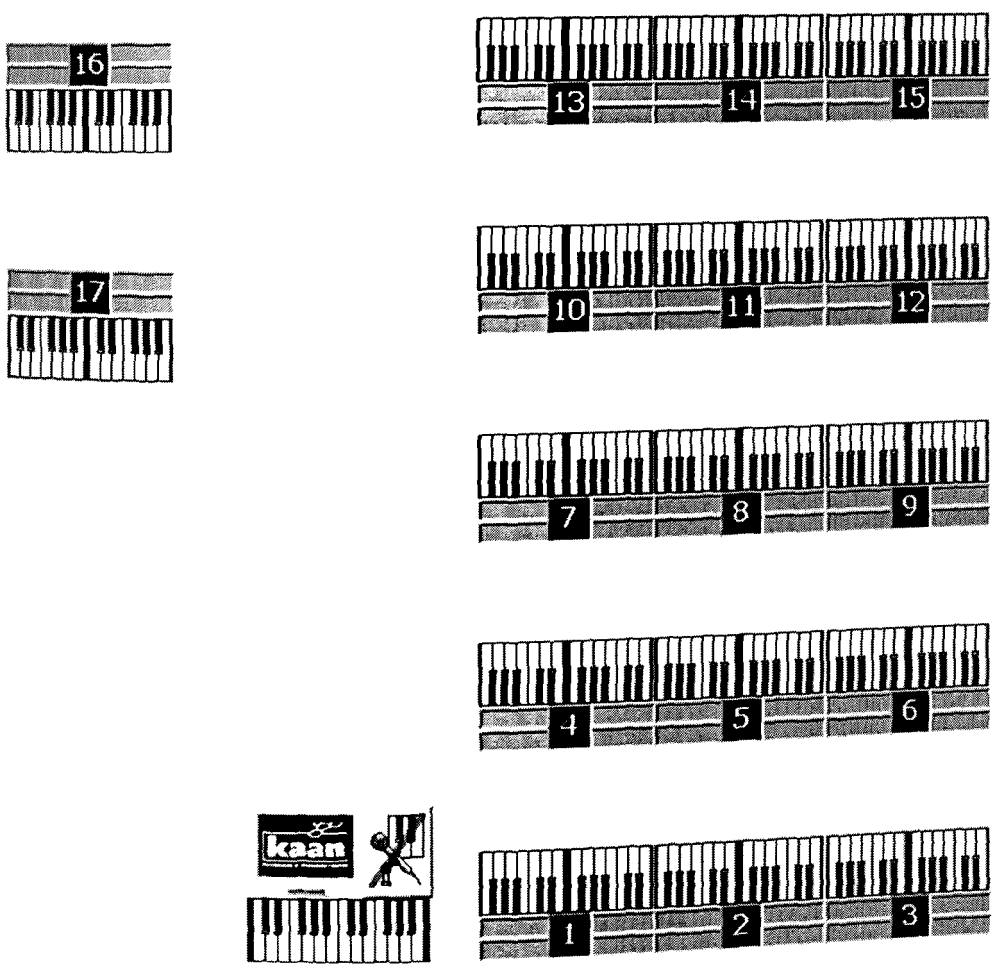


Figure 4-2: Classroom layout display (2001)

The station representations displayed the status of their physical EMI counterpart in real time, as well as functions applied to it (player, recorder, group, etc.). Other displayed information are shown in Figure 4-3:

- EMI address (physical station could also be numbered for easier retrieval).
- Keyboard mode (solo/split/duet).
- Group attached to and grouping side (to quickly view stations in this particular mode).
- Student information (for students sitting at the station).

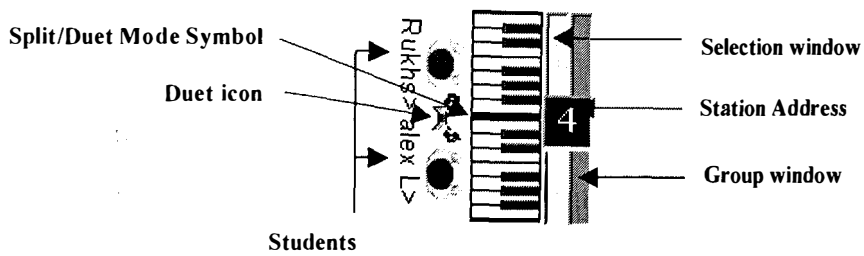


Figure 4-3: Student Station Representation (2001)

The teacher could be warned unobtrusively about problems with the music technology system using the station display. Two types of faults could occur as shown in Figure 4-4:

- Control problem on one command or a set of commands (intermittent).
- No communications available with the EMI (could not even ping back).

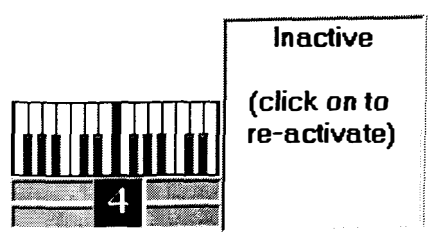


Figure 4-4: Stations with communication problems (left with control problems with a red background instead of black, right with lost communication) (2001)

In both cases, teachers could try to resolve the fault or inactivate faulty EMIs if necessary. If communication was lost, teachers and students still had to be able to work with the music technology system.

A teacher station was not set like a student station and needed to be different from student stations for faster referencing. Moreover, the teacher could listen and speak to students from the front using a headset. This function had to be quickly accessible as it could be used a lot. The teacher's station was designed as shown in Figure 4-5, with a toggle switch to set the Speak function.

Figure 4-5: Teacher Station Representation with a Speak button (2001)

KAAN systems could have different layouts, which needed to be set on site. This system layout process was sequential therefore a wizard was designed to help perform the task. The classroom layout wizard (shown on Figure 4-6) consisted of 4 steps:

- Give the number of slaves in the music technology system (only 1 master).
- Place the stations in the correct orientation and position, and save the layout after centring it on the Main Display Space.
- Re-address the physical EMIs correspondingly to the set layout. EMIs were set in an addressing mode and waited to receive a MIDI message (generated by a key press on an attached MIDI keyboard) to set a network address.
- Program the EMIs' EEPROM against attached keyboard.

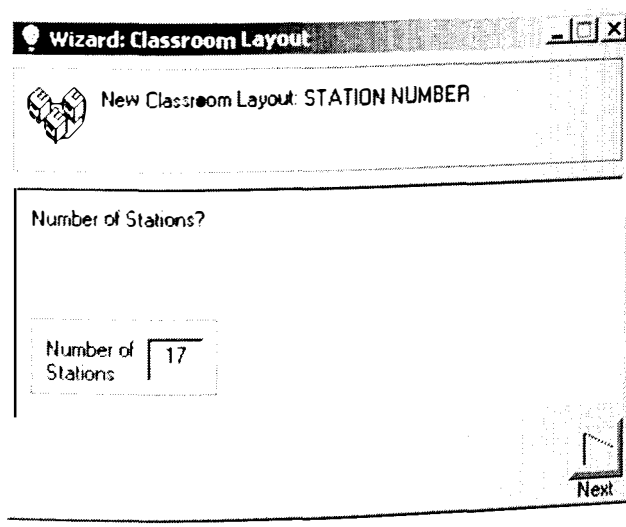
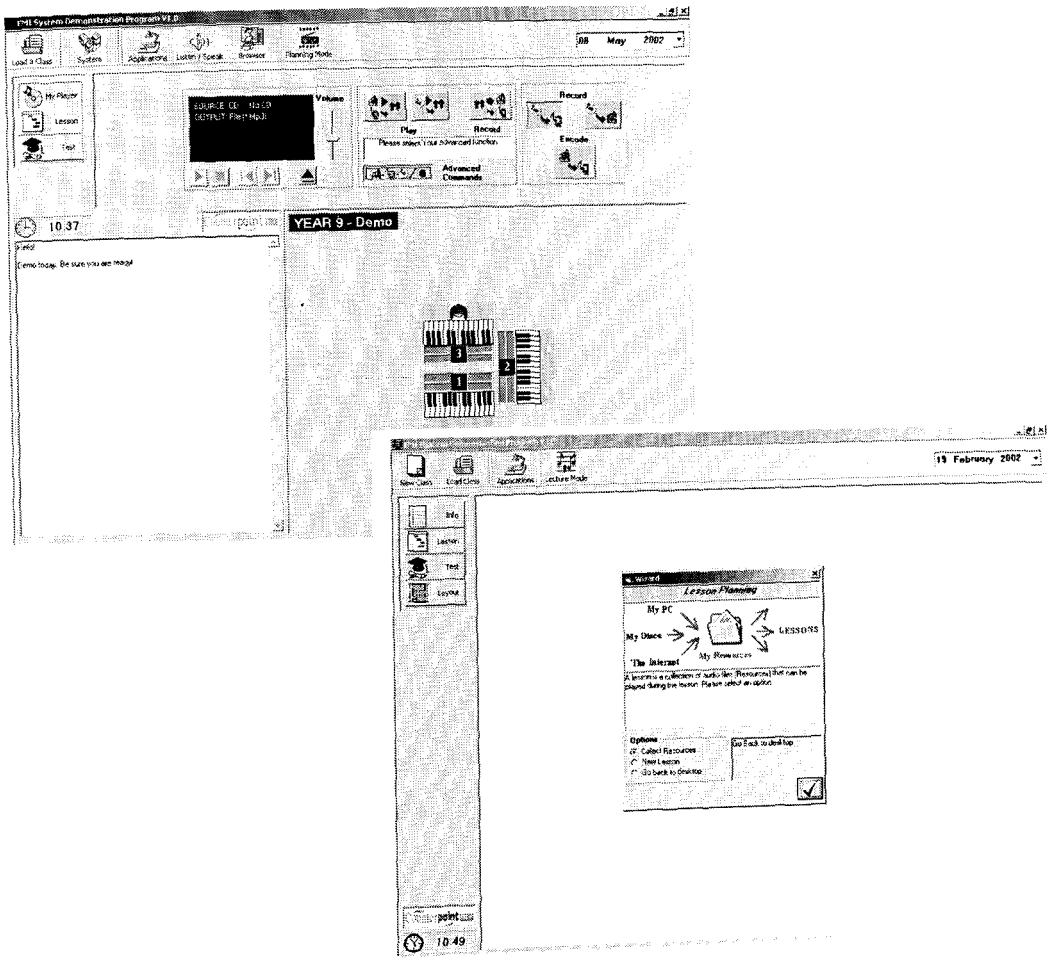


Figure 4-6: Classroom layout Wizard (2001)

Many GUIs were reviewed and tested with teachers to create the most intuitive interface for the music technology system. Wizards were broadly used with the first tested GUIs, as many tasks were sequential and difficult to handle (create classes of students, place these students on stations, group station together, etc.) On the first GUI, a space on the Main Display Space was dedicated for wizards to process tasks more quickly (see Figures 4-7).



Figures 4-7: First interfaces (2001)

On certain functions, wizards did not help process the task; they were deemed to be too rigid and slowed down operations. For example, 'student placing' function could be performed more easily and efficiently using 'drag-and-drop'. Only tasks requiring feedback from the user were then kept as wizards, other tasks were implemented with other methods. Now that the interfacing method was defined, the core control engine had to be created.

4.3. Core Interfacing

The new software system was designed to control and display the status of the KAAN system. The new communication protocol allowed each EMI to be set and read by a control device, here a computer. The new GUI had to enable teachers to set either one or all stations in the music technology system, and to display accurately the real state of the music technology system. Therefore the control structure of the interface had to be considered.

4.3.1. Control structures

Each physical station was represented with an object. This object stored and also displayed the status and set the parameters of its designated station. The object also responded to a mouse click or other events. There were as many objects as stations in the music technology system, and the graphical representation of these was manipulated to represent a classroom layout to assist in the identification of a station or student. These objects were called *Station Representations*.

A required object had to handle communication with the music technology system through a computer serial port. This object was called a *Transceiver* (see 3.4.3, page 101). There was one *Transceiver* per music technology *System*.

In a first control structure, the *Transceiver* was created when the program started. The *Station Representations* were then loaded from a file. After being loaded, the *Transceiver* "pinged" the music technology system to check the physical presence of the represented stations. The *Station Representations* were disabled if their designated physical station did not respond. Stored class settings were also loaded. Commands were given to the *Station Representations*, which were passed to the *Transceiver* to send to the music technology system stations. Teachers could adjust settings on the stations at any time through a setting window, which was activated by a right mouse click on a *Station Representation* (see Figure 4-8). The status was stored whenever the *Transceiver* was asked to send the message to the music technology system.

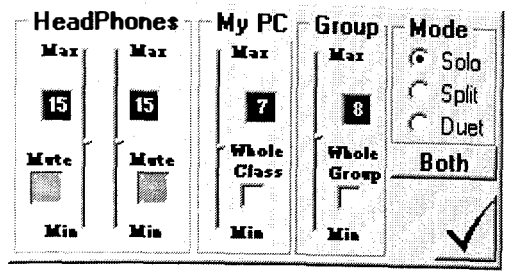


Figure 4-8: Control window (Student Station, 2001)

Teachers also needed to set the whole music technology system at once (for example Mute All). This was achieved through a generic module shown in Figure 4-9, which sent a command to every *Station Representation*.

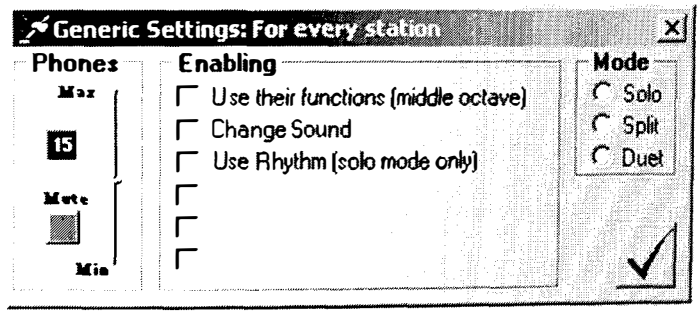


Figure 4-9: Generic Control Window (2001)

The object interaction of the first control structure is displayed in Figure 4-10.

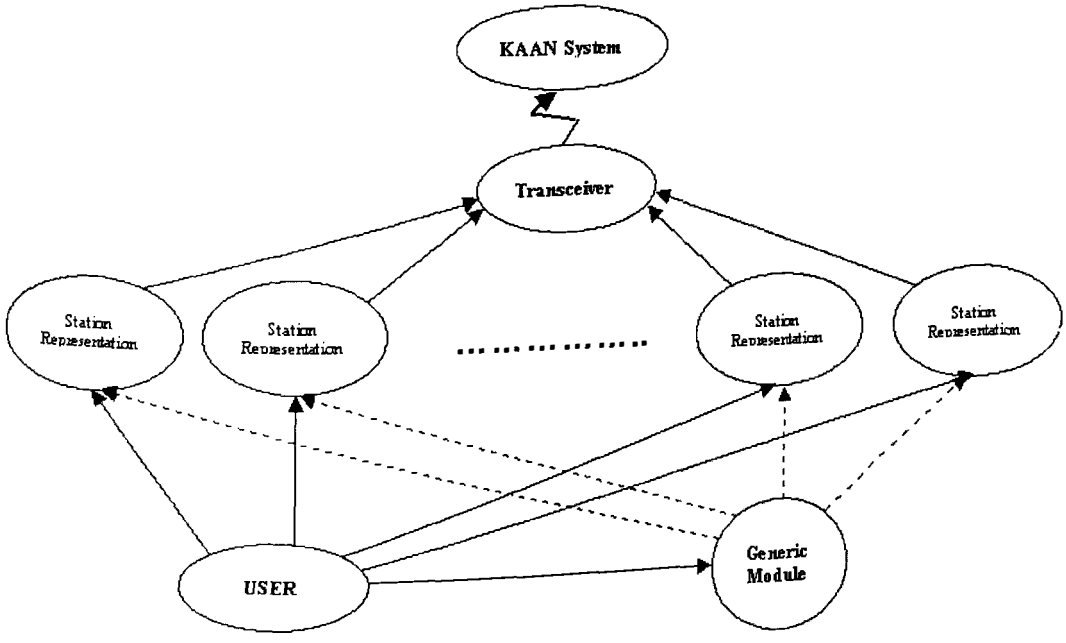


Figure 4-10: Structure 1 Control Flow Diagram (2001)

The first control structure had many limitations:

- *The structure did not take advantage of the communication protocol:*
  For generic (global) commands, there were as many messages as there were *Station Representations*, instead of only one. This could overload the music technology system and messages could be lost.

- *The control flow was not homogeneous:*
  The user had to control the music technology system through different entities (*Station Representation* and the generic module).

- *The control system had no feedback about the real state of the music technology system:*
  Station Representations could be disabled at the start-up if the corresponding physical station did not respond. If a station was to respond afterwards, neither the control system nor the user were able to restore the communication with the node. Moreover, if messages were lost, the *Station Representation's* status could be wrong. A user could not set the music technology system without accurate information!

A second control structure was then considered that could take advantage of every aspect of the protocol in order to minimise communication flow and get accurate information on the music technology system status.

In a second control structure, *Station Representations* were still needed to interface with the music technology system. The *Transceiver* was still required to handle communication.

Since the stored *classroom representation* could differ from the real configuration of a music technology system, it was considered better to have two objects per station, independent from each other: one object to get the status and set the real EMI, the other to deal with the display and interfacing with the user (see Figure 4-11). The first object was called the *EMI Object*, and the other the *Station Representation*.

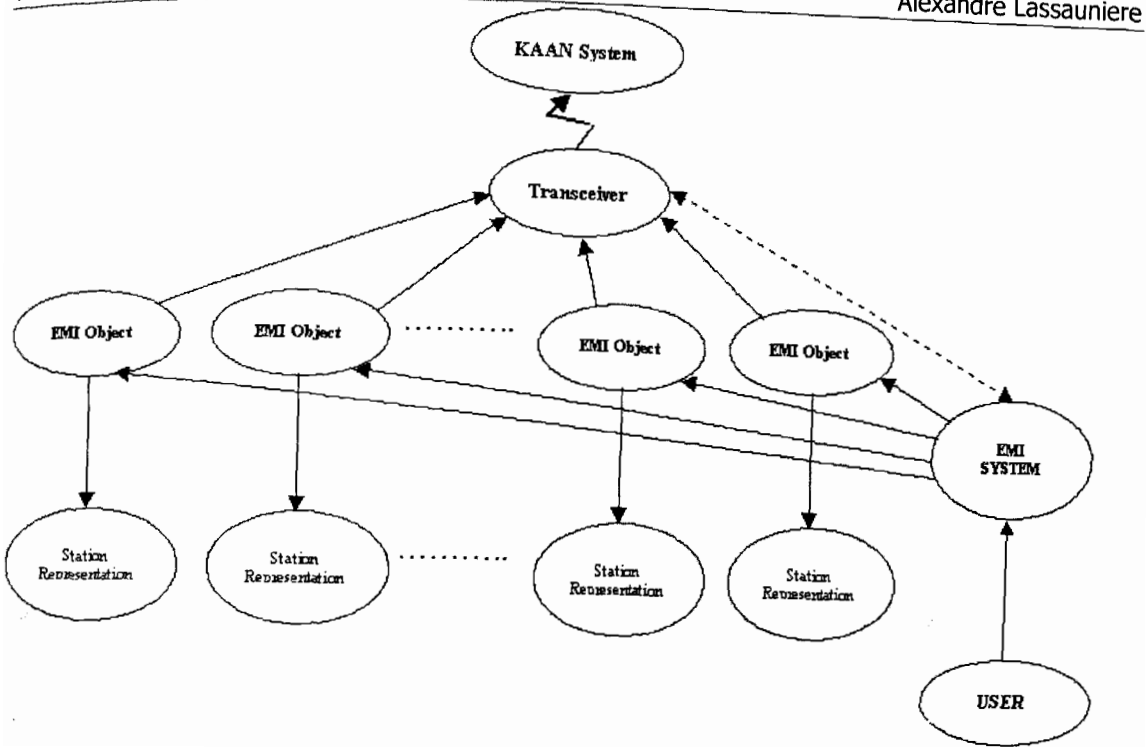


Figure 4-11: Structure 2 Control Flow Diagram (2001)

EMI objects were created at start-up and stored in a collection called the *EMI System*, after the music technology system had been scanned to determine its configuration. Each *EMI object* interrogated its designated physical station to determine its state. The stored *Station Representations* were then linked to the corresponding objects, if these objects existed. Any new physical station instigated the creation of a new *EMI object*, which was then linked to a corresponding *Station Representation*.

The *EMI system* object controlled all communication. The *EMI system* object belonged to the *Transceiver* and there was only one *EMI system* object per control system. Every command was processed by either an *EMI object*, or by calling any of the global settings (which were global representations of the *EMI Object* addressed settings).

Commands were sent from *EMI Objects* or the *EMI System* to the *Transceiver* to be sent onto the network. The *Transceiver* sent a message back to the *EMI System*, which told the appropriate *EMI Objects* to change their status. The *EMI Objects* then raised an event to their *Station Representation* object to change the display. With global messages, the *Transceiver* sent an event to every EMI object

to update their status. This method was similar to the first structure, in that the *EMI objects* sent messages through the *Transceiver* to a physical station, without having any confirmation that the message arrived.

The second method was more versatile since a verify test could be performed by the *Transceiver* after sending a message. This verifying tool was used to establish if there was a problem and used to initiate the resending of messages.

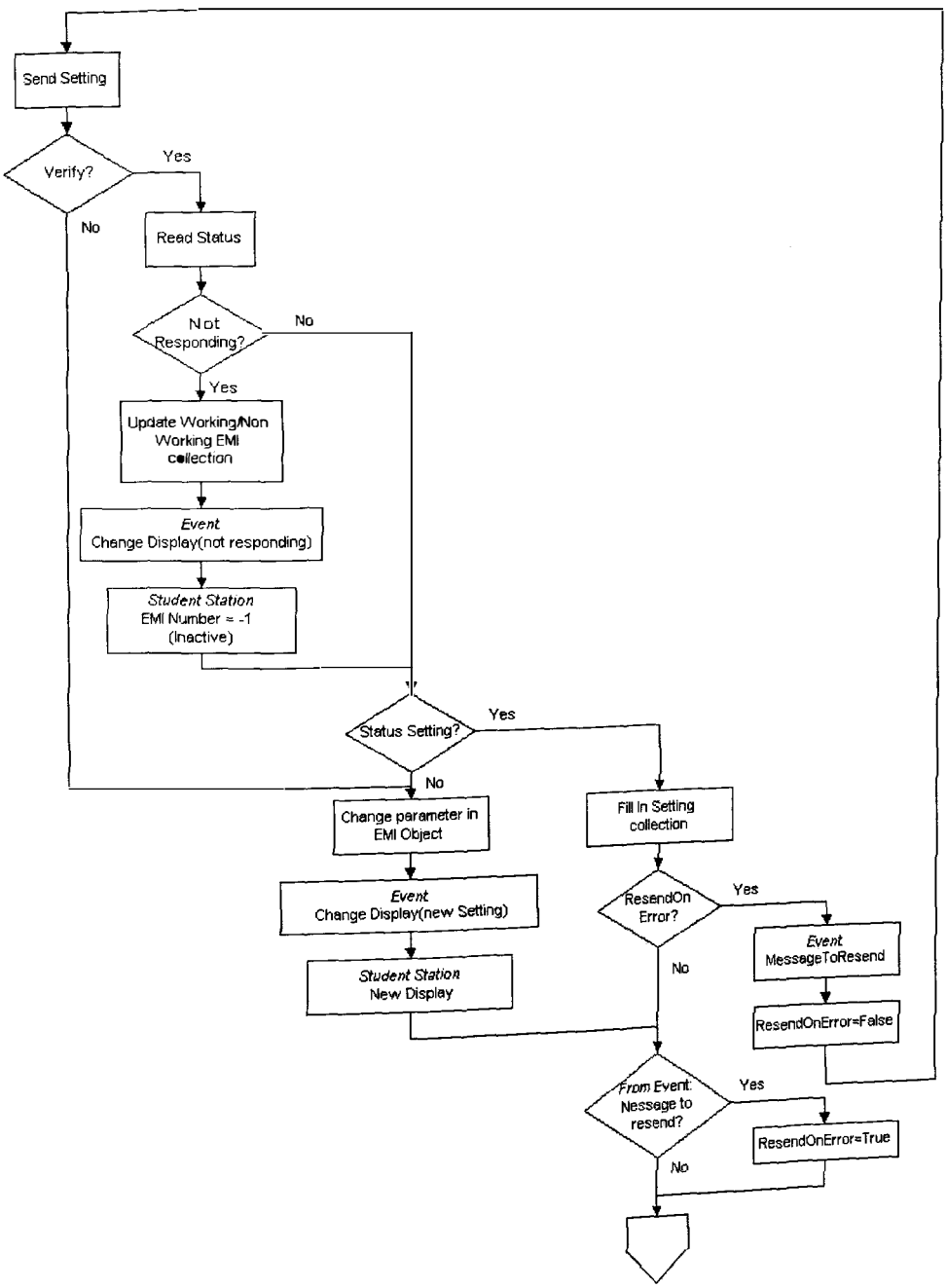


Figure 4-12: Single Setting with Error Checking (2001)

EMI Objects could determine whether a parameter had been correctly set or not. If a parameter had not been set, the message could be resent (as seen in Figure 4-12). If the problem remained after resending the setting, it could be assumed that communication had been lost with the physical station and the *Station Representation* could be disabled to show the user that a communication problem existed.

Station Representations visually indicated when they were in a 'communication problem mode' so that the user was aware of possible communication problems. The user could click on a 'not responding' *Station Representation* to 'ping' its EMI to check if the fault had been corrected.

A startup routine checked the physical system against the stored system layout to determine if EMIs were not working or even if there was a music technology system attached to the computer. Some debugging interfaces (shown below) were offered to the user to be able to use the music technology system correctly.

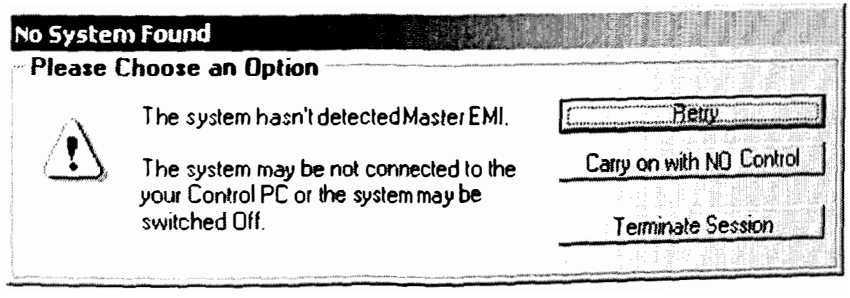


Figure 4-13: No System Found (2001)

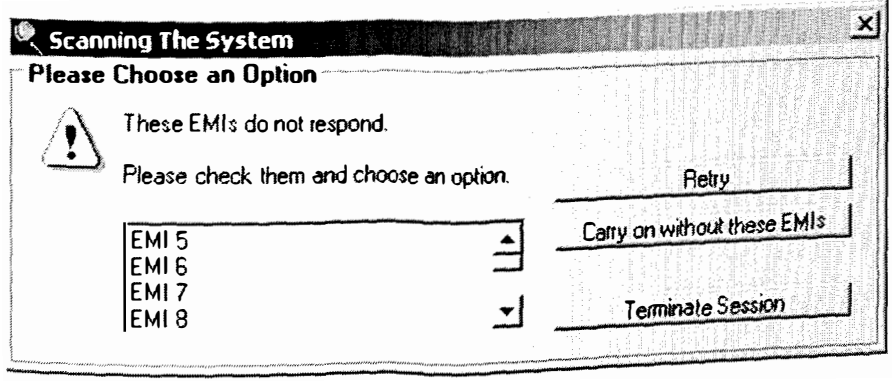


Figure 4-14: EMIs not responding (2001)

Figure 4-15 shows the interface with no control. This was used to interact with the new software system with the new KAAN system switched off or if it had crashed. Control functions were disabled and the interface was 'disconnected' from the control side. The user could try to retrieve control at any time by closing the 'No communication' window.

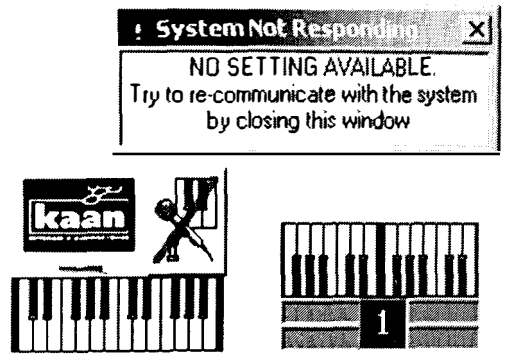


Figure 4-15: System with no control (2001)

The above routine explained what to do if fewer EMIs than initially set were responding. But EMIs could also be added to a KAAN system, just by connecting power and network leads. Control structure 2, with its verification process, allowed:

- Capture non-responding EMIs.
- Capture non properly working EMIs.
- Capture new EMIs (if their address was not used already in the 'static' music technology system).

New stations detected appeared and could be placed anywhere on the Main Display Space (see Figure 4-16). When removed from the music technology system, there was no reason to set them as inactive because they were considered as 'dynamic' stations, and were removed from the Main Display Space.

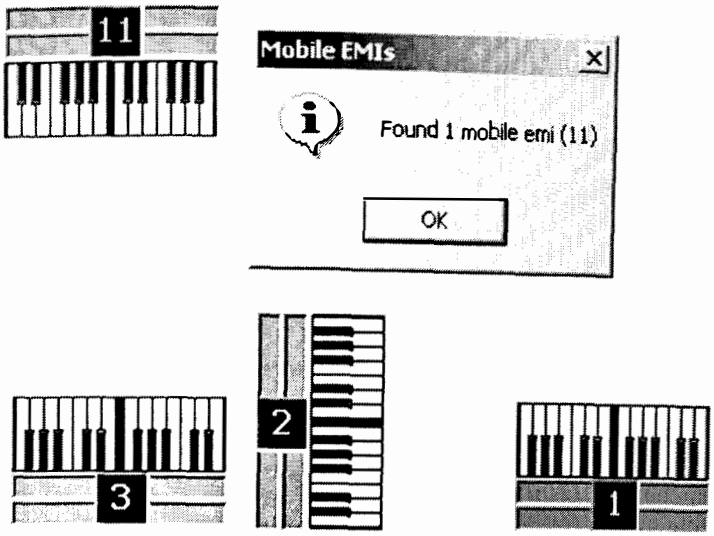


Figure 4-16: Mobile EMIs (2001)

Now that the core control engine was defined and tested, useable setting tools needed to be selected for the user.

4.3.2. System setting

The teacher could remotely control any EMI with commands defined in the new KAAN protocol. The music technology system setting interfacing with the user went through different stages to determine the best way to access these commands. These commands had to be set quickly to any selected station. Teachers also needed to read the correct status of the music technology system (or selected stations).

Below is discussed the evolution of setting tools through different versions of the GUI.

TWO SETTING WINDOWS FOR CONTROL STRUCTURE 1

This GUI was based on control structure 1 and had two different windows to set the music technology system (see Figure 4-17): A local one called by right clicking on a station and a generic one called from the toolbar to set all EMIs at once.

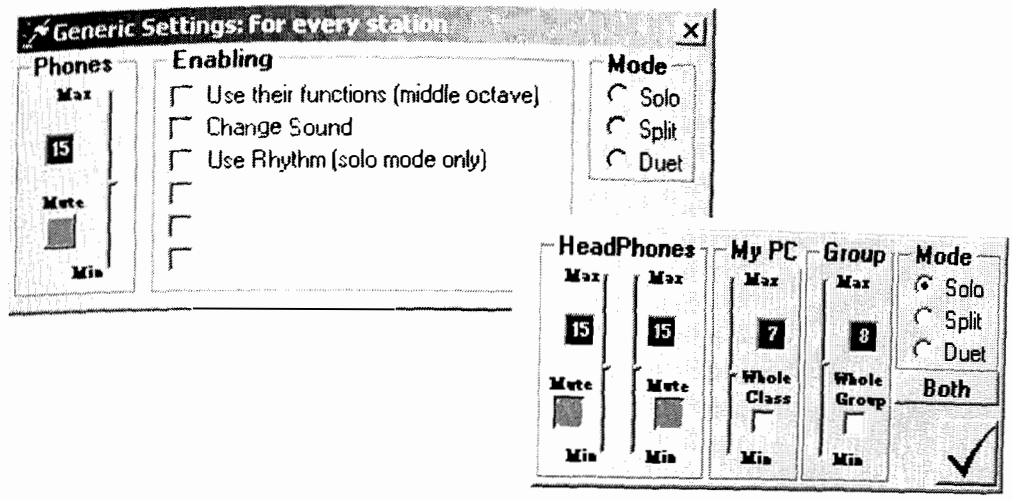


Figure 4-17: Early setting windows (Generic and Local settings, 2001)

A MOVEABLE, 'ALWAYS ON TOP' TOOLBOX FOR CONTROL STRUCTURE 2

As control structure 2 replaced the first control structure, a new interfacing tool was needed. Teachers set and read EMI status from a moveable toolbox that remained visible (see Figure 4-18). A selection manager handled selected EMIs and interfaced with this toolbox. To control any side of a station, or group, the user had to click on the correct part of the station representation.

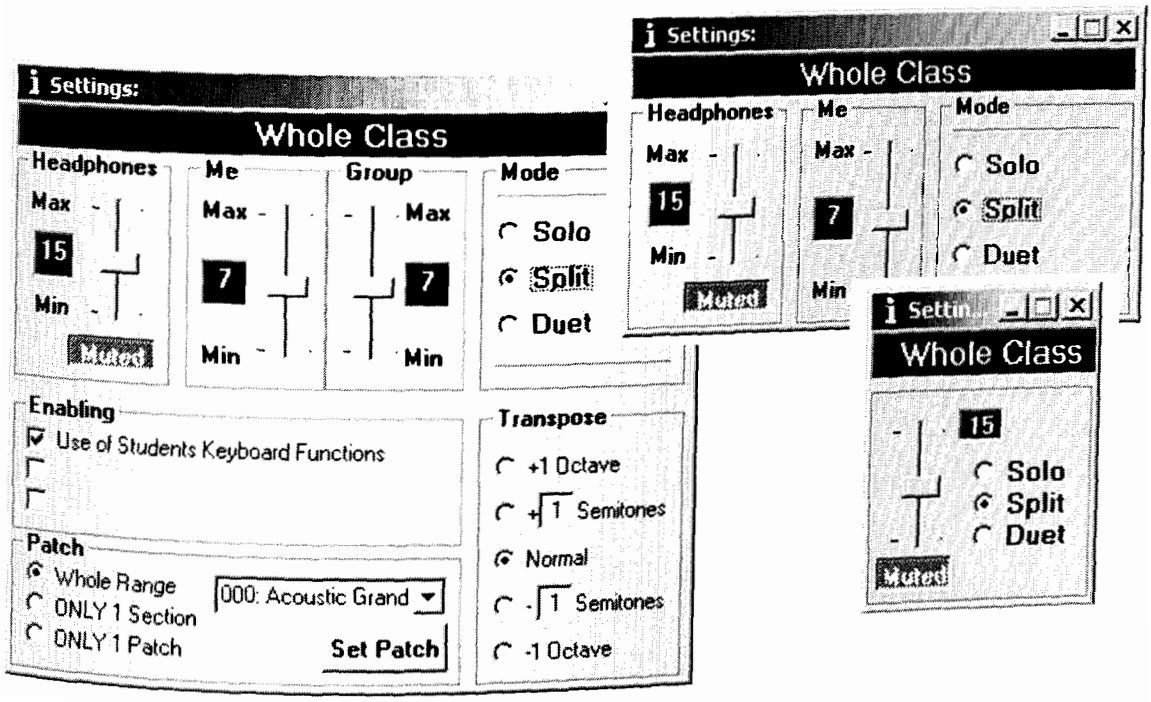


Figure 4-18: Moveable, 'Always on Top' toolbox in its three display modes (2001)

The whole class was selected either by default when no station was selected or by clicking on the class name in the Main Display Space. The teacher could select multiple stations using <ctrl> and <shift> keys with a MS Windows Explorer feel. Each time a new selection was made, the toolbox refreshed the status display to show the selection's status.

The main settings offered to the teacher were the keyboard mode, the main volume and muting facility. Other settings were available such as network and group volumes, keyboard transpose level and instrument (for split/duet modes only) and flags. Three views were proposed for the toolbox because many settings were offered with different priorities.

This toolbox was not kept because:

- Additional settings meant more space.
- Too invasive on the display.
- Command centred rather than utility centred.

To counter this, a radically new interface was designed.

A RIGHT POP UP SETTING WINDOW BASED ON USER'S UTILITY

The selection method was kept as such as it proved to be efficient in a classroom environment (from beta tests on site with feedback from teachers). The difference in the setting window resided in what to set and read:

- On/Off generic commands that needed to be available with only one mouse click, such as 'Mute all', 'Speak to all', etc.
- Device control:
    - Selection based for Student stations.
    - Sound card control for the computer (teacher side).

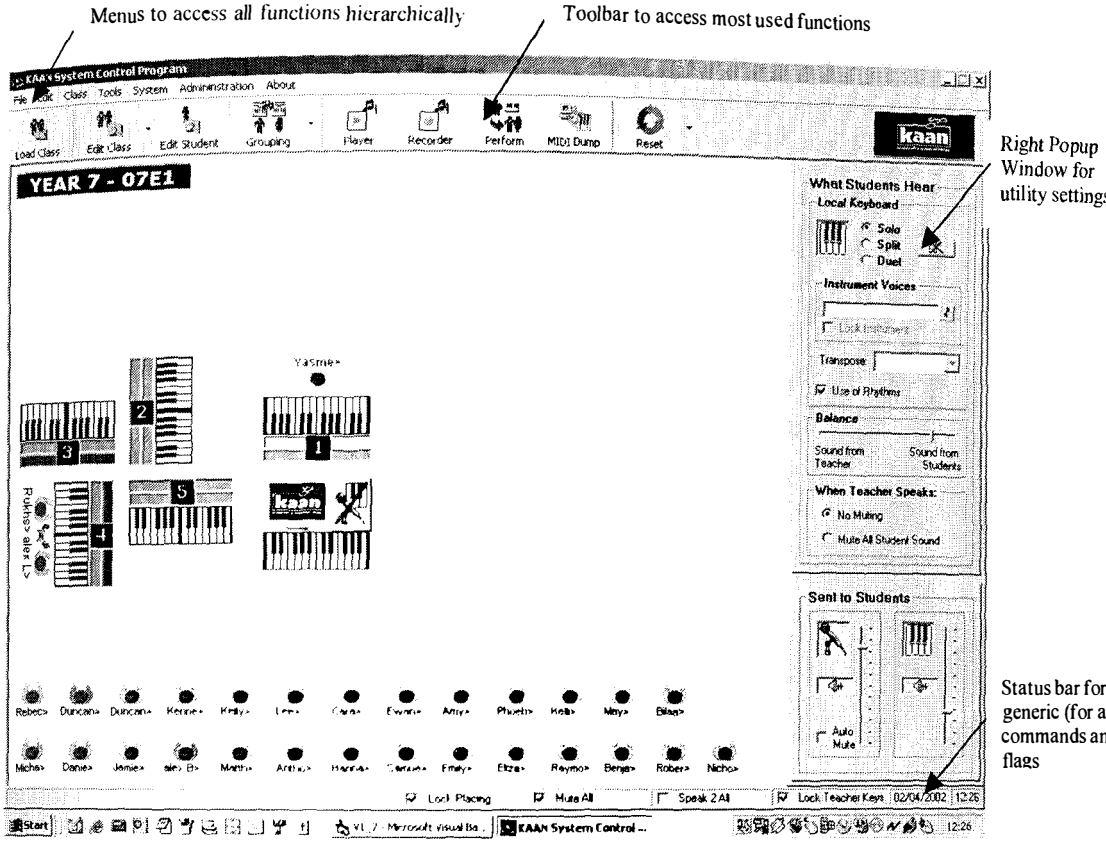


Figure 4-19: Released GUI with right popup setting window (2002)

The first group was put in a bottom window status bar and could be accessed and used quickly. The second group was implemented in a popup window. This window (shown in Figure 4-19) was separated into two frames:

- What students hear.
- Sent to students.

The idea with this kind of setting window was that KAAN was primarily an audio network and that the teacher could remotely control any keyboard linked to an EMI. This window offered EMI (and keyboard) settings as well as teacher's audio setting, and especially level balancing for listen / speak functions (the most used functions). Other speaking options (muting locally or remotely) were included to gather them together.

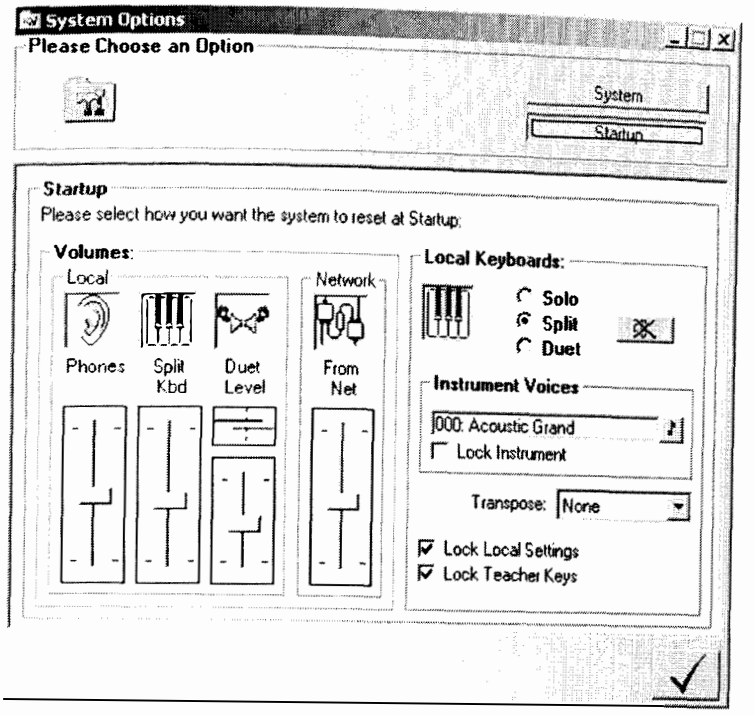


Figure 4-20: Option window (2002)

An option window (see Figure 4-20) allowed to set the music technology system with user-defined startup settings as well as rarely used functions (duet levels, buses volumes, etc.). These settings could be reset at anytime using the reset function.

The control structure and setting interface were defined, however the KAAN system offered more advanced features.

4.3.3. Main KAAN functions

The KAAN system was designed to be a controllable infrastructure in a music classroom. As seen in 3.4 (page 101), many low level commands could be sent to the students' EMI, such as mute or output volume. The teacher could use the control system from the computer at the teaching position and as a network: the teacher could then listen or speak to anyone, play a backing track or, record the students. Looking at these 'KAAN functions', these functions could be separated into two groups:

- Single commands.
- Macro functions (set of different commands).

These commands and functions could be applied on the current selection or on a stored selection. Some macro functions would need the help of wizards in case of a sequence and needed feedback from the user.

These features were:

A) Grouping.

B) Audio functions:

- Listen / Speak.
- Perform to the class.
- Player.
- Recorder.
- MIDI Sequence manager.

A) GROUPING

The KAAN system allowed the grouping of stations together so that they could hear each other, using bus 2 of the audio network (with a relay on it). Network bus 2 could be split into smaller sub-buses and used to group stations together. The only constraint was to group consecutive stations because of the configuration of the bus (see 3.3.3, page 84). This was not an issue for teachers as they usually grouped students together who could see each other.

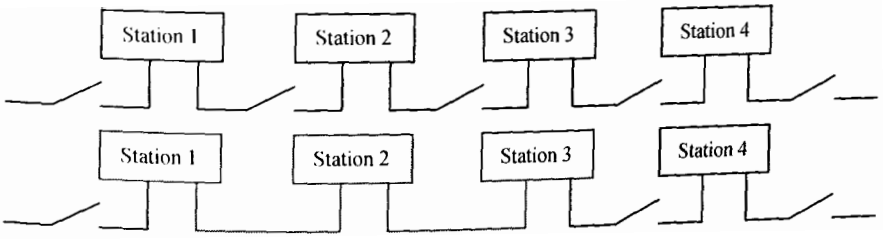


Figure 4-21: Configuration bus 2
(Top: no group / Bottom: Station 1 to 3 in a group, 2001)

The figure at the bottom of Figure 4-21 showed how to group stations 1 to 3 OR stations 1 and 3 (not station 2). To group stations together, the first step to follow was to physically link the stations together by closing the relays of all

stations between them. The next step was to define the group number and to attach them to the bus. In the above example, station 2 could be set off the group simply by not performing the last step (attach to the bus).

Figure 4-22: Grouping examples
(Left: First Instance / Right: Second Instance, 2001)

With the first instance of Figure 4-22, stations 1 to 4 could be grouped. If station 1 needed to be grouped with the station in front of it (station 2, so that the students could see each other), group 1 contained stations 1 to 2, with stations 1 and 2 speaking and listening to this group. Now, still on instance 1, if station 1 was to be grouped with the station to its left (station 3), stations 1 to 3 belong to group 1. However, station 2 was not attached to the group.

Again, with the first instance of Figure 4-22, if stations 1 and 3 were already grouped in group 1 and stations 2 and 4 needed to be grouped, a conflict occurred as station 2 already belonged to group 1. This is why the system layout was important. Stations could be grouped in two different ways:

- 'Auto pairing'.
- Customised grouping.

'Auto pairing':

This was fully automated. The control system first checked the number of stations responding (this could differ from the number of stations in the music technology system). If this number was even, the control system then grouped stations two by two automatically. If it was odd, then a wizard (see Figure 4-23)

asked how the first and last ones had to be grouped (either with the first or last group). When completed, the control system resumed to normal mode.

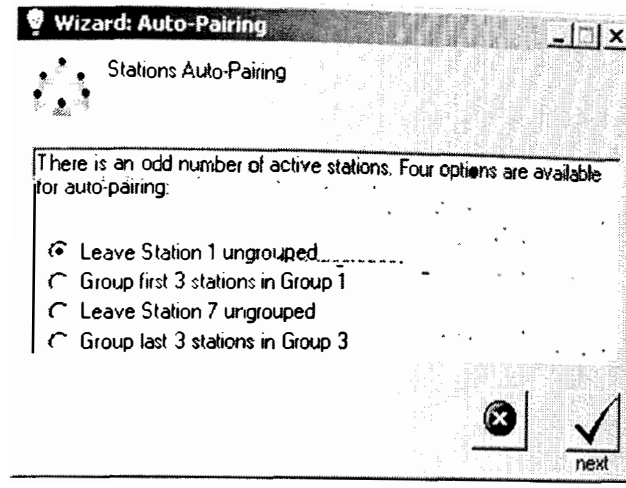


Figure 4-23: Auto pairing an odd number of stations (2001)

'Customised grouping'

Stations to group together needed to be selected first by using shift and control keys (as in MS Windows Explorer). A student, station or another group could be selected and all could be added in a new group. A group could also be selected and some stations deselected. A station could also be grouped on only one side. The control system internally checked the group number of the group to be set. The grouping procedure was initiated after the group button of the toolbar was clicked. The control system checked for any conflict and if so, asked for what to do (proceed or cancel). Then the control system set the group and the grouped EMIs were stored in the group collection for easier handling later on.

B) AUDIO FUNCTIONS

The interface controlled the music technology system remotely and displayed the current system status in real-time. But this was not its sole purpose. A teacher needed to use the control system to interact with students, which included listening and speaking, playing backing tracks or recording students' work. These functions were called KAAN Audio Functions. Their requirements (as shown in Table 4-2) were:

R1:	A teacher could listen to any student without the student noticing it. The teacher had to hear exactly what the student heard.
R2:	A teacher would listen to the students selected in the Main Display Space by default.
R3:	A teacher would speak only to selected students. Options could be set to mute the student audio inputs when the teacher spoke. Another option could be set to have the speak function always on or reset if other students were selected. The speak function could be applied on a mike and / or the teacher's keyboard. The balance between these two devices and the balance between the teacher's audio and the students' audio had to be setup easily.
R4:	A teacher could playback audio to students using a bus other than the speak bus, so that both functions (speak and play) could be performed simultaneously. The stations setup to receive playback audio had to be setup and kept in memory until they were reset. The balance between the teacher's playback audio and the students' audio had to be setup.
R5:	A teacher could record audio from students. The audio would come from any of the three Master EMI's outputs and thus corresponded to what the teacher heard. The speak function had to be disabled while recording. The setting up of the stations to be recorded had to be performed and kept in memory until they were reset.
R6:	A teacher could record in MIDI, but then the entire system would be in the special MIDI raw mode and no control was possible.
R7:	The player and recorder could be set for the same or different stations.
R8:	Students could perform to the class. Every station, even the teacher's station, would only listen to what was being performed.
R9:	If any student were performing, the player and / or recorder would be automatically assigned to the performing students if they were set.

Table 4-2: Audio Functions Requirements (2001)

4.3.4. Audio connections between the Master EMI and the computer

A static method to connect the Master EMI to the computer was selected, so that the system software could control the audio levels.

Volume and muting control were needed for KAAN functions such as player, recorder and speak. Computer to Master EMI cabling had to allow the use of the speak function in parallel with other KAAN functions.

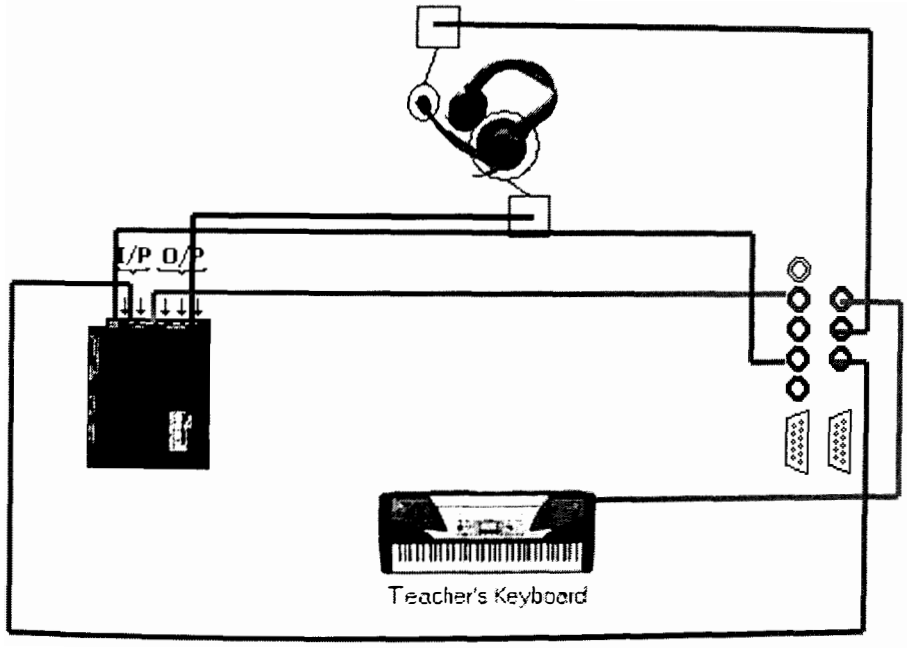


Figure 4-24: Master EMI - Computer connections (2001)

From this it was decided to use two sound cards (see Figure 4-24):

- One of lower specification for the speak function, to mix together signals from the microphone and teacher's keyboard.
- Another of higher specification for the player (output) and recorder (input).

The sound cards output levels were lower than for a keyboard, which was itself higher than mixer levels, therefore the resistors of the EMIs' pre-amplification blocks had to be customised between the Master and the slaves (see 3.3.5, page 90).

The Master EMI audio purpose was to bring signals from the music technology system to the computer for listening and recording and from the computer to the slaves for speaking and playing using the three audio buses. Again a static bus handling method was selected. Bus 2 was used solely for grouping between slaves; bus 1 was used for the listen / speak functions (on currently selected students); and bus 3 was used for the player and performing. The recording function used the listen function (on bus 1) with all control disabled while recording as the teacher needed to record exactly what was heard. Bus 2 or 3 could also be used on top of bus 1 to record in stereo. Functions conflicting with this setting were disabled after acceptance from the user.

4.3.5. Objects and Controls for KAAN audio functions

The first released version of the system software included all objects and custom controls within the main application. This was changed in the second version, with the use of external specialised Active Xs (OCX, see 2.2.4 page 38) that were 'plugged-in' the main application. The advantage of that method was to upgrade only parts of the system software without having to re-install the whole lot. These controls were:

A) Mixer.

B) Player.

C) Recorder (Audio + MIDI).

D) Midi dump / sequence handler.

A) MIXER

The *KAAN audio mixer* interfaced with the Windows mixer. The Windows mixer interfaced with sound cards drivers to control sound cards. The control was fully object oriented and Figure 4-25 shows its structure.

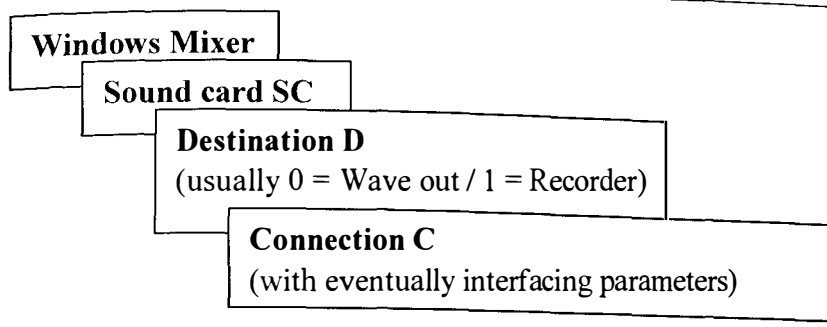


Figure 4-25: Windows Mixer Structure (2001)

A *mixerevent* control was used with the *mixer* control to capture events raised after each action on windows mixer.

B) PLAYER

A *KAAN audio player* played backing tracks from CDs, audio or MIDI files, and playlists (file listing audio files to play). The first instance of this control used two technologies to interface either with CD (low level Windows Application Program Interface (API) commands) or files (windows media player OCX). This was not efficient and only one CD drive could be interfaced with. The new windows media player 9 technologies [from Microsoft (n.d.2)] were used instead. The new technology (see Figure 4-26) managed any CD drives, large amount of file formats and broadcasts from the Internet.

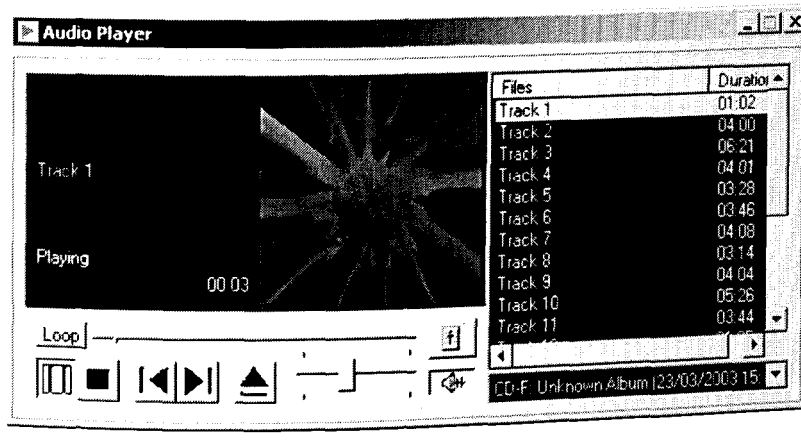


Figure 4-26: KAAN second Player (2003)

C) RECORDER (AUDIO AND MIDI)

A *KAAN audio recorder* recorded audio in a digital form for sharing. A first instance could record in wave or mp3. The main issue was that there was no direct mp3 recorder OCX available and to record in mp3, a wave file had to be created first then encoded, which took a long time. This was not efficient and the new windows media encoder 9 [from Microsoft (n.d.1)] technologies were used instead. This new technology (see Figure 4-27) managed recording both audio AND video from a video capture device in the new wm* format (wma for audio, wmv for video).

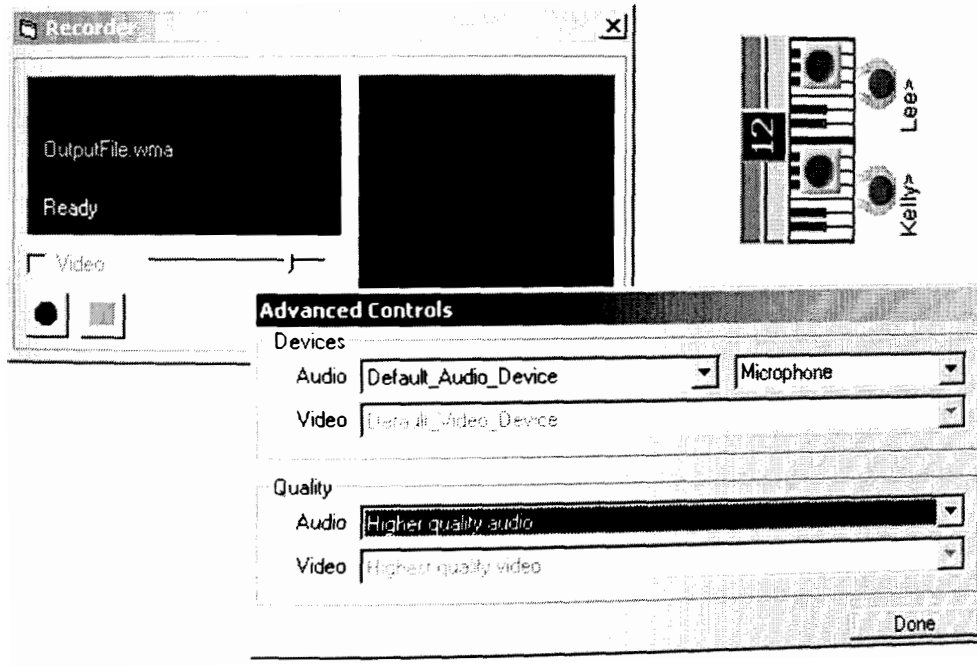


Figure 4-27: KAAN second Recorder (2003)

A separate *KAAN MIDI recorder* was developed as MIDI data did not come at any MIDI port and messages where mixed with KAAN protocol messages. The MIDI raw mode of the protocol (see 3.4.3 page 105 and 8.2.4 page 256) was used to perform MIDI recording, and therefore no setting could be performed during the recording.

D) MIDI DUMP / SEQUENCE HANDLER

Portable keyboards tended to have built in multi track sequencers, where students could store their own sequences. A sequence manager to download and upload sequences between the keyboard and the computer was required to avoid losing students' work. Sequences could be downloaded from the keyboard memory as Sysexs (see 2.2.3, page 35). As Sysexs were custom messages depending on the manufacturer and the keyboard model, each keyboard had its method to handle sequences. Some sequence downloads were initiated from the panel or from a 'sequence request' sysex. The control shown in Figure 4-28 was created to handle sequences from three different methods.

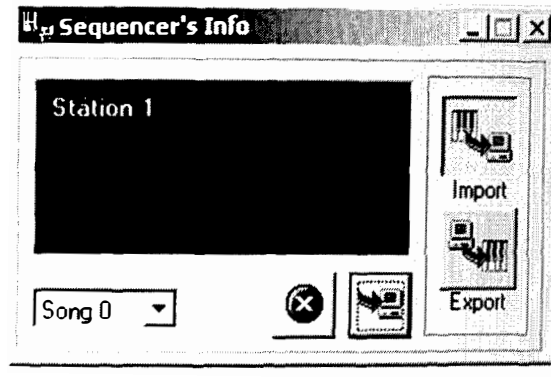


Figure 4-28: KAAN Sequence Manager (2002)

At this stage, the core engine and interfacing was ready and tested with accompanying audio features. The core engine was stored as an external OCX called *KaanEngine* to be used with other applications such as debug interfaces.

However, two of the system software specifications defined in 4.1 (page115) were not yet fully considered:

- **S6: Administration and Preparation.**
- **S8: Mobility.**

Mobility was partly achieved by using local keyboard functions (see 3.6, page 111). Another method was used to directly control the system interface using a tablet PC running Windows XP (tablet PC edition) and its inbuilt Remote Control function. This functionality allowed a client PC (in this case a tablet PC) to

remotely control a server PC (here connected to the KAAN system) using wireless networking. Teachers became more mobile and were able to control the music technology system while talking to the students (see Figure 4-29). This method was tested successfully by the author, however no teacher had tried any mobile system at the time of writing.

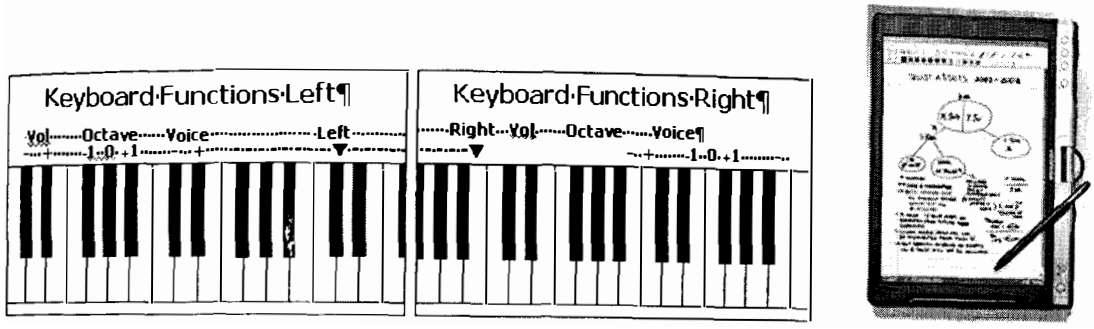


Figure 4-29: Mobility: keyboard functions and a tablet PC [Tablet PC picture from Microsoft (n.d.3)]

4.4. Administrative tools

Teachers needed to use the software system for purposes other than control, for example as an information system. Teachers needed to prepare lessons, tests or material to play to classes. Information about the students of these classes needed to be stored in a database for reporting purposes. The interaction between this information system and KAAN system through the system software was an important feature as teachers would then be able to capture students' work from KAAN system and store it in a database.

4.4.1. Information system: Electronic Student Assessment And data Management System

A database specification was determined with assistance from teachers, a Teaching Advisor and Dr Giles Tewkesbury. A relational database (MS Access) was selected because it could be interfaced from within Visual Basic and the Structured Query Language (SQL) could be used to query the database. The database was created with MS Access 2000 using a top-down analysis described in Appendix 8.3.2 (page 268).

The database was password protected as it stored student related information and to comply with the Data Protection Act [UK Government (1998)]. The GUI was also password protected and could be locked at any time and a logging password was needed to resume.

Many external files needed to be managed with the information system. A folder structure had to be considered to store these files. The first approach considered a class centred management, where students' work was stored in the student class. This method was difficult as class could be renamed and students moved classes.

A second approach was centred on an academic year folder in which all students' work was stored for a year. This allowed for easier backups and a new academic year would trigger the creation of a new academic year folder, in which the former year database would be copied and former year classes moved to the next year group. As student work and information were now stored digitally on the computer, it was required that a backup facility (incremental backup, with dated copies of the database) was implemented in case of computer failure (see Figure 4-30).

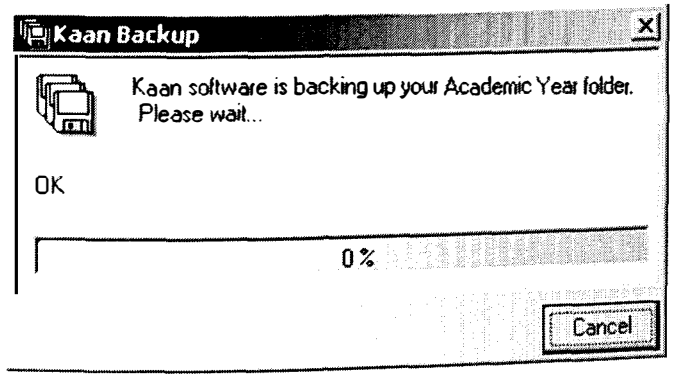


Figure 4-30: Backup window (2002)

The next section describes the administrative features implemented in the new information system. These features were appreciated by teachers testing the information system. This new information system was called ESAAMS, which stood for Electronic Student Assessment And data Management System. It used

the same interfacing method as KAAN system to access and manage students and was designed to operate on tablet PCs (see 2.2.2, page 24)

4.4.2. Administrative features

The database was created with the help of teachers and included all the information they wished to have stored. The main task was then to interface the information with the user. From the database relationship diagram in Appendix 8.3.2 (page 275), four main groups of interest could be defined:

A) Classes of students.

B) Workpieces.

C) Users / teachers.

D) Resources.

A) MANAGEMENT OF CLASSES OF STUDENTS

The teacher needed:

- *To manage:*
  - Students (new, edit, delete).
  - Classes (new, edit, delete).
  - Classes of students (add new student, import existing student, remove student, edit student's details).
- *To interface with KAAN system:*
  - Record or import / export students' work.
  - Playback students' work to them.
  - Show their details and marks.
  - Produce reports.

The management of classes and student was achieved with the use of an editor and a manager for both fields.

Students could be created and edited using the *student editor*, also called *student card* (see Figure 4-31). Two tabs enabled to edit details and workpieces

separately. For the students' details, general details were available as well as picture capturing using a webcam. Under the work tab, all workpieces could be found, edited and played back. Workpieces could be imported and recorded in the database or exported at another location under another name. Workpieces could include many files of different formats and could be sent to the application player, or the sequence manager or Windows default editor for a specific workpiece format.

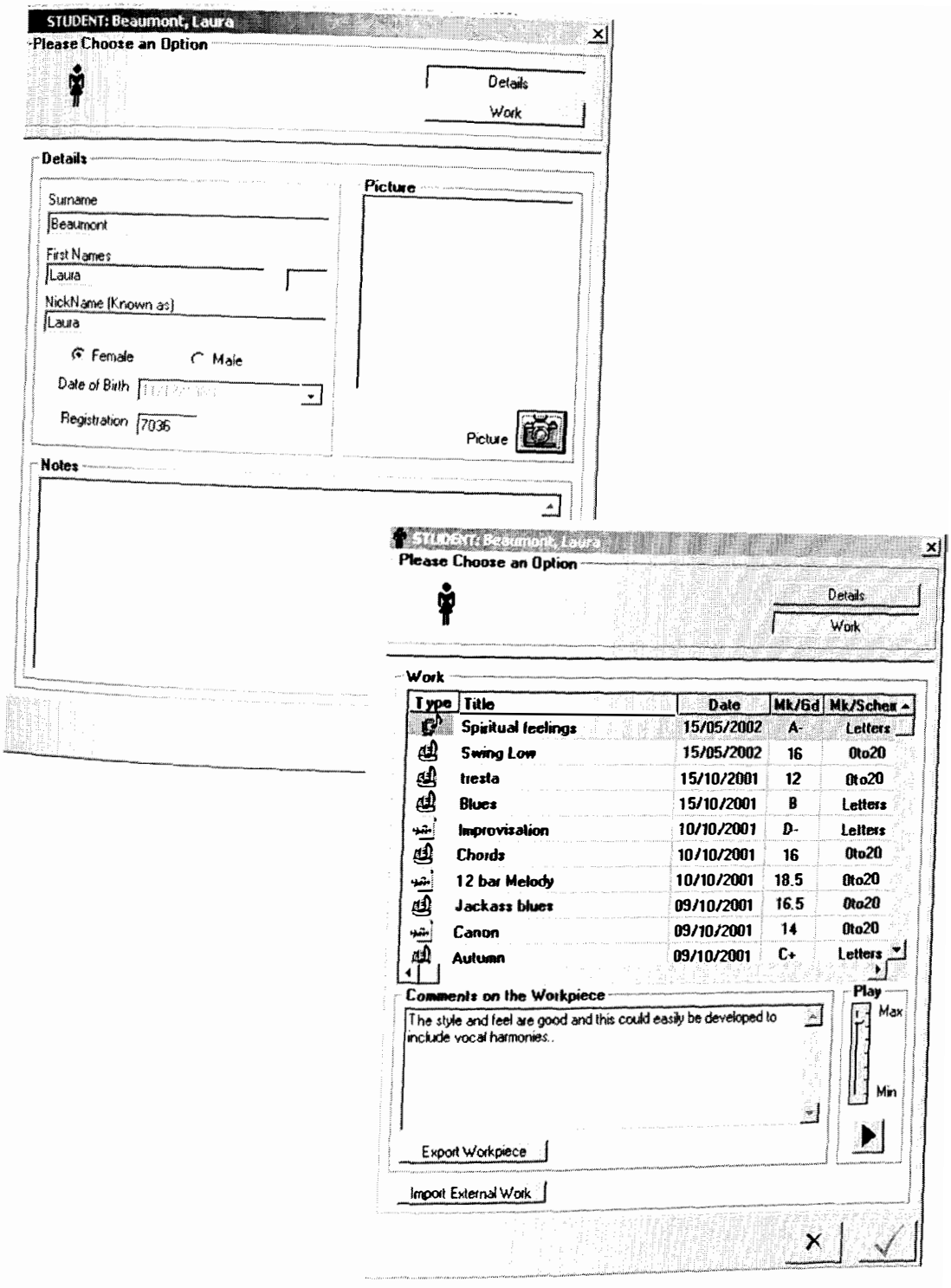


Figure 4-31: Student editor (or card) with details and work tabs (2003)

Student details could be retrieved from the database for amendment or deletion using the *student manager*. Queries could be performed on the student name or the class they were in.

Student names were displayed when a class was loaded (see Figure 4-19, page 133). This representation included their name (or nickname as chosen by the teacher) and their gender for easier recognition. Students were automatically assigned to the stations they used during the last class. Otherwise, they were left unplaced. Any student that used to be placed on a newly inactive station was automatically unplaced to be placed somewhere else. Students could be placed on any station by dragging and dropping them onto one side of a station.

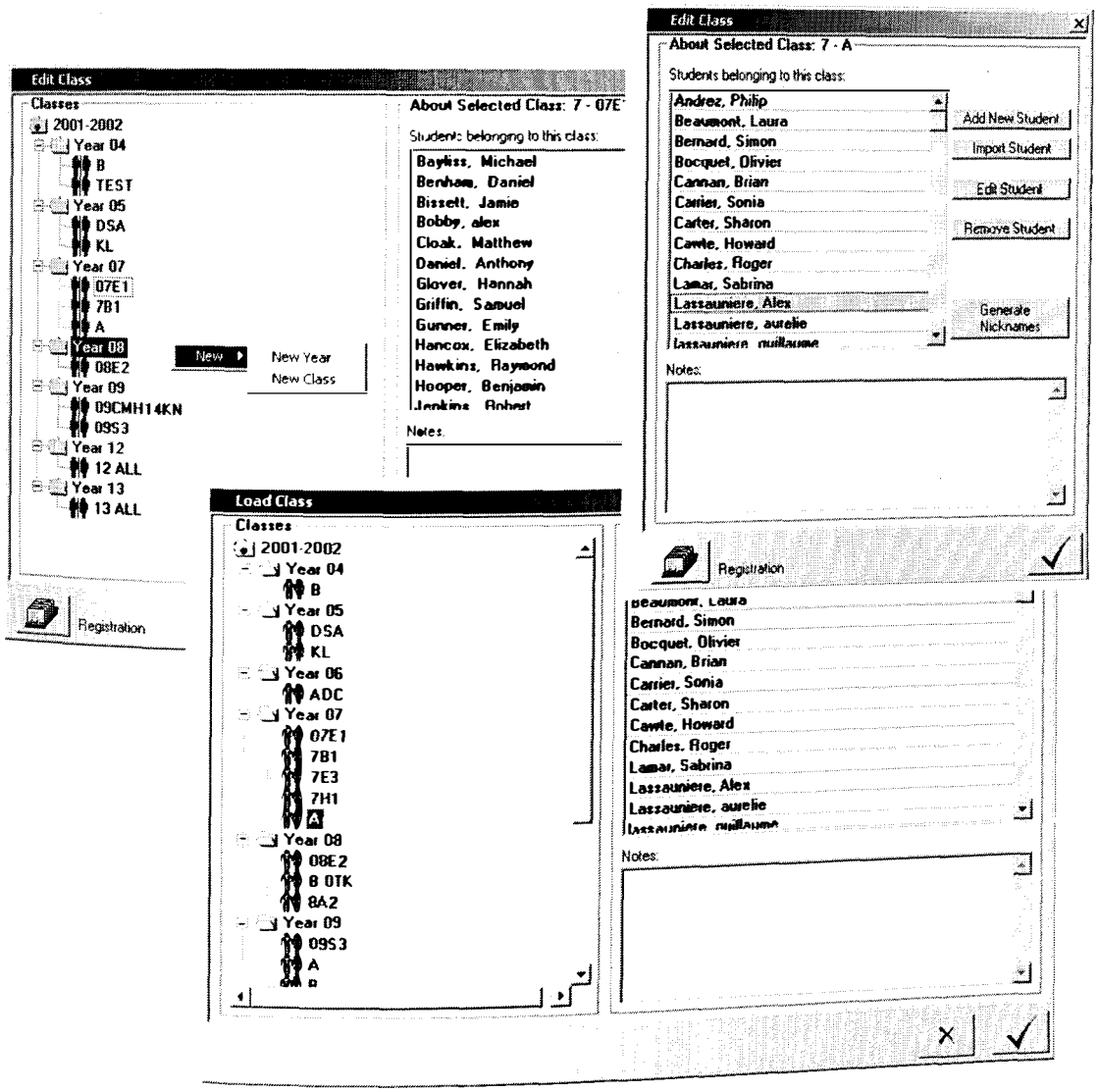


Figure 4-32: Class manager (2003)

Classes could be created, managed and loaded using the *class manager* (see Figure 4-32). Classes were defined by their name, year group and academic year. Classes also consisted of notes and students that could be added or removed. Unique student nicknames could be generated for each class to be shown on their representation.

The students could be created individually and 'imported' to classes, or be automatically created and added to classes using an import routine from the school database (see Figure 4-33). Schools generally used SIMS database [Capita Education Services (n.d.)], where classes, students' marks and other information were stored. Teachers could export the correct information as a *.csv (comma separated) file and import it into the KAAN database. The process just took 5 minutes to create around 20 classes, whereas using an individual setting could take half a day.

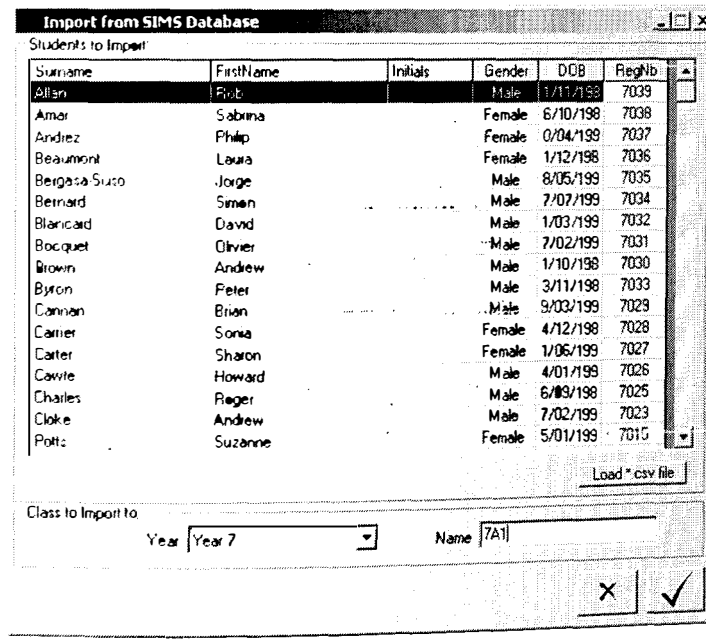


Figure 4-33: SIMS importer (2003)

The latest file structure enabled classes to be renamed, moved to another year group or deleted using the manager. Deleting a class did not delete students but just deleted their link with the class, such as removing a student from a class list.

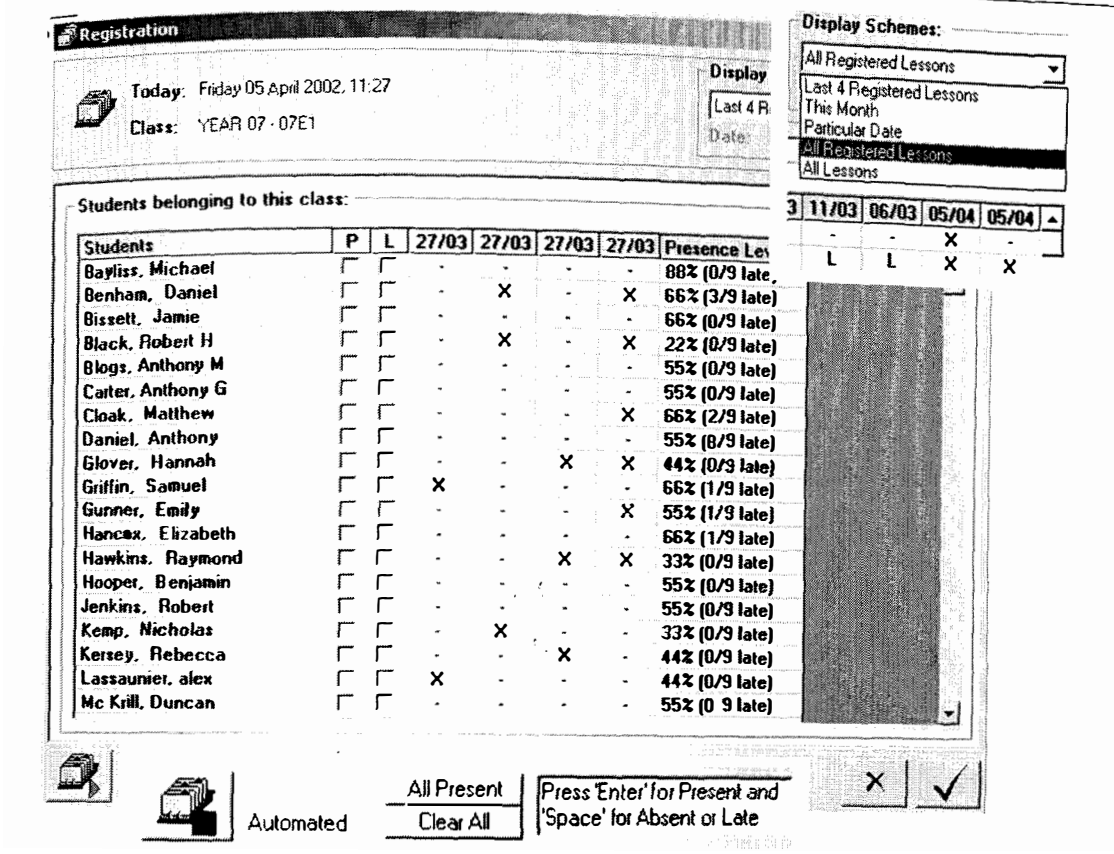


Figure 4-34: Registration manager with display schemes (top) and automated registration (bottom) (2003)

Teachers could register their students during a lesson. The process was achieved in the *registration window* shown in Figure 4-34. The process could be automated with teachers just required to press 'Enter' for present or 'space' for absent. Teachers could also display the registrations in many display schemes.

B) MANAGEMENT OF WORKPIECES

Schools and teachers had different means of marking students' work. To allow any teacher marking their students as they used to do, customised *Marking schemes* were created. *Marking schemes* consisted of captions (marks given to students, for instance A to E or 0 to 20) mapped to defined percentages (0 to 100%). A *marking scheme manager* (see Figure 4-35) enabled to edit these *Marking schemes*. An assessed workpiece could be given marks of different marking schemes, as only its percentage was stored against it.

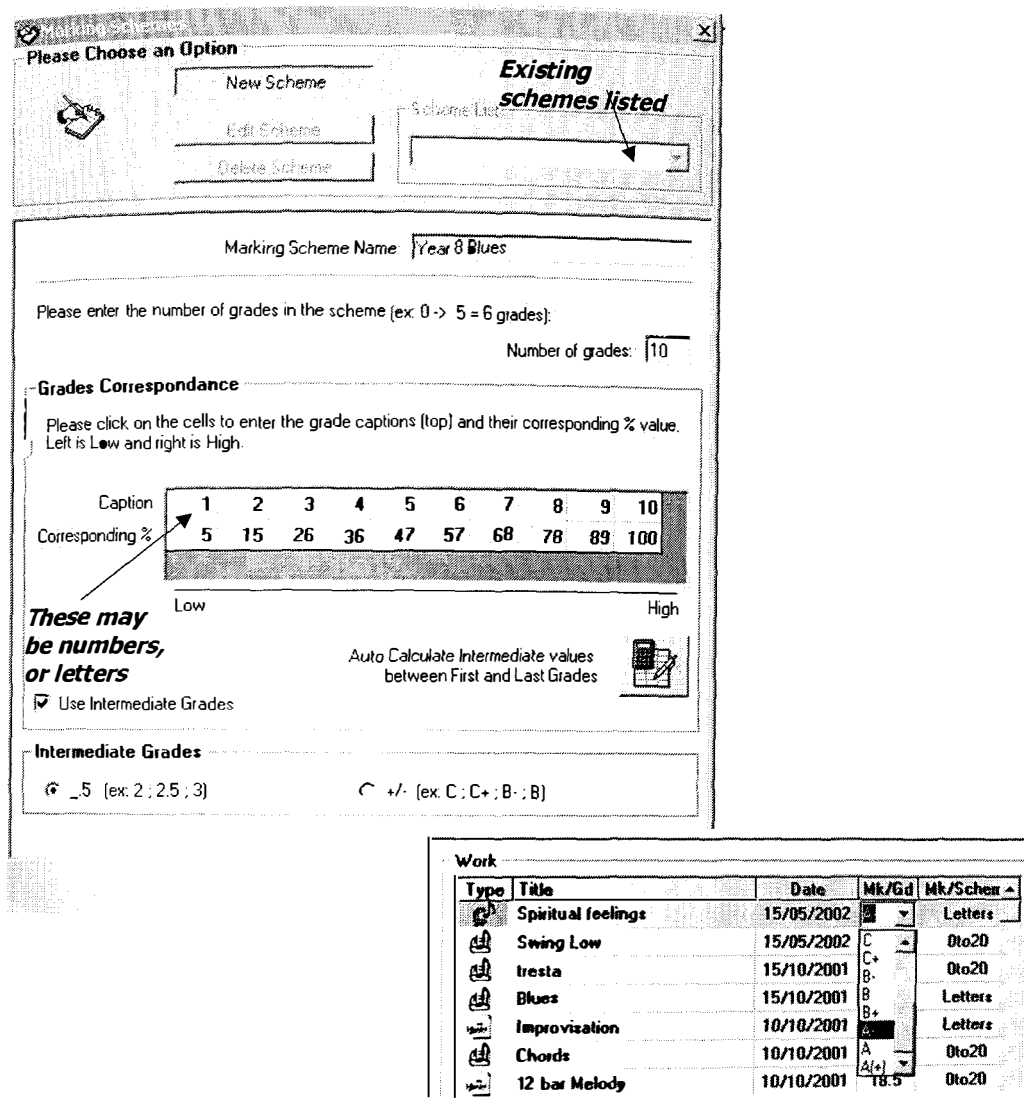


Figure 4-35: Marking schemes (manager on top) and their use (work examples on bottom) (2003)

Teachers needed to print reports from recorded information (students workpieces and registration). This was achieved using Crystal Report 9 technology [from Crystal Decisions (n.d.)] using a predefined report linking on live data through Visual Basic. Figure 4-36 shows the type of report produced. Templates could be created separately for each teacher if needed.

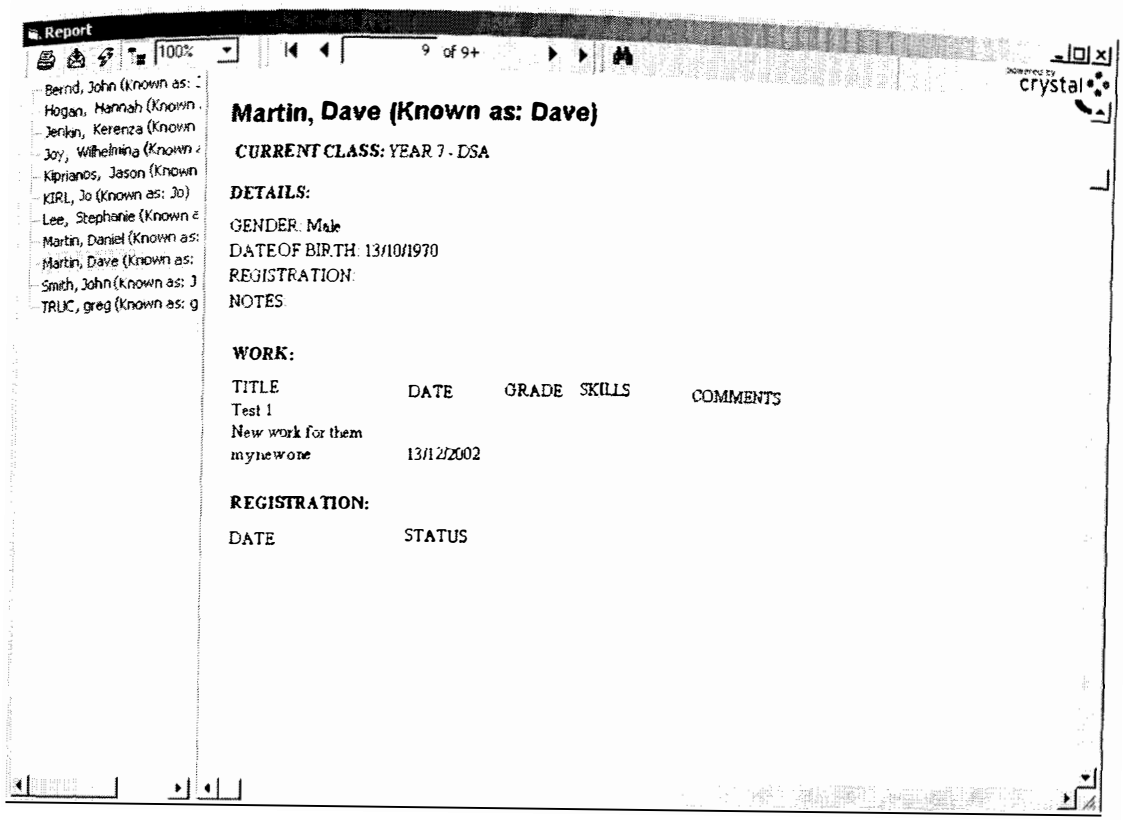


Figure 4-36: Reporting (2003)

C) MANAGEMENT OF USERS / TEACHERS

A context sensitive interface was applied to users. The information system was to be used by many teachers, each of them having a different interfacing method. As the interface could be locked and a logging password was required to unlock it, the facility was simple to implement. Three types of users were defined:

- *Administrator* with administrative rights to manage students, classes and other users.
- *KAAN Admin* like an Administrator plus debugging and KAAN layout creation rights (Note: on ESAAMS, the Administrator could setup many classroom layouts).
- *Standard users* No administrative right.

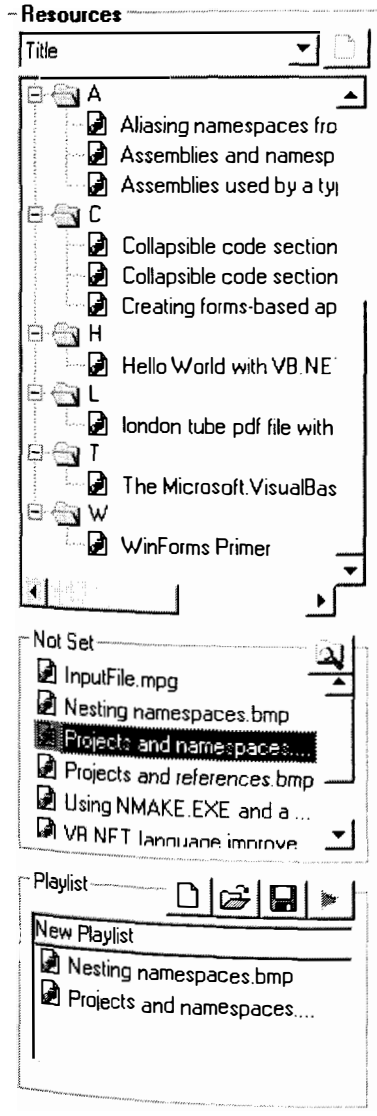
Figure 4-37: Forgot your password? (2002)

If the password was forgotten (see Figure 4-37), the teacher could call the collaborating company (contact details could be found in the user manual, the phone number was later added in the message) to give a key to enter to the information system. The key was a character-encrypted string of 16 characters using a simple Xor combination between 2 keys: the teacher password and another key. A contact at the company would pass the teacher's key to another program to generate the unlocking key to enter. After validation, the teacher could enter a new password. This method was selected to prevent students from getting the key and for its simplicity. If the company went bankrupt, the company would have to give the key generator to the teachers so that they still could use the information system.

D) MANAGEMENT OF RESOURCES FOR LESSONS

Teachers used KAAN system as an infrastructure to interface with students. They still lacked the ability to prepare lessons or tests in advance to use with students. A lesson with KAAN could be defined as the use of:

- A playlist of managed resources to play to students. These resources could be audio resources or any other kind of multimedia resources such as movie or pictures using display devices.
- Notes for the teacher on how to perform the lesson.
- Preset settings for recordings.
- A marking scheme to assess students' work.
- Other features teachers would find interesting to add.



A *web browser* control was created and implemented on the right popup window to enable teachers to interact with the Internet directly from within the information system.

Resources were managed through a manager also implemented on the right popup window for quicker access. The *Resource Manager* control displayed resources using different filters such as title, author or format (see Figure 4-38). Unmanaged resources could be dragged and dropped onto the correct filter field to update their database record. Playlists were also managed and could be sent to KAAN player.

Figure 4-38: Resource Manager (2003)

To manage resources, three steps were needed:

- A folder to gather resource files.
- A web browser to download new files.
- A resource manager to retrieve quickly resources and to put them together in playlists.

4.5. System Writing, Testing and Installation

For the first versions of the KAAN system, the design cycle did not follow any standard software engineering route as little similar product or documentation were found. That was why many prototypes were designed, tested and assessed by teachers with music technology and information systems installed at pathfinder schools. Eight cycles with major improvement took place. The last four cycles were installed at beta sites for testing and assessment:

V1.5.X (BETA RELEASE) installed at Durrington High School (Worthing, West Sussex, UK) to test:

- *Hardware:*
  - Installation and Maintenance procedures.
  - Operational parameters (fuse dimensioning, voltage drops, network current and heat dissipation).
  - Safety levels.
- *Firmware:*
  - Communication stability and problems.
- *Software:*
  - Control structures.
  - User interfaces.

V1.6.X (RELEASE 1) was installed and tested in seven schools:

- *Bideford (Devon):*
  - Two systems (One with 15 (+ 5 Slaves in a practice room) and the other with 16 slaves).
  - CTK651 keyboards + mixers.
- *Charlton (Shropshire):*
  - One system (15 Slaves).
  - CTK651 keyboards + mixers.

- *Davisons (West Sussex):*
    - One system (15 Slaves).
    - CTK671 keyboards + mixers.
- *East Brighton (East Sussex):*
    - One system (15 +1 Slaves).
    - CTK651 keyboards + mixers.

    Note: This school had difficult students who tried to dismantle and damage the system several times. This allowed the strengthening of the hardware.
- *Holly Lodge (Liverpool):*
    - One system (16 Slaves).
    - CTK651 keyboards + mixers.
- *Holy Trinity (West Sussex):*
    - One system (15 Slaves).
    - CTK671 keyboards + mixers.
- *Poole (Dorset):*
    - One system (14 Slaves).
    - CTK651 keyboards + mixers.

V1.7.X (RELEASE 2) was first fully tested in house, and then installed as first installation and tested at *Perins (15 + 1 + 1 Slaves, Hampshire)*, *Winton (16 Slaves, Hampshire)* and *Bound Stone (15 Slaves, West Sussex)*. All former schools were upgraded to Release 2 without any issues being identified.

V2.0.X (RELEASE 3 – ESAAMS – KAAN) was written as a unique music technology and information system working in two modes (ESAAMS mode as default, a KAAN Administrator only could change to KAAN mode). Both systems were first fully tested in house, and all former schools were upgraded to Release 3 without any issues being identified. The systems were designed to run on Win 2000, Win XP and Win XP tablet PC Operating Systems.

4.5.1. System Analysis (Release 2 to 3)

A full system analysis was created after release 1 to produce release 2 as a full re-write. Release 3 required another system analysis to include ESAAMS features and dispatching of functionality to OCXs and Dlls (see 2.2.4, page 38) to improve independencies between objects and future software system migration to Visual Basic .Net.

A Unified Modelling Language (UML) system analysis was produced for the system software along with a conceptual model. UML was chosen as it specially dealt with programming objects. System functions could be defined to determine the essential functionality of the software system. Concepts could then be determined as software system entities interacting with each other to perform the system functions. A conceptual model described the interactions between concepts. System Functions were too general and could be described in Use Cases, which in turn could be split into more specific System Sequence Diagram functions. System analysis for KAAN – ESAAMS (release 3) is documented in Appendix 8.3.2, page 268.

4.5.2. Testing

A special testing application (see Figure 4-39) was created to allow general and focused testing. It used system UCs as functions to test and the testing could be focused on certain System Functions. The testing mode and Use Cases to test could be set independently for any user. Bugs were reported using the window shown on Figure 4-40.

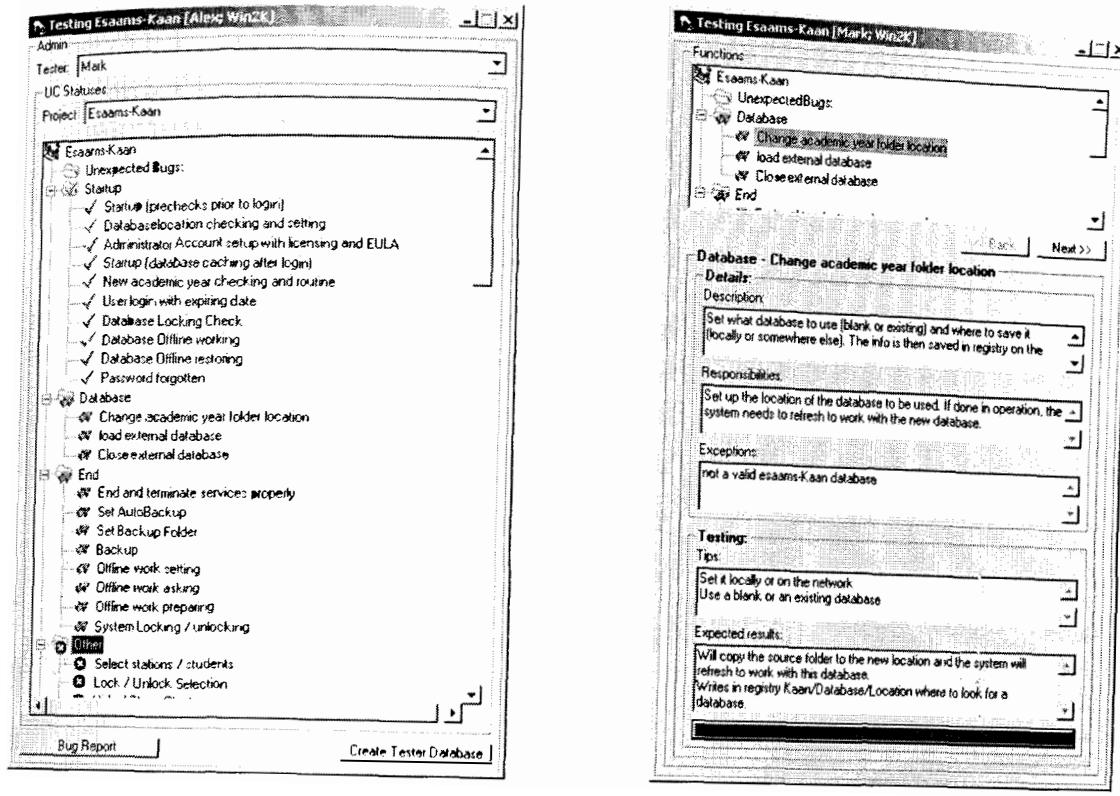


Figure 4-39: Testing application (Administrator setting for Mark (left) / Mark interface (right)] (2003)

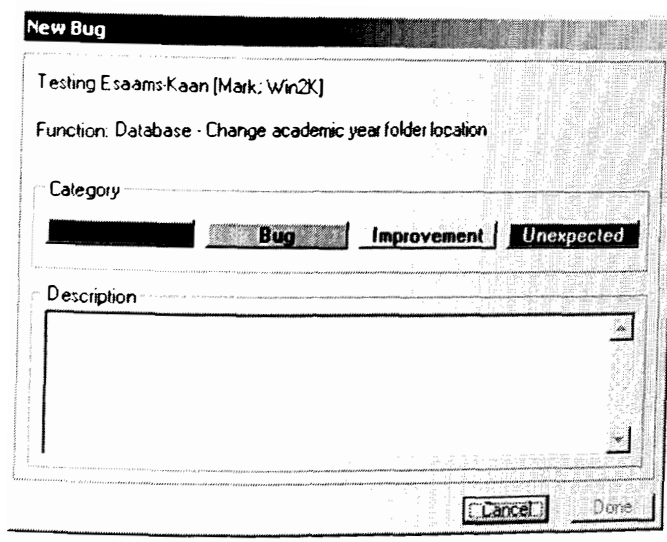


Figure 4-40: Bug reporting window (2003)

4.5.3. Installation

MS Visual Studio Package & Deployment Wizard was used to create the setup application for the software system. It collected any required files (exe, dll, ocx and others) to operate the software system.

Software system patches were sent to teachers with a special setup application. Future work planned the inclusion of a web upgrade feature in all custom components.

4.6. Chapter Discussion

This chapter explained how a new software system to interface with the new KAAN system was designed, tested, and installed. This induced the creation of another new information system called ESAAMS for administrative purposes.

The specifications set in 4.1 (page 115) were reviewed as follows:

S1: Control the new KAAN system.

The following were implemented:

- KaanEngine OCX.
- Control Structures.
- Setting Interfaces.
- KAAN audio functions and grouping.
- Debug tools.
- Mobile EMIs.

S2: Easy to learn and to use.

The following were implemented:

- HCI methods.
- No help section.
- User manual and training.

S3: Attractive and Meaningful.

The following were implemented:

- HCI methods.
- Classroom layout.
- Setting Interfaces.

S4: Adaptable and Upgradeable.

The following were implemented:

- OCXs, Dlls plug-ins.
- Creation of ESAAMS.
- Patch setup.

S5: Fast Interaction.

The following were implemented:

- HCI methods.
- Classroom layout.
- Setting Interfaces.

S6: Administration and Preparation.

The following were implemented:

- KAAN database.
- Creation of ESAAMS.

S7: Accurate System Control and Status Display.

The following were implemented:

- Classroom layout.
- Setting Interfaces.
- Control Structures.
- Debug tools.
- Mobile EMIs.

S8: Mobility.

The following were implemented:

- Keyboard local functions.
- Tablet PC.

The new audio infrastructure to help music teachers was ready and at the time of writing installed in eleven schools and used for up to two years. Feedback from teachers was needed to check for unforeseen issues in using KAAN systems and methods to solve them.

CHAPTER 5. TEACHING IMPROVEMENT WITH THE NEW KEYBOARD AND AUDIO NETWORK (KAAN) SYSTEM

An aim of this research was to help music teachers to deliver their lessons more efficiently. A first proposal, considered by music advisors [Odam (1997), Rogers (1997), Salaman (1997), Odam and Walters (2003)], was that teachers lacked appropriate tools to:

- Use keyboards and other instruments efficiently.
- Control instruments to keep students focused on the lesson.
- Manage students' work better by:
    - Saving work on shareable media.
    - Accessing efficient administration tools.

To address these issues, a new Keyboard And Audio Network (KAAN) system was created and tested with teachers. New base software was created to interface with the new music technology systems. The user interface was designed using HCI techniques (see 2.2.4) so that teachers could quickly learn to use it efficiently. Many iterations were required to create a usable interface. No help section was provided, as the aim was to create a user-friendly and intuitive interface. Teachers received a user manual. Training was performed by a Teacher Advisor working with the collaborating company and trained by the author.

Administrative tools were written and implemented in the base software to create a suitable new solution for teachers. These tools were later marketed independently under the brand name ESAAMS.

Feedback was collected from teachers using the new music technology and information system to gain an overview of the benefits and any remaining issues or drawbacks impacting their teaching.

This chapter presents and discusses feedback results from teachers using the new music technology and information systems. From these results, proposals to resolve remaining issues were determined. Each proposal was investigated in mini projects and compared to the other. Because of time issues, only the most useful was fully researched. The selected sub-project is discussed further in Chapter 6.

5.1. Impact on music teachers

The results from usability-testing and feedback from teachers were collected to check the KAAN system had created a new, ready-to-use infrastructure to help music teachers use instruments and material and to manage students' work more efficiently.

To collect the information, two surveys were sent to teachers. The survey method was chosen initially rather than a direct interview due to the location of the schools where the music technology systems were installed. At the time of writing, KAAN systems have been installed in eleven schools countrywide and have been used for up to two years (Figure 5-1). They were proving to be reliable and simple to maintain.

Figure 5-1: KAAN system being used on site (Durrington High School, West Sussex, UK) (2001)

Each survey was designed following the methods explained in Ackroyd and Hugues (1981), Hart (1986) or Diaper (1989), and with assistance with a Teacher Advisor working with the collaborating company.

A first survey was created to get information about current software system functions and proposed functions to implement to improve the base software. The survey contained three parts:

- Frequency of use of existing functions, to check usage habits and to see if the training could be improved.
- Teachers' proposals for future functions to be implemented.
- Prioritising proposed future functions.

After considering the results from the first survey, a second survey was created to gather more personal and open information on the teachers' use of the new music technology and information systems, the benefits, drawbacks and issues encountered. The results were then compared with observations from people working with teachers using the systems, including the author, installers and maintainers, trainers and support.

Observations during real lessons and interviews were conducted with almost all teachers using the KAAN system as well as a Teacher Advisor. The aim of these interviews was to understand the use of keyboards for teaching.

The survey templates, results, examples of interview summaries and answers are in Appendices 8.4 (from page 283).

5.1.1. Feedback results

Results are considered from:

A) First Survey.

B) Second Survey.

C) Interviews.

A) FIRST SURVEY

A survey (see template and results in Appendix 8.4, page 283) was sent to the eight teachers using the first and second release of the base software. Four

surveys were received out of eight (50%), which suggested that teachers were willing to give their opinions (the usual answer rate being 20%).

The first part of the survey concerned the frequency of use of existing functions, to check usage habits and to consider whether training could be improved. Teachers were presented with a set of Use Cases [Yeates *et al.* (1994) or see page 39) from the software documentation. They could provide feedback using any of these answers:

- 0: What is it.
- 1: Never.
- 2: Never but can use it in Future.
- 3: sometimes.
- 4: Often.
- 5: Always.

The results showed that:

- Seven features with answer 0, indicated that some features were either not understood or not discussed during training on the new software systems.
- Fifty-one features with 3 or 4 out of sixty-four (for both releases) showed that more than three quarters of the proposed functions were frequently used.

63. Right popup settings window disappearing options
1. Forgot your password
5. Restore communication in offline mode
34. Record work onto tape
7. Alarm
21. Instrument favourites management
52. Set student's picture (picture capture with webcam) (Release 2 Users ONLY)
64. Startup settings management
2. Change password
4. Use system with no control (offline)
3. Lock interface
56. Use SIMS database records import facility
59. Delete classes

Table 5-1: First Survey, Part 1, less frequently used functions (2002)

Table 5-1 shows the functions with answers below 3 in order of frequency of response (the highlighted ones indicate one or more answer 0):

The results indicated that features 63, 34, 7 and 21 were not used and the further development of such features would need to be discussed. One-off functions (52, 64, 56, 59) were rarely used, which was expected. Features dealing with communication problems (5, 4), could have recorded effective software system operation, but in fact showed that teachers were not aware that a hardware crash did not involve a software crash. A more worrying matter was the security offered by features 1, 2 and 3. Teachers appeared not to understand the impact of having confidential data in the information system.

The second part of the survey dealt with proposals for future functions. This was important as this could show if teachers understood the potential of the new music technology and information systems and their flexibility, as well as their expectations. The results showed:

- Recording (in mp3) improvement:
    - Allowed setting and notes typing while encoding.
    - Wave editor + track appending.
- Database interfacing improvements:
    - Student import.
    - Reports.
    - Deleting work.
    - Registration available on the network.
- Setting of non-GM instruments.

These improvements (except the wave editor, which was considered as a whole application in itself) were considered during the development of the new information system and led to the creation of a new system called ESAAMS (see 4.4, from page 144).

The third and last part of the survey dealt with prioritising future functions. Teachers fed back the importance of functions that could be created in the future.

This part was deliberately considered last so as not to bias answers in the second part of the survey. Top results are shown in Table 5-2:

	Priority
Lesson creation (resources + notes + settings + marking schemes + tests)	8**
Handle any kind of Workpiece / resources and implement appropriate editing tools (as well as visual resources)	6**
Advanced SIMS import / export	5*
Audio tools implement. (Audio explorer, encoder/ripper, drum editor, wave/mp3/midi editor, CD writer, video/still capture,...)	5*
Dual Network System (a PC on each student's desk)	5*
Email facility (in + out)	5*
On line help/tips/updates/manual	5*
Remote control with PDA or webpad	5*

Table 5-2: First Survey, Part 3, prioritised proposed functions (2002)

The number of * showed the number of top grades. These results showed that tools to prepare and improve lessons were most attractive, followed by features that could help teachers to improve the way they used the new music technology and information systems.

Security policies only scored 2, which confirmed the earlier observation that teachers may not realise the importance of this issue.

B) SECOND SURVEY

Teachers were observed using the music technology and information systems on site. Benefits, drawbacks and issues were noted (see template and results in Appendix 8.4, page 286):

- *Benefits:*
  - Students no longer had to share a keyboard and could work independently.
  - Students could play with another person or as part of a larger group without having to move around the room.
  - Music files could be shared.
  - Teachers could monitor students and record their performances.
  - The new systems appeared to be popular with both teachers and students.

- The new systems helped to improve students' compositions.
- Instruments were ready-to-use, less liable to be damaged, and used more effectively.
- Teachers could use other CAI (see 2.1.2, page 10) or audio software with the whole class.
- Interaction with students was enhanced and more targeted to individuals or groups, rather than to the whole class.

- *Drawbacks and Remaining Issues:*
  - Teachers still needed to organise themselves to deliver their lesson efficiently.
  - The interaction with students using the music technology system was good. However, time was sometimes wasted using the interface. Students then lost focus on the lesson and teachers had to regain control to continue their lesson (although this was an improvement compared with teaching without the music technology system).
  - Teachers used and interacted with the music technology system in many different ways, some of which were not anticipated (for example recording students' work in the sequencer for 'after lesson' MIDI recording).
  - Keyboards were used as primary instruments with KAAN system. A multitude of keyboard playing skills was taught, especially with new students (Year 7).

C) INTERVIEWS

Teachers confirmed that (see template and results in Appendix 8.4, page 287):

- Keyboards were useful instruments to teach music.
- Many teaching activities could be performed on a keyboard.
- New students had variable prior keyboard skills [even if this was to be taught in Key stages 1 and 2, Heinemann Educational (n.d.), British Department for Education and Employment (n.d.), Mills and Murray (2000)].

- Keyboard teaching skills depended upon teachers and lesson planning and conduct.

5.1.2. Discussion

The music technology system achieved its target to create a new infrastructure to use music materials for lessons more effectively. Teachers were happy with the music technology system and some were willing to work with the author. However, teachers did not always consider the security issues with the data contained in their information system. Furthermore, a few problems still remained before implementing new and more advanced features. These issues concerned:

- Lesson teaching (preparation, organisation and delivery).
- System usage (interaction, functions too long or tedious to perform).
- Communication issues.
- Use of keyboards for teaching.
- Different students' skills.

The first issue could involve better self-organisation methods for the teacher, which will not be dealt with in this research, as teachers could be trained or documented on the subject by their Local Education Authority. Other remaining issues required some training for the teacher on the potential of the new music technology systems and keyboards. Besides, some tool to give minimal keyboard skills required to follow the lesson to all students could be implemented.

Therefore, a software artificial assistant was needed to help music teachers during a lesson to perform some tasks on the new music technology and information systems. Such an assistant would require some intelligence in order to perform on its own in a desired way and evolve with teachers' requirements.

The assistance could be provided to:

- Perform and anticipate teacher's tasks, to decrease teacher's workload and disturbance **(Sub-project 1)**.
- Help teaching basic keyboard and music skills to new students. This could be performed individually so that teachers could start their lessons to students with the same basic skills **(Sub-project 2)**.

5.2. Assistance from intelligent agents

5.2.1. Habit capturing

As seen in 2.3.3 (page 49), intelligent agents aimed at simplifying the work of users could be used to assist music teachers. By anticipating teachers' actions, some tasks could be performed quicker. Other tasks could also be performed in the background while teachers perform other tasks more related to their teaching. For this user modelling is required to monitor and discover habit patterns.

The user should have access to the model and be able to modify it. The user would need to control what has been acquired and decrease the level of paranoia created by this 'Big Brother' approach. The user may be able to help the agent by confirming ambiguous habits and add other known habits. Therefore, there should be an appropriate easy-to-use language allowing the user to modify the knowledge base but protect it from being corrupted [Terveen and Murray (1996)].

Different kinds of habits were categorised as:

- Absolute habits where a pattern was recognised to appear always or often at the same period of time, in absolute time or after a major event (program start up, load new class, etc.).
- Relative habits where a pattern always appeared or often appeared after another pattern.

Two problems now arose:

- How to learn habits?
- How to recognise them?

Habits needed to be learnt and recognised 'on-line' in order to anticipate events. Habits may change and therefore the learning base needed to be updated frequently. The user may know some habits and may want to add them manually to the knowledge base. Habits may differ in type:

- One task may be processed after one event.
- One task may be processed after a pattern of events.
- A pattern of tasks may be processed after one event.
- A pattern of tasks may be processed after a pattern of events.

Examples:

- One teacher always loaded Class1 every Monday at time1 (Class1 only changed once a year).
- One teacher had his own way of performing his teaching in a defined sequence of actions.

Habits are recognised after certain events have occurred. As these events need to be processed in order to retrieve known patterns or to capture habits, they have to give enough information about themselves. A possible solution is to stamp every event describing its type and when it occurred and recording the context. This stamp may be processed and compared to stored stamps and patterns. A Sequential Query Language Database (see 2.2.4, page 43) was considered because of its ability to process SQL statements. If a pattern already existed, a statistical 'trust' parameter could be updated. This parameter could be used to determine the confidence that a pattern had been recognised as a habit [Ebihara (1992), Lee and Lim (1994), Leemis (1997)]. Different options could be proposed to the user to process. The method to complete the tasks of a recovered habit may be launched as a macro (or collection) of tasks. The intelligent system could also check if the proposed options were correct. A stamp contained information on the last event. However, the former events would also be essential and must be used since a pattern is not likely to be only two events in length. A history of events could be stored (such as an 'undo' command). Research determined the best stamping format and collecting method to be used. Figure 5-2 shows a method used to capture habits.

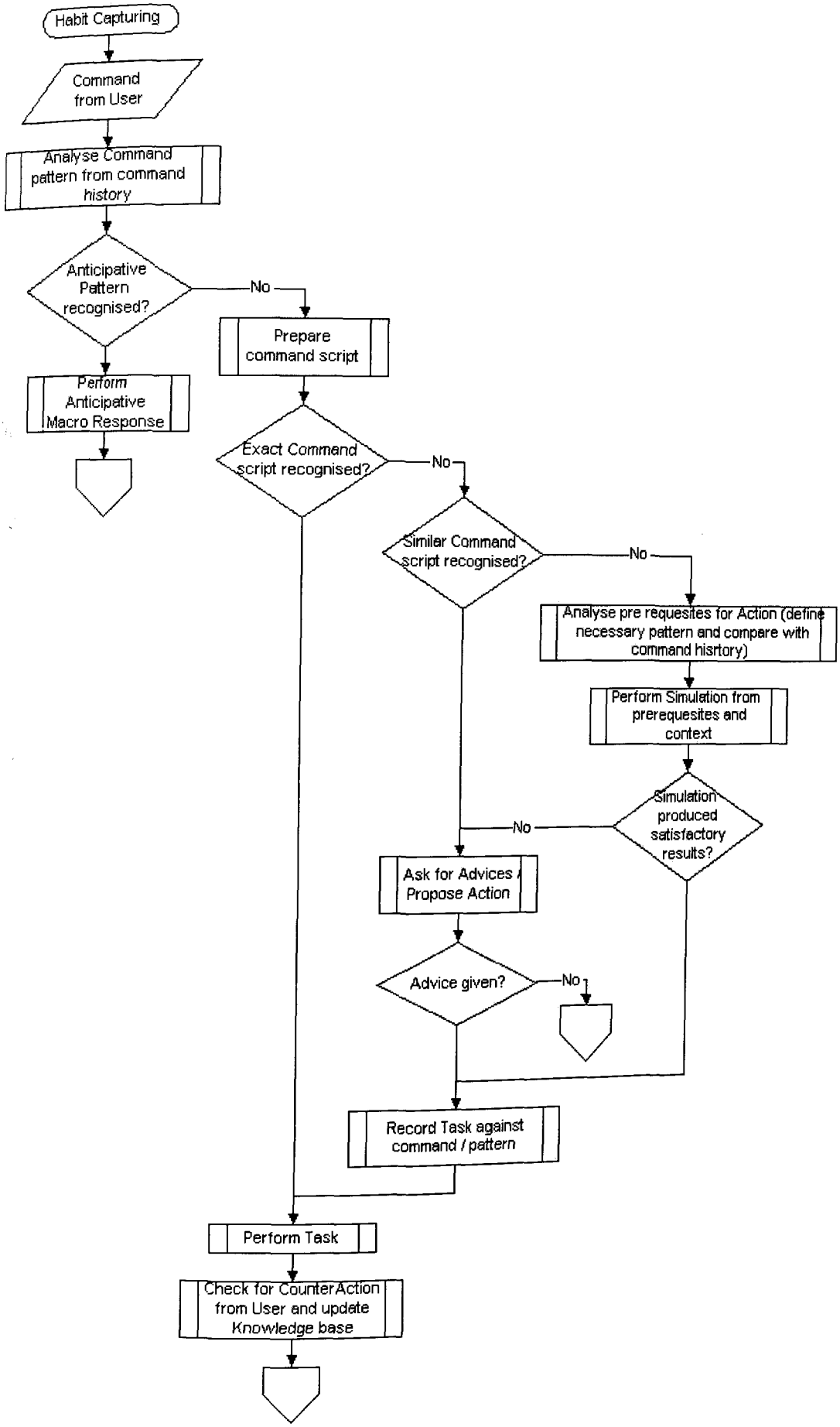


Figure 5-2: Habit Capturing (2002)

The fields of information a stamp contained:

- *Action details:*
  - Context
  - Type
- *Parameters:*
  - Parameter 1
  - Parameter 2
- *Timing details:*
  - Index 1 (/Application start up)
  - Index 2 (/other reference event)
  - DeltaTime 1 (/Application start up)
  - DeltaTime 2 (/other reference event)
  - Month + day (literal)
  - Month
  - Day (Number)
  - Day (literal)
- *Time* (Hour:minute:second)
- *Relatively to last event*
  - Deltatime
  - Last event details:
    - Context
    - Type
  - Parameters:
    - Param1
    - Param2

5.2.2. Mini project: Yahtzee

To better understand the role of habits, a program was written to capture habits. A Yahtzee game was selected because of the way each individual player scored points. The scoring depended upon individual habitual tactics.

Yahtzee is a game with five dice, which can be played alone or in a group. The game consists of thirteen rounds. In each round, a player rolls the dice up to three times and then scores the roll in one of thirteen categories. The player must score once in each category, which means that towards the end of the game the player may have to settle for scoring zero in some categories. The score is determined by a different rule for each category. The object of the game is to maximize the total score and the game ends once all thirteen categories have been scored.

A score can be between 0 (the chosen category scores nothing from the roll) and 50 (Yahtzee with five similar dice faces). Sometimes it is better to score 0, a low score or not the top possible score, in order to leave better chances later on.

Each player would score differently according to the context. The scoring tactics usually follow the habits of the player and little time is involved with this habit, just context.

The writing of the program followed:

- Build the game.
- Show available combinations with non-null scores with different colours.
- Offer the best score within the scoring category with another colour.

As explained above, a player can decide not to score the maximum following some tactical rules and priorities. It was these rules and priorities that needed to be captured. A general scoring rule appeared to be to score the maximum except if it interfered with other personal goals (choices and habits). For instance, was the selection 66655 better to score with a 'Full House' worth 25 points, a '3 of a kind' worth 28 or 'Sixes' worth only 18 points but that could help gaining a bonus of 30 later?

The intelligent system needed to determine the player's goals hierarchy. At the beginning, this was set as a stereotype and then updated. Then the intelligent system captured the player choosing a non-maximum score and tried to determine why against the context, as well as scoring before the last roll. The context was determined by the category remaining to score (which also indicated the round), the dice faces and the roll number.

5.2.3. Discussion on task assistance

It was concluded that task assistance was feasible and could be performed in two forms:

- Sequential helper for tedious tasks.
- Task anticipator for habitual tasks.

The sequential helper was being planned and will be put in place in the software system in the future with predefined sequences, and could be extended to macro capturing as implemented in MS Word or Adobe Photoshop. A task anticipator needed more research and results from the Yahtzee project showed that:

- Different kinds of habits existed.
- Some habits were time related.
- Stereotypes could be used to provide quick results.
- Could take a long time to capture and be able to propose or act for the teacher.
- Interaction with the teacher was required to avoid a counter effect.

With KAAN and / or ESAAMS, habits could be of several types:

- Schedule capturing (what class at what date / time).
- Lesson process (teacher and year group dependent).
- Interfacing options (like MS Agents).

If this hypothesis was to be fully researched, further work included the determination of:

- Teacher stereotypes.
- Stamp handling for habit capturing.
- Interaction with the teacher.

As concluded at the end of this chapter, this hypothesis was not continued and none of the above was researched.

5.3. Automated teaching and testing

As seen in 2.3.3 (page 53), Intelligent Tutoring Systems (ITS) existed in various fields to teach in a customised way by modelling a student. Work from Gerlič (1998) showed interesting general templates to develop new ITS, for any field (like Figure 2-14: Third Application Type, page 58).

5.3.1. Intelligent Tutoring System to teach keyboard skills

Various teaching techniques could be useful for teaching music:

- Use student interaction (learning by doing) and get semantic and working memory.
- Assess student level and check knowledge is fully integrated to allow next step teaching.
- Use of concepts, as music is a language and can be defined by rules.
- Diagnose errors and train to solve errors against misconceptions.
- Use different teaching methods for different students and different teachers.

Furthermore, a student model could be useful for administrative purposes to report current students' knowledge and problems. An ITS could be implemented with the new KAAN system, using teacher's computer and keyboards. The computer would run both the ITS and KAAN's interfacing engines. The interaction between the ITS and the students would be:

- *Student ➔ ITS / Computer:*
  - Key presses captured by the EMI and sent to the computer (see 3.4.3, page 105).
  - Other: time (to respond) could ponder the responses.
- *ITS / Computer ➔ Students:*
  - Speech: pre-recorded sequences OR text-to-speech engine, the audio will be sent to the appropriate EMI through one of the audio buses.
  - Other: New keyboard models from Yamaha (EZ, EX series) and Casio (LK series) had embedded key lighting facilities used for their KAI teaching. Higher specs models of the series included remote key lighting through reception of a MIDI message. This could be used as a visual aid.

A **first research plan** was to observe actual music teaching and to computerise this teaching. The idea was to capture the teachers' way of teaching as this varied from one teacher to another. A hybrid system could be used to watch test marking from the teacher. It could ask questions about the marking methods in a non-interruptive way to build a model of the teacher's teaching strategy. The research plan followed the phases explained below:

- A teacher and a student sit in two rooms. The teacher tests the student remotely, using KAAN system. The teacher asks students questions and marks responses, from captured key presses and time to respond (Figure 5-3).
- The author or a monitoring program captures the interaction (or information flow) between the teacher and the student. The author then models the teaching process from the gathered information.
- An Intelligent Assistant using the teaching model is added in the interaction between the teacher and the student (Figure 5-4). The assistant proposes actions to the teacher for a given task. The teacher can then accomplish proposed tasks by clicking on a button. If the action is unsatisfactory, the teacher can bypass the assistant and give his own directives. The teaching process is also monitored in order to refine and validate the AI model. The new intelligent system assesses the information created above and automatically makes decisions (correct-incorrect).
- Extend decision making to 'very good', 'good', 'satisfactory' and 'unsatisfactory'.
- Model user(s) to build a model of their achievement and appropriate levels for re-examination / examination.
- The teacher is replaced by the assistant, which now carried out the task. The student and the teacher are questioned on the teaching process to validate the AI model.

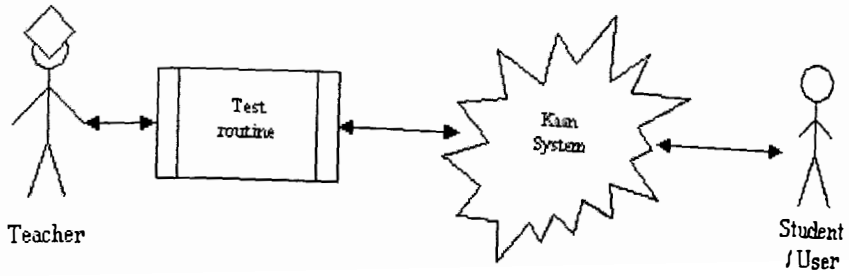


Figure 5-3: First ITS/KAAN system (2002)

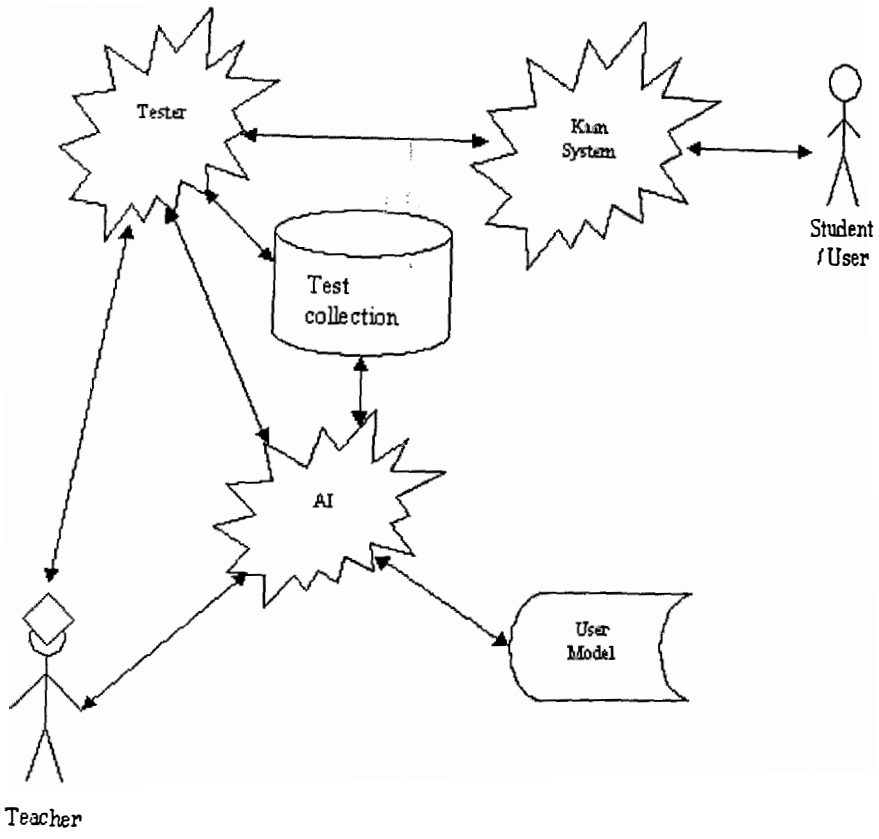


Figure 5-4: Second ITS/KAAN system (2002)

This research plan was not followed after observing teachers in action. It was found that teachers delivered small chunks of keyboard knowledge only when required for their lesson, without checking students' understanding of the new knowledge. Therefore, the hybrid system was not created.

A **second research plan** tended towards the observation of existing CAI, KAI and ICAI techniques. The idea was to keep only methods and structures that would best fit the creation of an ITS to deliver customised keyboard and music skills.

First, to determine the useful keyboard skills and tests that could be completed with students using a KAAN system, KAI and CAI systems were reviewed. The findings were then confirmed and amended by a Teacher Advisor. These skills were to be able to:

- Notes and chords:
  - Recognise or play a note or a chord.
  - Recognise or play intervals between notes.
  - Play a series of notes or chords (with a rhythm or not).
- Play (tap) a rhythm.
- Answer yes/no/other (quiz).

Second, ICAI techniques were reviewed and tested. As seen in 2.3.3 (from page 53), teaching methods were described in Self (1988). The methods were to:

- *Compare student knowledge against 'perfect' (expert) knowledge:*
  - Tasks were executed through a set of rules. A student model of his knowledge was compared and critiqued against Expert knowledge to show misconceptions and errors.
  - Student's level was assessed by checking the knowledge transfer between the tutor (expert) and the student. The intelligent system then checked the knowledge was fully integrated as well as prerequisites to teach the next lesson.
  - A philosophy was 'learning to learn'. The tutor - expert model was compared with the student model in order to reduce differences.
- *Capture bugs (i.e. misconceptions and errors) to solve them:*
  - Student actions were predicted to anticipate bugs.
  - Misconceptions were captured by the intelligent system that would then try to get the number of misconceptions down to 0. The intelligent system captured student's error patterns and attempted to solve bugs.
  - Errors were diagnosed and the intelligent system trained to solve these errors against misconceptions.

- *Use actions history to build the model:*
    - A learning history (student model building history) was used to get long-term memory that was compared with episodic memory. A generalisation was made to build analogies between the two.
    - Experience, (for example actions history), frequency of use of functions and help were captured to detect errors.
    - A philosophy was 'learning by doing'. Student interaction with the intelligent system was used to capture his semantic memory (previous knowledge) as well as his working memory (immediate understanding).
- *Have an intelligent student modelling agent learning at the same time as the student:*
    - A student model built concepts in the same time as the real student. The student and model then tried to collaborate to build a common knowledge.

Some of these techniques were useful. Student interaction with the intelligent system was used to obtain semantic and working memory. Student level assessment and knowledge checking were fully integrated to teach the next lesson. Music was a language, and was therefore defined with concepts and rules. Errors were diagnosed and the intelligent system could train to solve errors against misconceptions. Different teaching methods were used for different students and different teachers.

The second research plan was selected as it built on existing knowledge and techniques. Any technique could be used in this research, however to check its feasibility two sub-systems needed to be created and tested:

- Keyboard Skills Expert sub-system to assess student's responses. An expert system was found appropriate for the ITS as it could generate correct/expected answers to compare with student's responses.
- Lessons delivery and testing sub-system to interface between a student and an ITS.

A Keyboard Skills Expert system and a small application to interface between a student and the PC were created as mini projects to check and validate the feasibility of the hypothesis. Conclusions were then drawn upon the obtained results.

5.3.2. Mini Project 1: Keyboard Skills Expert

A new Keyboard Skills Expert system was required to assess student responses. To do so, it needed to recognise a task and its parameters (such as 'play a C chord') and to transform the key pressed by the student into a usable format to be compared with the expected answer.

The scope of the new expert system was to deal with Notes and their relationships and chords, which were special collections of Notes. As the required expertise was limited, an Object Oriented (OOP, see 2.2.4 page 38) approach was chosen to develop the expert system.

A *MusicES* object was created that contained all necessary methods to deal with note or chord objects. The aim of *MusicES* was to transform a script (name of an object or relation between objects) into a collection of *note* objects and vice versa. Its expertise could be extended by adding new methods, which applied the corresponding music rules. The knowledge was elicited [see Ackroyd and Hugues (1981), Hart (1986) and Diaper (1989)] from the author, who was himself a keyboard player and keyboard skills tutor.

A simple user interface was created to test the *MusicES* object (Figure 5-5). Another control called *Kbd* was created to represent a keyboard that could be used later with the intelligent system. A collection of *Note* objects could be sent to *Kbd* to light up or down *kbd* keys and mouse clicks raised events delivering newly (un)pressed keys.

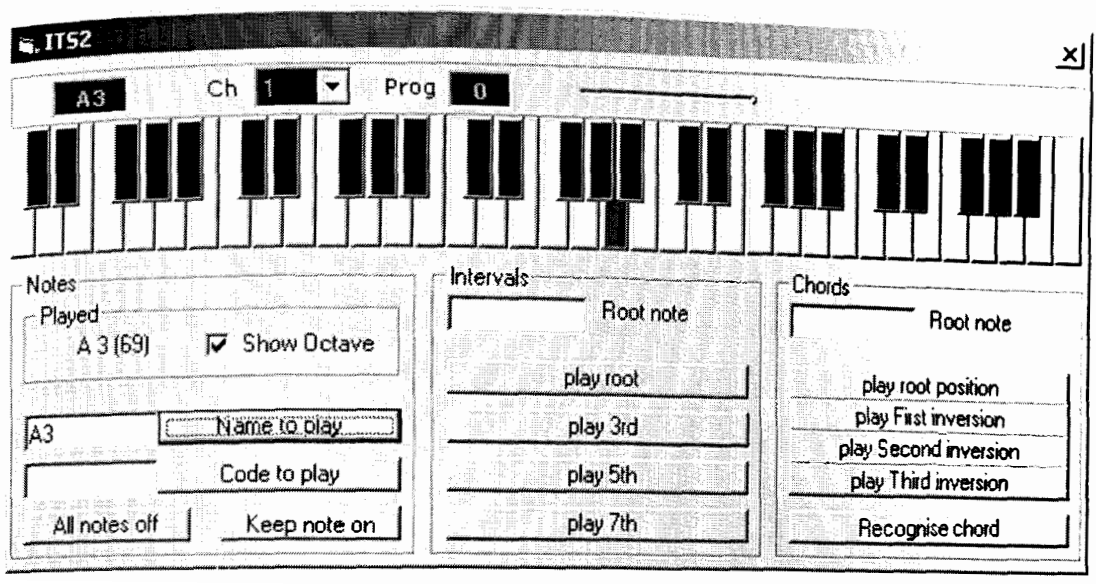


Figure 5-5: Keyboard Skills Expert (2003)

The testing of the *MusicES* object produced satisfactory results and the expertise produced included:

- Note recognition — from script (name or code) or from key presses.
- Interval recognition — as above, with special chord related intervals such as third, fifth and seventh.
- Chord recognition — as above, with major / minor / seventh and inversion recognition.

5.3.3. Mini Project 2: Interfacing Method

A second mini project was aimed at checking that an interfacing method could be put in place for use with the ITS and KAAN. As a reminder, the interfacing methods were:

- *Student → ITS / Computer:*
  - Key presses capture.
  - Time (from Windows).
- *ITS / Computer → Students:*
  - Pre-recorded sequences OR text-to-speech engine.
  - Key lighting (from a MIDI message).

To capture key presses with KAAN, a control implementing a *Transceiver* object (see 4.3.1, page 123) was created to interface with the music technology system both ways (EMIs setting and Keypress capturing). This control, named *TransServer*, could retrieve the changes in key status (pressed / unpressed) of all the keyboards in the music technology system, every tenth of a second. The *TransServer* raised events in case of useful information with the collection of keys and the EMI involved. This control was necessary in case of multiple simultaneous station handling.

A text-to-speech engine (engine producing audio speech from text) was preferred to pre-recorded sequence for the speech as this was more flexible. If scripts to be spoken could be built in real time, parameterised lessons and tests could be generated. A text-to-speech engine generated audio (or spoke) from a given text, itself using AI techniques. Its counterpart also existed to recognise and transcribe spoken words into text. Windows already implemented such an engine for its accessories / accessibility / narrator feature for blind people. Other companies also produced more advanced engines for automated telephone calls handling or other applications. Each voice engine could be fed with various voice fonts for greater flexibility.

Key lighting was controlled by sending MIDI messages (notes on / off) to the appropriate keyboard through KAAN. A Casio LK50 keyboard was preferred to its Yamaha counterpart as the whole key lit up instead of a LED above the key.

An object called *Sequence* was created to deliver sequences of speech and key lighting actions to the students that would become the base of delivered knowledge to students for lessons and tests. A *Sequence* object contained a collection of *SequenceAction* objects that performed single actions such as playing / key lighting notes or chords, speak or wait a certain time. A *Sequence* object was responsible for the sequence playback. Each *SequenceAction* was played sequentially and the *Sequence* had to wait for one *SequenceAction* to finish before starting a new one. *Sequence* objects created the sequence (or collection of *SequenceActions*) from a script of a determined format. The script was a string

composed of chunks describing the actions. Each chunk had a header, a value and a terminator. Five headers were available:

- N- to play one or more notes.
- C- to play one or more chords.
- T- to speak the following text.
- W- to wait the following amount of seconds.
- S- to change speech speed.

The terminator was the character | for all chunks. The value of the chunk differed from the type of chunks. For a W- type chunk, the value to follow was to be a value determining the time to wait (for instance W-1.5|). For a T- type chunk, the text following the header was the text to be spoken (for instance T-I am speaking|). For both N- and C- types, the text to follow was the name of the notes or chords, such as N-Ab| or N-C#3|, or C-Bbm7|. Where the octave was indicated in the note name, only the specified note was played, otherwise all notes corresponding to that name were played. For a chord, the third octave was generally chosen as the chord was played in the middle of the keyboard. Many notes or chords could be played in the same time by separating them with comas (N-C,C#| or C-B,Bb|). To get an 'all notes off' effect, a N-| was given, but any new N- chunk would set all currently played notes to 'off'. The *MusicES* object described earlier was used to procure the notes to be played from the script. An example of simple script is:

T-You know how to play a 'A',|N-A|T-a 'C'|N-C|T-and a 'G'.|N-G|W-1|N-|T-A 'B' corresponds to the white key BETWEEN a 'A' and a 'C'.|N-B|T-a 'D' corresponds to the white key on the right of a 'C'.|N-D|T-And so on up to a 'G'.|N-E|T-E|W-1|N-F|T-F|W-1|N-G|T-G|W-1|T-After a 'G' is a 'A'.|N-A|W-1|N-|T-To summarise|N-A|T-A|W-.5|N-B|T-B|W-.5|N-C|T-C|W-.5|N-D|T-D|W-.5|N-E|T-E|W-.5|N-F|T-F|W-.5|N-G|T-G|W-.5|N-A|T-and A.|W-.5|

Such scripts enabled the creation of sequences of speech, key lighting or any other actions that could be needed in future. Furthermore, as actions were created from scripts, some actions could be parameterised and created in real time, for example for testing purposes. A template script could be given with text

parts to be modified with pre-specified parameters and a list of possible values. This gave for instance:

- Stored Script: T-Please play the note ~3 ~2 a ~1 note.|
- Parameters:
  - 1: A|B|C|D|E|F|G|A sharp|B sharp|C sharp|D sharp|E sharp|F sharp|G sharp|A flat|B flat|C flat|D flat|E flat|F flat|G flat|
  - 2: (-)below|(+)above|
  - 3: [1..12]*1 semitones|[1..6]*2 tones|

➔ Real-time generated script: *T-Please play the note 4 semitones below a E flat note.|*

The expected answer here would be calculated from the MIDI code of an Eb note + (-1) * [4]*1 semitones (a difference of 1 semitone is a difference of a value of 1 in the MIDI code of the initial note).

The interfacing application produced satisfactory results and the Sequence object could generate sequences including speech and collections of notes that could be handled later on by note player, such as the *Kbd* control described in B) or a KAAN engine. Figure 5-6 shows a new control called *KaanKbd*, which aimed at handling *Sequence* objects to interface with a user:

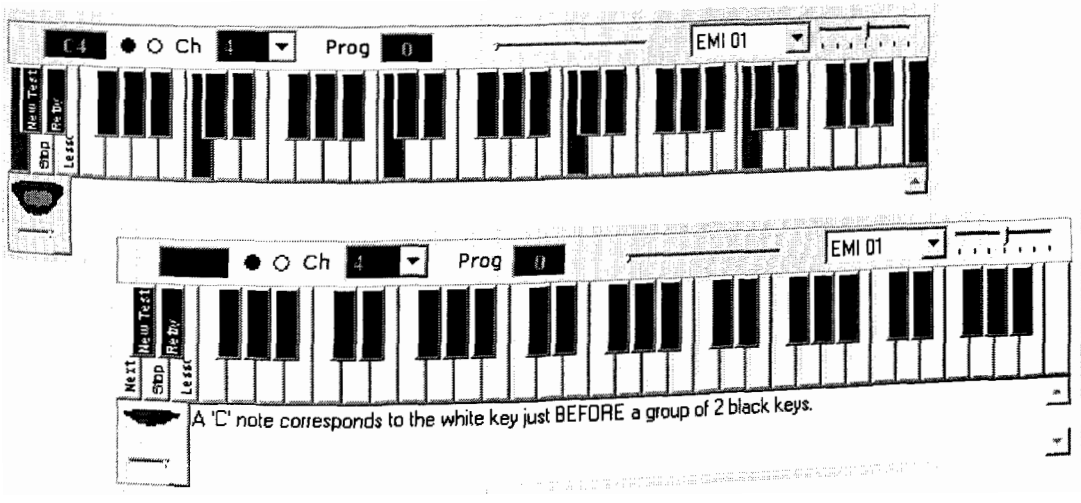


Figure 5-6: KaanKbd Control in operation (2003)

5.3.4. Discussion on automated teaching and testing

Automated teaching and testing proved to be feasible and could be performed with KAAN systems using the objects and controls created in the mini projects. The sequence player could be of some help for the sequential helper described in 5.2.3 (page 174). Customised testing could also be created in real time using the scripting method described in C).

ITS methods could be of some help to customise the teaching of students and to extract useful administrative information from their knowledge model. This could be particularly helpful for teaching basic keyboard skills to new students (year 7) and to follow lessons using keyboards. An ITS could also be applied to any student wishing to learn how to play the keyboard (in the same way as the KAI systems but focused on students abilities).

If this hypothesis was to be fully researched, further work included the determination of:

- The skills to be taught.
- The way and order to deliver these skills.
- An appropriate teaching and assessment strategy.

As concluded in the next paragraph, this hypothesis was selected for further research and all of the above was researched and detailed in Chapter 6.

5.4. Chapter Discussion

Two intelligent methods to assist music teachers and solve remaining issues with the new KAAN – ESAAMS systems were described in this chapter. The assistance could be provided as:

- An intelligent assistant that would observe the teacher and anticipate some habitual tasks so that the teacher could stay focused on the lesson **(Sub-project 1)**.

- Teaching tools that could help students gain appropriate keyboard skills to follow the lessons more easily **(Sub-project 2)**.

It was decided to investigate the feasibility of both methods. Both hypotheses were investigated within mini-projects and proved to be feasible. Both solutions were generic and could also be applied to other applications. It was also discussed that a sequential helper could be implemented and was planned for the next release of the software system.

The help provided by an Assistant to music teachers using KAAN systems could be substantial if the teacher frequently performed sequential tasks, which may not be the case. Moreover, each teacher tended to work differently from others and lesson delivery differed from class to class and student to student. Thus, the help might only benefit a few teachers.

In another case, the help provided by an ITS, without fully replacing the teacher, could be valuable as:

- Students would get similar basic keyboard and music skills.
- Teachers would have time at the beginning of the year to perform administrative tasks and to start knowing the students.

Therefore, it was decided to fully research intelligent teaching methods. This part of the research is described in Chapter 6.

CHAPTER 6. INTELLIGENT TUTORING SYSTEM (ITS) FOR KAAN

Chapters 3 and 4 described the creation of the new KAAN – ESAAMS systems aimed at creating a new infrastructure in music classrooms to help music teachers. Chapter 5 showed that the music technology system offered benefits but some issues remained. Two hypotheses for intelligent assistance to overcome these issues were described and investigated. After research and testing, it was found that a useful and feasible solution for teachers was to implement an Intelligent Computer Assisted Instruction (ICAI) system or Intelligent Tutoring System (ITS) within the new KAAN - ESAAMS systems. The new ITS was to deliver customised keyboard skills to students to help them follow their lessons (using keyboards).

This chapter discusses the creation of the new ITS. Tests were carried out and results are discussed that demonstrate the benefits and drawbacks of the new intelligent system.

6.1. ITS Creation

In Chapter 5 research projects on ITS and teaching techniques were reviewed. Mini projects were described in 5.3 (page 175) to check that an ITS for KAAN system was feasible. A new system – user - interfacing method was defined along with a new simple expert system for basic music concepts.

A 'Push' teaching method (use of a feedback ITS) was chosen instead of a 'pull' method (use of a diagnostic ITS) as explained in 2.3.3, page 59. This 'Push' teaching method was based on the 'Cognitive Load Theory' described by Wilson and Cole (1996) and the 'Constructionism' described by Mergel (1998) and Cavallo

(n.d.1, n.d.2). Both theories were thought to be better suited for the students as students would assimilate new Knowledge that would be remembered longer. Moreover, this method could be implemented on an existing CAI system to make it Intelligent. Therefore, in that case, an ITS system may be described as a CAI system with feedback loops to manage new student's knowledge. This new knowledge is then assessed to re-orientate the teaching to best fit student's learning needs. Knowing this, it was decided to build a CAI system first on which the intelligence (feedback loop) was added to form the new ITS system.

6.2. CAI system for KAAN system

This part of the research began with the creation of a new CAI system to teach music using KAAN. The CAI system consisted of a tutoring base where lessons, tests, parameters and answers where recorded. This information was then handled by a controller that would manage lessons and test delivery. Key press feedback would come either from the *Kbd* control created during the mini projects or the KAAN system. Responses were assessed and the result would be fed back to the student. An automated mode delivered the lessons and tests automatically.

The first step was to create lessons and tests to populate the tutoring base.

Music is a language defined with concepts, fonts, symbols, syntax and rules. Rules could be put together to define an expert system (created in 5.3.2, page 181). Concepts were bricks of knowledge that sometimes needed to be assimilated in a certain order so that the knowledge built remained solid.

Concepts / Sub Concepts / Lessons / Tasks: Knowledge could be decomposed into Concepts (C). These could be further decomposed into n levels of subconcepts (SC). Each sub concept could then be explained in a number of lessons (scripts with eventual examples) each of them having a task (with defined parameters) that could be used for practicing. The last lesson of a subconcept could be used to check that that subconcept had been understood and comprised a task with all the parameters used in the lesson tasks. That subconcept task

usually did not have a script lesson with it as it was used to assess that the subconcept had been understood. This methodology was applied.

Subconcepts sometimes required other subconcepts to be taught beforehand or could be independent. A table indicated their dependencies so that an ordered lesson list could be loaded at startup. From this, the subconcept task of the last subconcept of a concept was called a concept task and was used to test or examine student's knowledge.

For example the concept 'Note', was decomposed into four subconcepts:

- 'White notes' (or keys on a keyboard). Told the name of the notes (A to G).
- 'Sharps' [one semitone above a given (white) note]. The exceptions of B sharp and E sharp were not introduced in that concept, as they were more appropriate with subconcepts or concepts related to scales.
- 'Flats' [one semitone below a given (white) note]. As for sharps, the exceptions of C flat and F flat were not included.
- 'Any' (concept lesson including all subconcepts).

These subconcepts had to be given in that order (sharps and flats could be interchanged) as base note names were needed. However, a new concept was introduced in between to talk about intervals-tones and semitones. Each subconcept had three lessons (except 'any' as it was used as a concept assessment) describing their subconcept with small steps of increasing difficulty.

6.2.1. Tutoring base

A generic tutoring base was designed in MS Access. The database system analysis can be found in Appendix 8.5.1 (page 289).

The tables included in the database shown in Figure 6-1 are:

- *Concepts*
- *SubConcepts* related to a concept.

- *PreviousSC* to determine the order/dependency of the subconcepts.
- *Lessons* related to a subconcept.
- *Parameters* each lesson came with a set task and pre-defined parameters.
- *Tasks* parameterised script with expected type of answer that can be used for many lessons.
- *Feedbacks* expected feedback with what to check and the spoken feedback from the response.

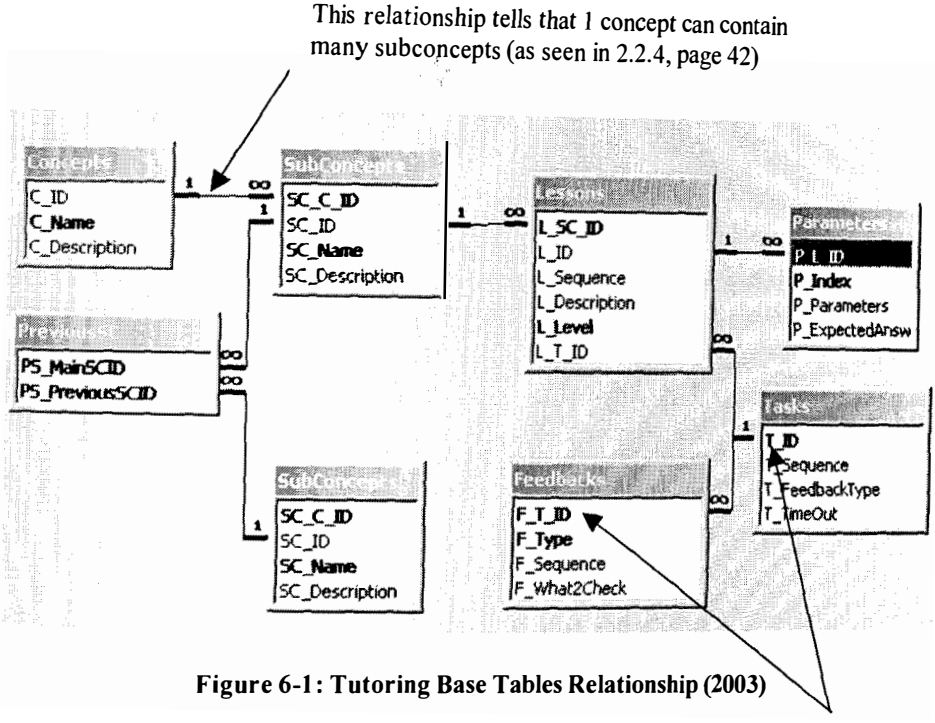


Figure 6-1: Tutoring Base Tables Relationship (2003)

Concepts and *SubConcepts* were defined by a unique ID, their name and a description. The *PreviousSC* table ordered *SubConcepts* relative to each other. With appropriate interfacing, a teacher could order subconcepts.

Lessons were defined by a unique ID, description level and sequence. The level determined the order within the *SubConcept*, as a *SubConcept* could contain more than one *Lesson* to describe it. The sequence contained the script that would build the lesson sequence required to interface with the user. Part of the knowledge was embedded within this script.

Tasks were defined by a unique ID, Sequence, FeedbackType and TimeOut. The sequence contained the script that would build the lesson sequence required to interface with the user. However, this script was different from the *Lesson*'s sequence as it was parameterised. The parameter, depending on the lesson given, could be found in the *Parameters* table along with its expected answer. The expected answer was defined in the script method so that the expert could handle it. The *Tasks* FeedbackType determined the type of expected feedback (for example one note or a chord). The TimeOut set the time to wait before it could be decided that the student could not answer the task.

Feedbacks were defined by a unique ID, Sequence and What2Check. The sequence was similar to a *Tasks* sequence with a parameterised script that could build upon the student's answer. What2Check set assessment rules in order to determine if the student responded well. These rules also determined the type of feedback to give to the student.

For example with a Major chord task:

- Task script for Chord concept: *T-Please play a ~1 chord.|*
- Expected answer (to be used by *MusicES* for assessment): *C-~1|*
- Available parameters (for subconcept Chord Major, lesson 4 (all major unaltered major chords): *A|B|C|D|E|F|G|*
- Randomly chosen parameter: *D*

→ **New task script:** ***T-Please play a D chord.|***

Expected answer: ***C-D|***

Feedback for chord tasks:

- Correct / Incorrect assessment (as shown in Table 6-1):

What2Check	Feedback sequence
Correct (all played notes belonged to chord)	T-Correct|
Incorrect (at least one of the played notes did not belong to chord)	T-Incorrect. You played a £ chord.| *MusicES* tried to determine what chord was played for feedback. If not, just said that the chord did not exist (this depended on the Knowledge and rules entered in *MusicES*, in this research the scope was limited to major, minor and seventh chords).

Table 6-1: Correct / Incorrect Answers (2003)

- Played Notes assessment (in order, as shown in Table 6-2):

What2Check	Feedback sequence
Correct (all played notes belonged to chord)	T-Correct|
Root note was not recognised in played notes	T-Incorrect. The $ is missing.| $ = root note
Third was not recognised in played notes	T-Incorrect. The $ is missing.| $ = third
Fifth was not recognised in played notes	T-Incorrect. The $ is missing.| $ = fifth

Table 6-2: Multiple parameter assessment (2003)

The tutoring base had to be populated and a user interface had to be created. The knowledge to be implemented dealt with music and basic keyboard skills. Possible skills were collected and checked with a teacher advisor. These were:

- Note recognition.
- Intervals recognition (difference between notes).
- Chord recognition.
- Rhythm tapping and recognition.
- Fingering (the way to place fingers on an instrument to play a piece of music).

The first three concepts were selected for implementation to validate the theory and CAI system, as there were many subconcepts to teach. The other two were set aside for future work.

Knowledge was auto-elicited by the author using methods explained by Ackroyd and Hugues (1981), Hart (1986) and Diaper (1989). Diaper (1989) stated that three different representations of knowledge existed:

- *Implicit or tacit knowledge*: a brainstorm was conducted to get as many music terms and concepts as possible. The author then observed music teachers and looked at their music teaching methods to complete the list for various key stages. In parallel, the author looked at CAI and KAI systems (see 2.1.2, page 10) as well as Heinemann's music matters [Heinemann Educational (n.d.)].

- *Intermediate or mediating knowledge: explicit, public representation of the elicited knowledge*. The knowledge was gathered in groups and subgroups that later became concepts and subconcepts, explained with clear and plain terms. The concepts and subconcepts were then sorted in order of difficulty and linked to keyboard playing. Lessons were built by collecting all parameters describing the subconcepts (for example subconcept 'white note' had parameters A, B, C, D, E, F and G). Then, these parameters were gathered in groups of increasing difficulty [for example a C chord (C, E and G notes) was easier to teach than a B chord (B, D# and F#) and therefore was taught earlier].

- *Representation encoded with the expert system*. This part was described when creating the new *MusicES* object.

A teacher advisor then checked this knowledge. The results can be seen in Appendix 8.5.2 (page 295). Concepts and subconcepts found to be best for teaching were (not in order):

- *Notes:*
  - White notes 4 lessons.
  - Black notes (sharps) 3 lessons.
  - Black notes (Flats) 3 lessons.
  - Any 1 Lesson.

- *Intervals:*
  - Alterations:
    - Tones and semitones: 2 Lesson.
  - Major 7 Lessons.
  - Minor 3 Lessons.
- *Chords:*
  - Major 7 Lessons.
  - Minor 8 Lessons.
  - Inversions 5 Lessons.
  - Advanced Chords:
    - 7th 8 Lessons.

Lessons were designed to build complex knowledge in the subconcepts. Each of them had one task with one or more parameters determined randomly from a list of preset options. Usually the last lesson of a subconcept had no lesson sequence. The purpose of the last lesson was to test the entire subconcept with possible options from its lessons. The 'Notes / Any' subconcept was special as it tested the entire concept, meaning that a student was asked to play any existing note (white or black). This type of subconcept was not present for the other concepts. However, tests for chords gathered more and more options as knowledge increased.

The Tutoring base was then populated in parallel with its interfacing application to verify its sequences.

6.2.2. Interfacing Application

The new CAI application consisted of the tutoring base and an interface. It was created in VB6 as were the other controls and applications. The aim of the new CAI application was to deliver preset lessons and tests to a student through the new KAAN system. The teaching strategy was simple and could be automated. It consisted of giving all selected (checked) lessons in order. Lessons were delivered in the order:

- Lesson sequence (if any).
- Lesson task sequence (parameterised).
- Wait for answer (timeout included, first played note for a note, the number of required notes for a chord).
- Assess feedback.
- Deliver feedback sequence (built from the answer and the task, with or without the correct answer).
- If automated, use function keys on the left side of the keyboard to control the lesson delivery:
    - C: go to next lesson if answered well.
    - D: stop / pause the lesson.
    - E: replay lesson or task command.
    - C#: give a new test.
    - D#: redo last test.

The lesson started with the lesson a student selected. The lesson was delivered in the order detailed above and the student had to practice on tests related to the lesson before progressing to the next lesson. The CAI system assessed the answer in a correct / incorrect / timed out fashion using the *MusicES* expert. The answer was given as a key press, which was fed back as a MIDI note code. This note code was transformed as a note / chord name by the expert, and this name was then compared with the expected name for the test. A number of correct answers (in total and / or in a row) could be set to allow the student to progress to the next lesson.

The teaching strategy was built around the idea that students could be delivered any knowledge (lesson) and could train on the new knowledge for as long as they wanted. The CAI system would tell the students if it felt they 'understood' the knowledge and could 'safely' go to the next lesson. The students could then replay the lesson and practice on tests, as many times they wanted. As tests were parameterised and could get a range of options, the new test and redo test buttons were there to practice on new or old parameters. The correct answer could also be fed back to the student to show what was expected.

To handle the lesson delivery a new control was created. The *LessonPlayer* control was responsible for loading lessons, in order, from the tutoring base and to 'play' in a given 'strategy' or 'order'. Each *Lesson* consisted of a *Test*, which dealt with the real-time test script building the expected answer. The *LessonPlayer* played the lesson sequences from its 'library' and the test sequence from the corresponding *Test* object. The note feedback was forwarded by the *LessonPlayer* to the *Test* object, which assessed the answer within an instance of the *MusicES* object. The feedback sequence was then built from the expected answer and the actual answer and fed back to the student via the *LessonPlayer*. The *LessonPlayer* could then check the correctness of the answer to go or not to the next stage.

Here is an example of lesson delivery with task assessment. The lesson is Notes-Sharp-Lesson 2. The sequence was as follows:

System	*Delivered the lesson sequence with speech and keylighting. This particular lesson dealt with sharpened notes and introduced F# and G#.*
System	*At the end of the lesson, the system asks the student to either replay the lesson (function key E) or to practice the current lesson with tests (function key C#). Both function keys were lit on and they sounded to show that an action was awaited.*
Student	*Selects C# (Test). Note that a timer ran to automatically select a function after a determined time (could be set for each student, usually 10s).*
System	*Gave a task: 'Please play a G#' and waited for an answer. A time out timer was launched to limit the task time (depended on the task).*
Student	*Played a C: wrong answer.*
System	*Assessed the answer and fed back to the student: 'Incorrect, you played a C. The correct answer is G#.' G# was lit on to show the expected answer.*
System	*Then offered three options to the student: replay the lesson (E key), Retry the test (here to play a G#, with D# function key) or to have a new test (with another parameter, with C# function key).*

The CAI system counted the number of correct answers for a lesson as well as the number of successive correct answers. A correct answer was counted only once per task, which meant that a student could answer correctly the first or second time (with retry). All other correct retries were not counted. To progress to the next lesson, a student needed to answer correctly:

- 2 tasks, with 2 consecutive correct answers for a task with 1 parameter.
- 3 tasks, with 2 consecutive correct answers for a task with more than 1 parameter.

The CAI system considered the lesson was understood and offered the student the chance to go to the next lesson with function key C. These numbers of total correct and successive correct answers were set as such at the beginning of the research and could be changed if results showed that more or less correct answers were required to allow student progression to the next lesson. Results and test observation showed that they were correct.

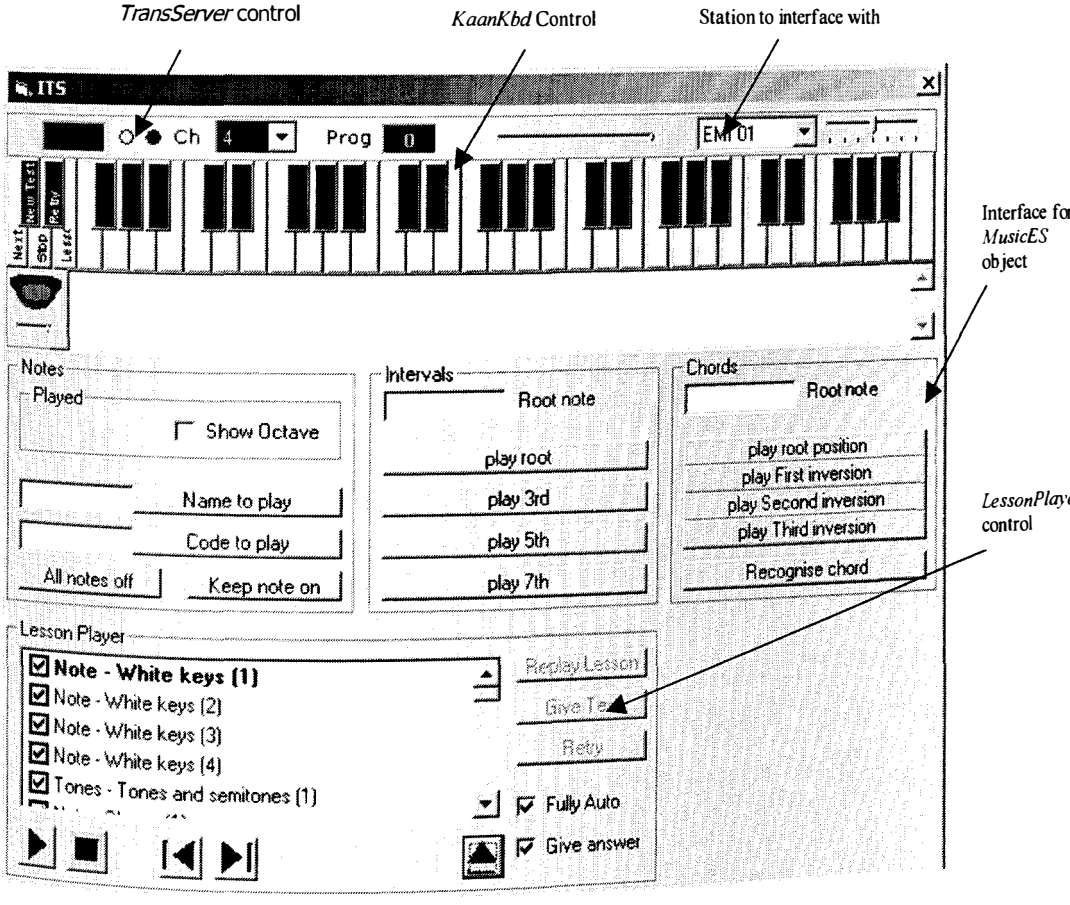


Figure 6-2: CAI System with KAAN/User interface, Expert System and Lesson Player (2003)

Figure 6-2 shows a screenshot of the CAI system Interface with:

- The *KaanKbd* control for PC interfacing in case of no KAAN system.
- The *TransServer* control to interface (send/receive) with the KAAN system.
- The *LessonPlayer* control to manage and interface lessons with the student.

Figure 6-3 shows the structure of objects interaction used in the new CAI system to teach lessons and assess test feedback.

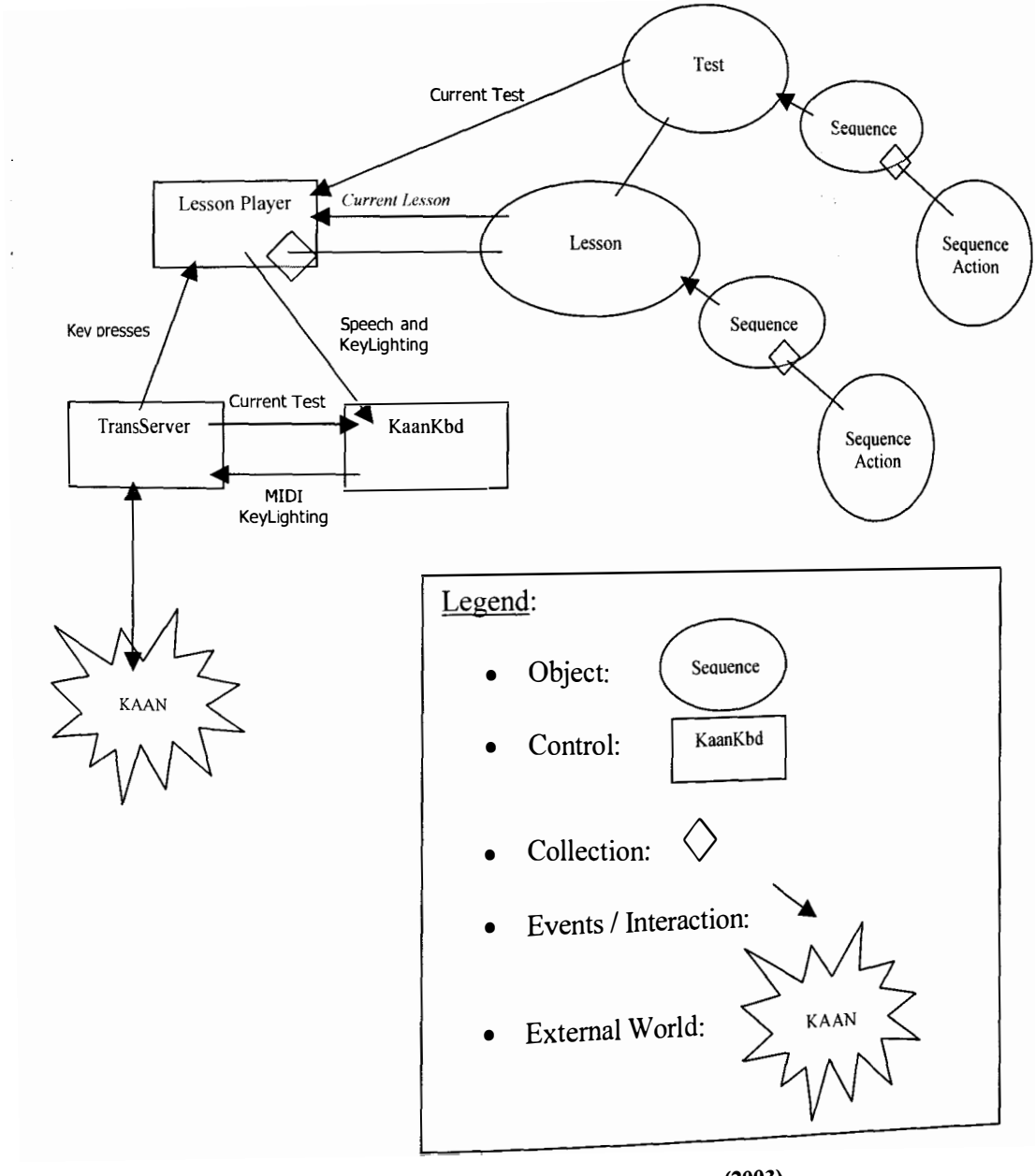


Figure 6-3: CAI Objects Structure (2003)

6.2.3. Discussion

The new CAI system was not fully tested, as the aim was to check the feasibility and the techniques used to deliver lessons and tests for the next ICAI system. Feedback was not stored in a database at this stage for intelligent feedback as the new CAI system could start from any point.

The technique worked, especially with the key lighting and the function keys. However, further issues arose:

- The sequences were built sequentially, meaning that the actions were performed one after the other. Only one sequence could be performed at a time without using multithreading (multi tasking from one application, available in VB.Net not VB6) and no other action could be performed while playing a sequence. This meant that at this point only one student could be taught at a time. Future developments could include the rewriting of the code in VB.Net or another development language allowing the use of multithreading. Another idea would be to distribute the interfacing, using a PC network in parallel to the KAAN system (as shown in Figure 3-26, page 95)

- Microsoft text-to-speech Engine supplied with Windows was used. This engine was easily available but speech quality was poor compared to newer engines and the output sound card could not be set. To teach to multiple students at the same time, the speech/sound sent to the students had to come from different sound cards feeding the 3 or 4 buses of the music technology system. However, the speech engine was easy to use and produced appropriate results. For future work, a better speech engine could be used.

Despite these issues, it was decided to continue researching this sub-project in order to define methods and concepts that could be improved. It was decided not to research the other sub-project investigated in 5.2 (from page 170) as the benefits from this research were considered to be potentially more valuable. It was decided to continue with VB6 (without multithreading, but familiar to the

author) instead of VB.Net (with multithreading, but unfamiliar to the author) because of the lack of training on this new development language.

The next step for the research, after having developed this CAI application, was to investigate methods to apply Artificial Intelligence to customise the teaching.

6.3. ITS for KAAN system

The intelligence to be implemented within the CAI application dealt with the feedback loop and management of student's answers. This was performed through a student model module to allow the tutoring module to deliver the appropriate lessons in defined strategies.

The student model module as well as teaching strategies had to be defined. Then, methods could be analysed and implemented in the existing CAI application to make it ICAI.

Student model module: From reviewed ICAI techniques and more generally Gerlič's third structure (see 2.3.3, page 58), three entities defined the student model module:

- Error Knowledge Base.
- Knowledge State / History.
- Individual Differences.

The first entity dealt with 'learning bugs', as some ICAI systems tried to solve them. The second one obtained a 'mental image' of the current knowledge and could provide a learning curve from its history. The third entity dealt with students' particularities that might bias the teaching, for example the student could be visually, aurally or physically impaired.

It was decided to implement the second entity (Knowledge State / History) only. This entity allowed the capture of information about the students' semantic and working memories. This was considered to be sufficient for the tutoring module to deliver appropriate lessons. Furthermore, this entity also allowed statistical

reports to be generated. Individual differences and the error knowledge base could be implemented later to improve the intelligent system (in order to shorten the teaching by telling the students what was wrong with their knowledge).

The Knowledge State / History entity consisted of 2 parts:

- *Student Knowledge:*

  Every time a concept or subconcept task was answered OUTSIDE a practice exercise (in a test for instance), the subconcept was dated and recorded with its appropriate grade. Grades were:

  - Seen — First time encountered in a subconcept task.
  - Known — When a subconcept task was answered successfully.
  - Understood — When a concept task was answered successfully, every concept's subconcept was upgraded.

- *Teaching Action:*

  Actions received by the student were dated and stored. Tasks were also stored with their parameters and the student feedback. This could be used to generate the error base later.

Teaching Modes: Observing how lessons were delivered for any subject, it appeared that a lesson could consist of up to 3 modes:

- *Teaching:*

  Deliver knowledge to student from Lessons. A lesson consists of a spoken script and visual examples.

- *Practicing:*

  Tasks (Lesson or subconcept) are given to students for practicing. The feedback would not be used to update the Student Model.

- *Testing / Knowledge Assessment (KA):*

  Capture the student's current knowledge. Student's Knowledge could be graded as:

  - Seen — The concept is delivered as subconcepts in lessons and this knowledge is EXPECTED to be known.
  - Known — The concept seems to stay in memory.
  - Understood — The concept is understood and can be used with others.

Tests (subconcept or concept) were given in order of graded seen / known or unseen subconcepts. The Student Knowledge was then upgraded to known (for subconcept tests) or understood (for concept tests) for a good answer or could be downgraded to 'seen' in a case the student did not respond well. The KA process can be of 2 types:

- Push: Each subconcept task is performed in order until one is answered badly.
- Pull: Each concept task is performed in reverse order until one is answered correctly. Then, each concept's subconcept is performed in reverse order until one is answered correctly. Variants can exist either by pushing concepts or subconcepts in the process.

The push method was preferred so as not to de-motivate students. The push method executed only when students did not know a concept, which was considered better for self-motivation and esteem.

For the KA and practice processes, students' answers needed to be assessed. In the new CAI application, these answers were assessed as 'right' or 'wrong', which did not give an accurate idea of the student knowledge. Therefore, the answers were 'marked' with 5 different grades, taking into account the time to answer:

- Incorrect and Slow.
- No answer (check if question was understood).
- Incorrect and Fast (check not a silly mistake).
- Correct and Slow (check not a lucky answer).
- Correct and Fast.

6.3.1. Modifications to the tutoring base

The tutoring base needed to be modified to include tables for the 'Intelligent modules' as shown in Figure 6-4. At that point, only the Knowledge State / History (as seen in 6.3, page 201) part of the intelligence was implemented. This part included:

- Student details **{1}.**
- Student Knowledge (related to students and subconcepts. A history of the modification of this knowledge was also implemented) **{2}.**
- Student teaching actions (related to students and lessons/tasks with history, parameters given, responses and feedback) **{3}.**

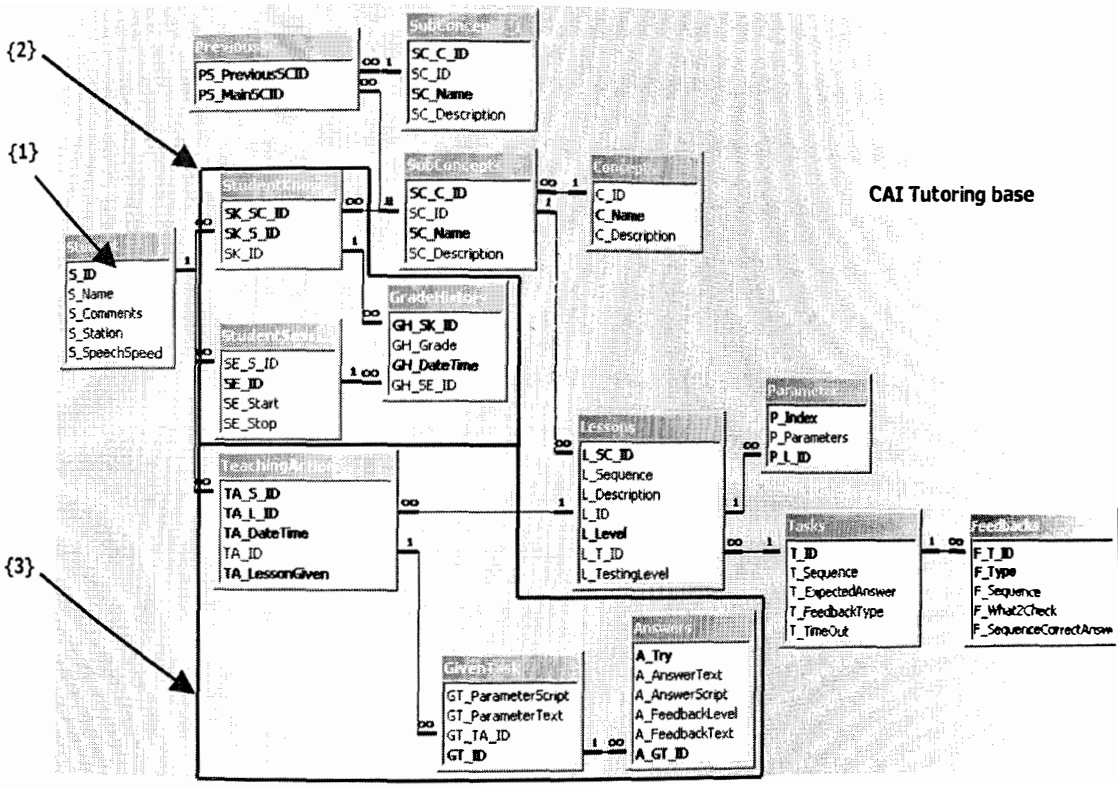


Figure 6-4: Final ITS base, including Intelligence bases and the Tutoring base from former CAI system [relationship model changed from Figure 6-1 (page 191) to better fit]

(2003)

So far, the intelligent system could manage:

- Concepts, subconcepts, lessons and tasks in a given order.
- Students and the modelling of their knowledge.

One important teaching feature was missing: how to deliver efficiently the three teaching modes presented in 6.3, page 202. This feature was called a teaching strategy.

Three teaching modes were presented in 6.3 (page 202), which described a different part of the knowledge transfer between a teacher and students. There existed multiple ways to put them together in order to maximise the transfer and

to check that a student had assimilated the knowledge correctly. These ways could be called a teaching strategy. Three strategies that could be applied in the developed ICAI system are:

- Non or semi customised teaching.
- Semi or fully customised teaching (type 1).
- Fully customised teaching (type 2).

These strategies took into account basic assumptions and rules:

- There was no external intervention except to start and stop the process.
- A student could interact with the intelligent system only from a keyboard.
- At the beginning, all processes might include an option for the student to either practice seen concepts or to go straight to teaching matters. Tasks for seen lessons could be used for practice (straight after the lesson or whenever else in a 'practice mode').

All three strategies are detailed below and then compared. Any of them could be used in the ICAI system, however only one was selected and implemented. Future developments could include a strategy selection for the teacher to use for the automated teaching.

S1 - non- or semi-customised teaching: This strategy only considered the teacher's lesson plans, and all students were taught the same knowledge whatever prior knowledge they had. Two operational modes appeared (see Figure 6-5 and Figure 6-6):

- Perform the testing Online (S1-A).
- Perform the testing Offline (S1-B).

Online meant that testing was performed during the teaching process, whereas offline meant that tests were given outside the teaching process.

S1-A was also split into two:

- In S1-A1 and a variant of S1-B, tests grades were obtained as indicators ONLY on the instantaneous student's knowledge and the teaching carried on without taking care of it. This strategy was considered as non-customised as no student modelling was used to deliver the lessons.

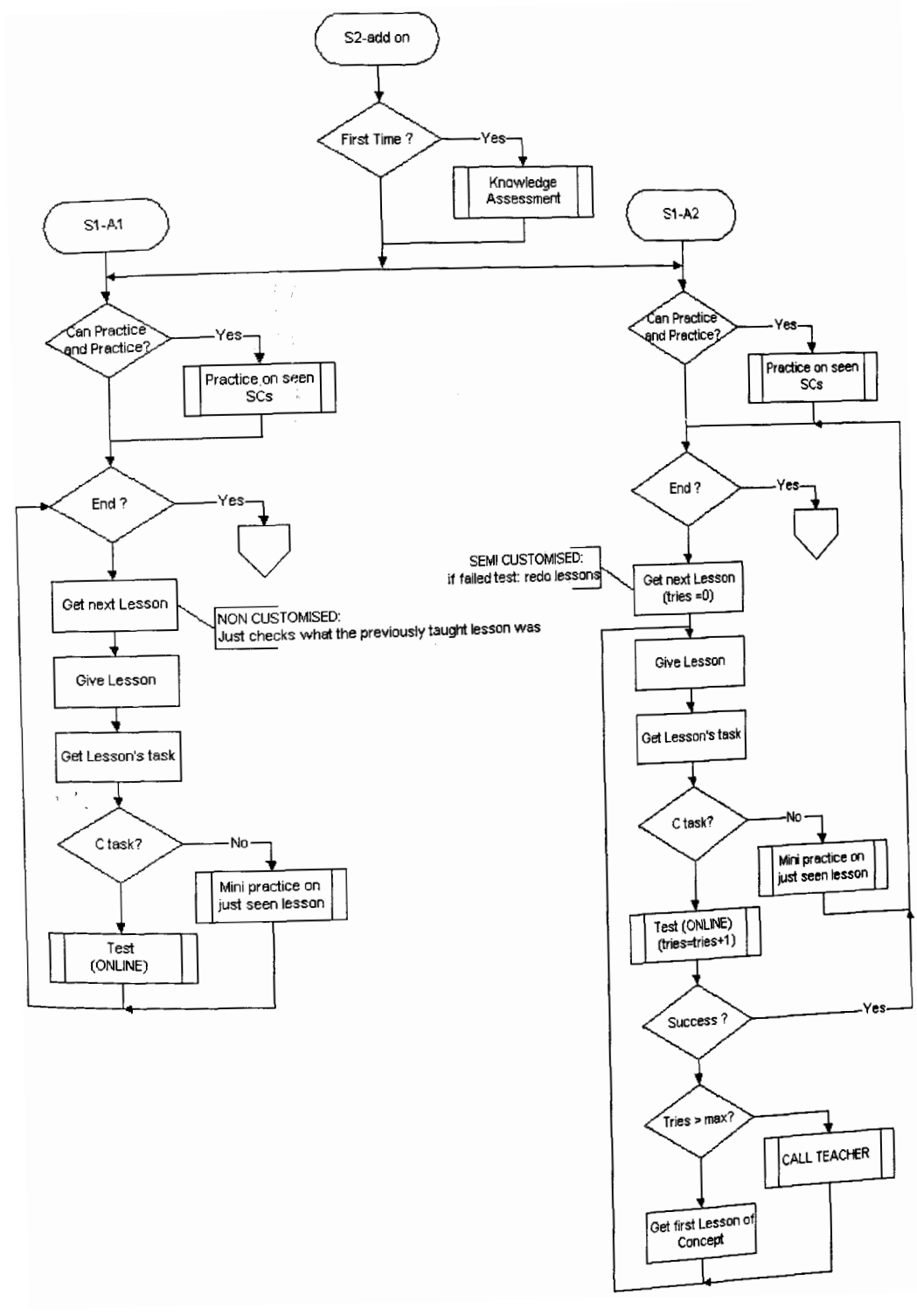


Figure 6-5: Strategies S1 and S2 (types A1 and A2, 2003)

- In S1-A2 and another variant of S1-B, test feedback was taken into account to carry on or not the teaching, or to set the next lesson to teach (in S1-B, after the results of offline tests). In SI-A2, the lessons could be re-delivered a certain number of times until the concept / subconcept was understood and the teacher could be called if a student had problems with a concept.

S2 – semi- or fully- Customised teaching Type 1: Uses S1 processes but a Knowledge Assessment process is carried out the first time to capture the student's prior knowledge.

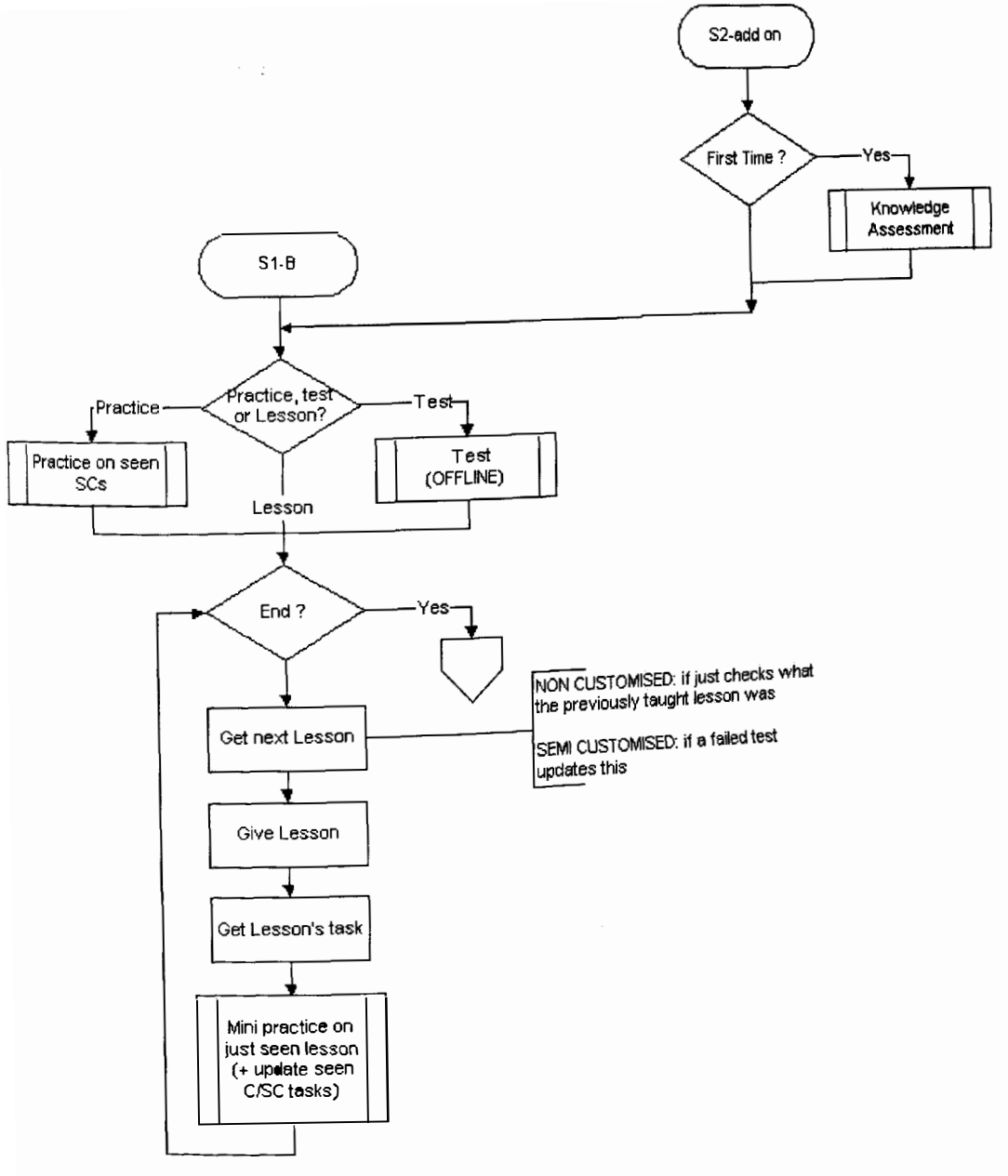


Figure 6-6: Strategies S1 and S2 (type A3, 2003)

S3 – fully-Customised teaching Type 2 (Figure 6-7**):** S3 differed from the first strategies as testing and Knowledge Assessment were considered as similar and performed online at the beginning of every lesson. Each subconcept grade was up/down graded during that process and teaching was customised to the new captured student knowledge. The lessons and tasks were then only used for practicing.

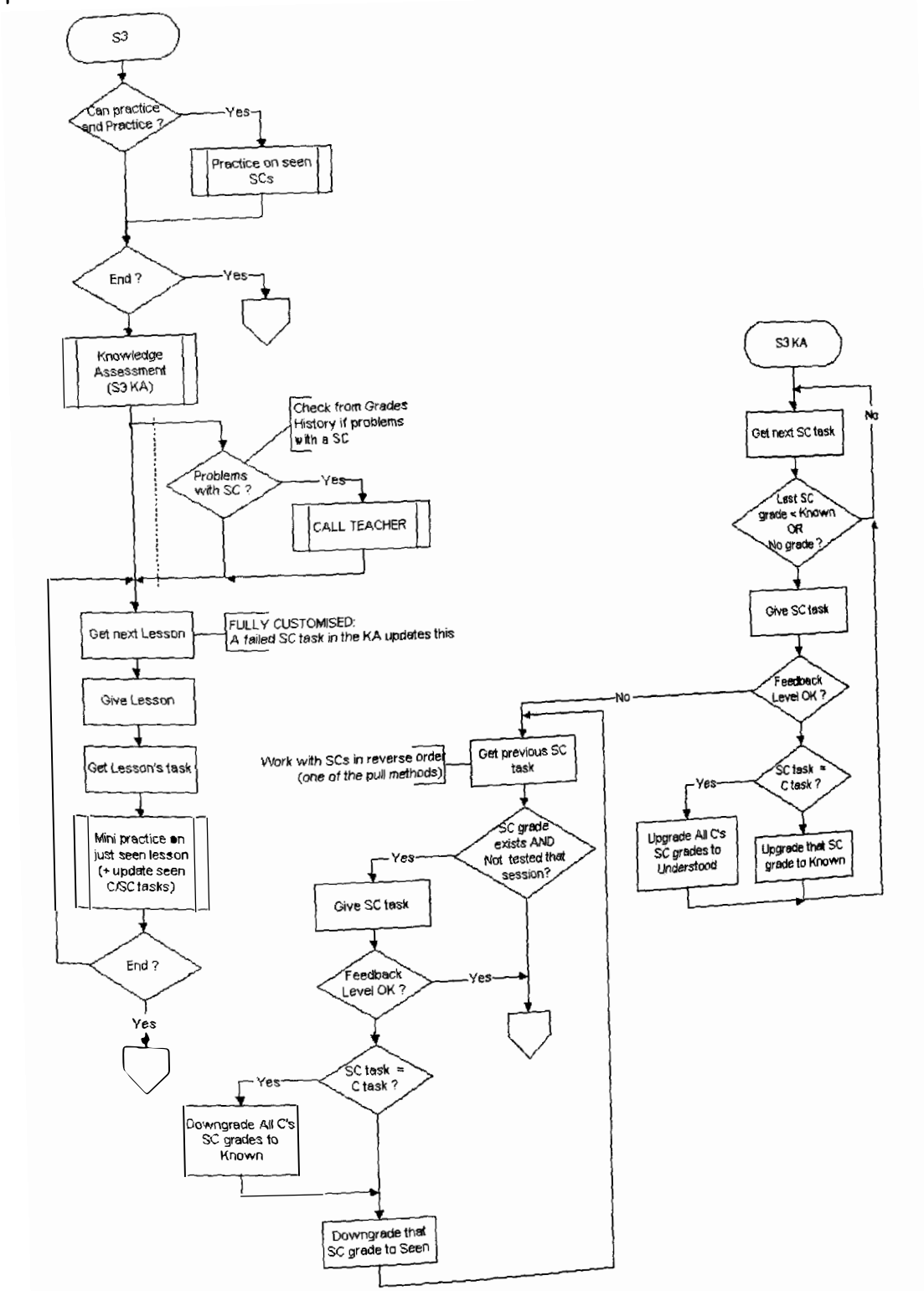


Figure 6-7: Strategy S3 diagram with Knowledge Assessment routine (2003)

6.3.2. Discussion

Strategies S1 and S2 were seen in operation in school classrooms. S3 considered the individual student, which could only happen with personal teachers. It took into account knowledge varying from one teaching session to another.

Table 6-3 shows a comparison between the different strategies, highlighting their drawbacks and advantages.

TYPE	DRAWBACKS	ADVANTAGES
S1	• Non- (S1-A/B1) / semi- (S1-A/B2) customised • All students are considered to be of the same knowledge level • No teaching-flow control (S1-A/B1) ➔ *Students HAVE to follow lesson plan* • No backward process / knowledge downgrading if matters are forgotten ➔ *Can be difficult for weaker students and boring for the stronger*	• All knowledge will be delivered on time (S1-A/B1) • Customised teaching-flow control (S1-A2) ➔ *Students are entitled re-tries in case of unsuccessful tests*
S2	• Semi-customised (S2-A/B1) • No backward process / knowledge downgrading if matters are forgotten • No teaching-flow control (S2-A/B1) ➔ *Students HAVE to follow lesson plan* ➔ *Can be difficult for weaker students and boring for the stronger*	• Fully-customised (S2-A2) ➔ *Students' knowledge is assessed and students are entitled re-tries in case of unsuccessful tests* • Students prior levels are considered (but just once) • All knowledge will be delivered on time (S2-A/B1)
S3	• No guaranty all knowledge will be delivered and on time ➔ *The teacher may need to control student's progress from time to time to check the student is not stuck at some point.*	• Fully-customised ➔ *Students' knowledge is constantly assessed and students are entitled re-tries in case of unsuccessful tests* • Students prior levels are considered (every time) • Backward process / knowledge downgrading if matters are forgotten

Table 6-3: Comparison of the described teaching strategies (2003)

The Knowledge Assessment mode of strategy S2 could only use a Push method (see 6.3, page 203) as it was run only once. It was the same with strategy S3 for the first time, however it needed to be different at other times and to use a Pull method to allow knowledge downgrading.

Strategy S3 had a drawback that could be overcome, as test grades were stored historically. The intelligent system could check multiple unsuccessful tests and then decide on an action, which was to call the teacher as the subconcept lessons were already delivered a certain, pre-determined number of times.

As Strategy S3 had only one drawback and was fully customised, this strategy was selected for implementation in the new ICAI system.

6.4. Testing

The new ICAI system was created including the new tutoring base and the new strategy detailed in Figure 6-7 (page 208). A *StudentManager* control was created to manage and load students to the *LessonPlayer* control. This control now managed lesson and test delivery using the chosen strategy. Any action was recorded in the database against the loaded student and a Knowledge Assessment mode updated the loaded student's knowledge. The new application interface was as shown in Figure 6-8.

The bottom window was a control called *StudentKnowledge*. The idea behind this control was that the student learning curve could be drawn if the intelligent system could get the level of the student knowledge at different moments in time. This knowledge level was calculated from the knowledge grading history table. It was equal to the addition of all the grades of subconcepts dealt with at a certain date using grade values such as:

- Seen=1.
- Known=2.
- Understood=3.

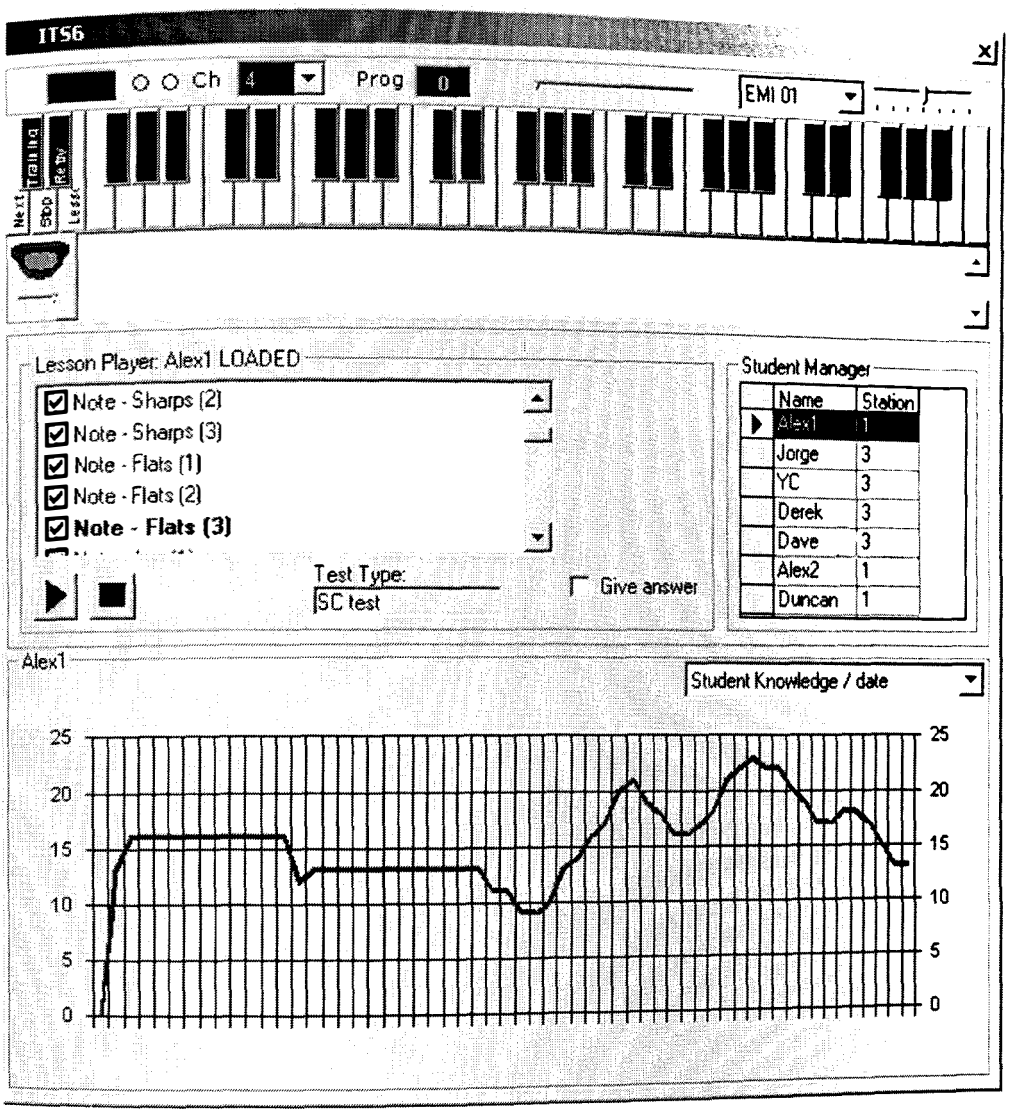


Figure 6-8: ICAI system Interface (2003)

A learning curve consisted of a succession of knowledge levels plotted against knowledge updates or sessions. A knowledge level was calculated as the sum of all the knowledge grades of all sub concepts graded at one moment. As subconcepts were seen one after the other and could be updated (upgraded or downgraded) within a KA session, learning curves could be obtained. The learning curve shown in Figure 6-8 was plotted against dates or, more accurately, against each knowledge update. The learning curve could also be plotted against sessions; a session being the time spent on the intelligent system by a student between the automated teaching starting and stopping. This produced curves like the one shown in Figure 6-9 for Student Alex1 (seen in Figure 6-8). Student Alex1 (in fact the author) was only used to test and debug the intelligent system before other students tested the system.

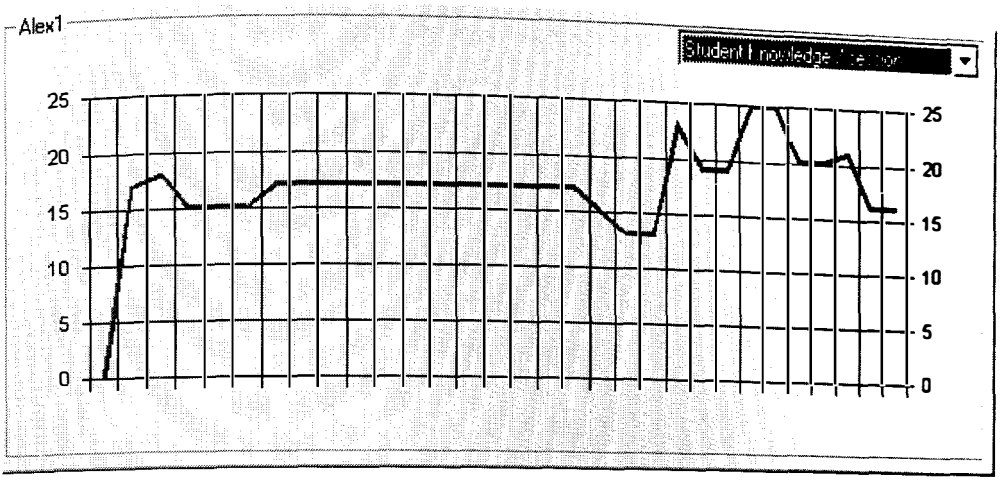


Figure 6-9: An example of a learning curve against sessions (2003)

6.4.1. Testing method

The new ICAI application (or ITS) was written and tested with several 'students'. Details about the students who tested the intelligent system at the time of writing are in Table 6-4:

Student ID	age range	gender	Comments	prior knowledge (music / keyboard)	sessions
A	40-50	M		Medium / Medium	5
B	20-30	M	Spanish	Beginner/Beginner	2
C	20-30	M	Malaysian	Beginner/Beginner	3
D	40-50	M		Beginner/Beginner	2
E	40-50	M	Teacher Advisor	Expert	#
F	20-30	M	French	Medium / Medium	2
G	20-30	F	French	Beginner/Beginner	2
H	30-40	F		Beginner/Beginner	1
I	14	M		Beginner/Beginner	1
J	15	F		Medium / Medium	1
K	30-40	M		Medium / Medium	1
L	20-30	M		Beginner/Beginner	1
M	40-50	F		Beginner/Beginner	1
N	18	M		Beginner/Beginner	1
O	20-30	M		Medium / Medium	1
P	11	F		Medium / Medium	2
Q	16	M		Beginner/Beginner	1
R	15	M		Medium / Medium	1
S	8	M		Medium / Medium	1

Table 6-4: Students' details (2003)

A Teacher Advisor (student E) checked and validated the lessons and tests scripts as well as the knowledge delivery method and order.

Testing consisted of running the intelligent system with the students. Each session action was stored in the database and a learning curve could be plotted from the data. All sessions were observed and notated, and some videotaped to amend and validate written observations. Each student was then questioned to check the immediate after-teaching views of the student. The answers to these questions were considered in improving the new intelligent system. They were stored on spreadsheets along with system changes to get a status of the intelligent system and the students. Post-teaching questions were:

- What have you learnt?
- What was a problem?
- How could it be improved?
- What was good or worked well?
- Anything else?

These testing sessions were designed to:

- Debug the intelligent system.
- Validate:
    - The Interfacing method.
    - New Knowledge transfer (teaching) methods.
    - New Knowledge assimilation and remembering data.

6.4.2. Results

Results from post-teaching questions can be found in Appendix 8.5.3 (page 300). The ITS changes are indicated after testing sessions with bug fixes. At the time of writing twenty-nine tests had been performed with nineteen students (aged from 8 year old to the age range 40-50).

Some examples are presented for students, to explain teaching effects. Examples were grouped to show patterns for discussion. Examples:

A) Example 1: Initial tests to improve the interfacing.

B) Example 2: Beginners.

C) Example 3: Case of students D, M and Q.

D) Example 4: More Advanced students (Case of students A, F and S).

A) EXAMPLE 1: INITIAL TESTS TO IMPROVE THE INTERFACING

Five tests were used to improve interfacing between the intelligent system and students. After these tests, a new version of the ITS was designed. A later version of the ITS was designed after running four more tests. Bugs and crashes occurred during these tests. Misunderstanding on the interfacing also caused test interruptions or actions from the author, who observed the sequences. Tested students were students A, B, C and D, all these students re-tested the intelligent system later with newer versions of the ITS.

Results from the tests showed that:

- Lesson sequences needed to be rephrased, for better understanding and also to improve sequencing between speech and keylighting.
- Better voices were required for better understanding.
- Better routing and advising through the lesson was required along with motivators. This was achieved with transitions using speech directives and function keys lighting. Moreover, a special introduction was played at the first session to explain the intelligent system operation. Not too much speech was needed (so as not to get the students bored).

Student A tested the intelligent system twice at this stage, more to find flaws and faults than for teaching. This added to comments from student E allowed for quicker debugging of the new intelligent system.

B) EXAMPLE 2: BEGINNERS

Beginner students were students B, C, D, G, H, I, L, M, N and Q, ranging from age 14 to 40-50. Both female and male, English and foreign students tested the intelligent system. The number of sessions on the intelligent system varied from 1 to 3. As some students had only one session with the intelligent system, the sub-concept Interval-Tones and Semitones gained more feedback. This subconcept was placed just after the first subconcept (Note-white keys) and before the two subconcepts Note-sharp and Note–flat to introduce the term 'semitone'. This term was rarely introduced by music teachers, especially to describe sharps and flats. That was why many students (even more advanced ones) failed in the KA session at this subconcept. However, this subconcept was left at this position and was useful to determine early how students reacted with stress and more difficult concepts. Once they passed this subconcept, their lessons appeared easier.

Results from the tests showed that:

- Students learnt differently:
    - Some hated the intelligent system as its principle relied on teaching by doing and practising, leaving the students to try to understand their errors. Usually these students had problems with the semitone subconcept. Please see section **C)** for further details.
    - Some found that the intelligent system was performing well and that they learnt well. They liked the idea of getting small, manageable bits of knowledge and the ability to directly practice them. Second and third sessions for some students showed that the knowledge was assimilated correctly (from short term to longer term memory) as the KA did not downgrade their knowledge level. This can be seen in Figure 6-10, which shows the learning curve of student C after three sessions (28/02/03, 07/03/03 and 13/03/03). Usually, these students used the function keys well and would have benefited from a pause in the middle of a teaching session to practice freely on the learnt concepts. It was also found that a 'no answer' function key was needed during the KA session as some students waited for the timeout to have feedback from the intelligent system.

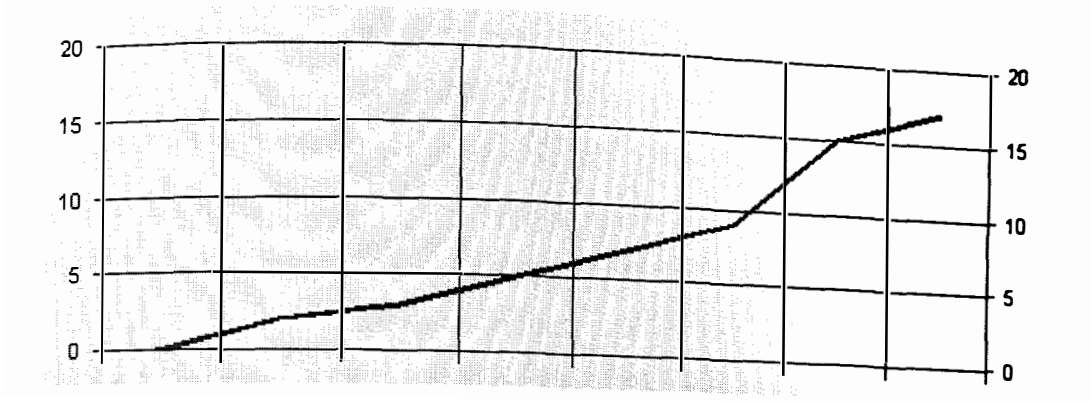


Figure 6-10: Learning curve for student C after three sessions (2003)

- Some students found the intelligent system boring at some point because either the bits of knowledge were too small or the teaching advice became annoying. That was why interfacing parameters were set against students to provide a better teaching experience. Future work could also include a method to assess whether students felt the lessons were too easy to move to subconcept or more difficult lessons.

The speech engine caused problems as even English people had issues understanding what was said and tasks failed because of it. However, after some time, students tended to get used to the voice and performed better. It was also found that the interfacing methods depended on the environment: key lighting could not be seen with too much sun and speech could not be heard with rain dropping on the roof.

C) EXAMPLE 3: CASE OF STUDENTS D, M AND Q

Students D, M and Q had problems with the intelligent system. All said that they did not like the intelligent system and that they learnt nothing. They also became stressed and annoyed.

Student D had two sessions, one in the debug phase discussed in **A)** and the other with the latest version. After the first session, it was found from the information gathered in the post-teaching interview, that the intelligent system

lacked guidance, level confirmation and motivators. This left students not knowing what to do, why things were needed and at what level the lesson was. This, added with a lack of motivators depressed students, who felt blocked and could not learn further. Following this feedback, motivators (for example 'well done') were added after all successful training sessions. This proved to be efficient, even with advanced students, who had difficulties with the last lessons.

Student D performed a second session with the latest version and finished apparently happier. This student wanted to practice on a subconcept until it became instinctive. This was allowed by asking for more tests instead of going to the next lesson and explained why the student did not go far in the teaching. Future work could consist of providing another level of advice as well as error solving. It could be assumed that such students might have difficulty trying to understand their errors by themselves.

Students M (Female, 40-50) and Q (Male, age 16) also had problems with the intelligent system. Student M was frightened by technology and her learning was blocked by using it. She was stressed and had issues using functions keys as she relied solely on speech and did not look down at the keyboard to use it. Her feedback confirmed the author's assumptions (that she was afraid to use technology, and that she would have issues to learn using the new intelligent system) as she said that she was not good enough to learn with the system. Student Q was also stressed, but for another reason. He did not like someone watching him not performing well. He said afterwards that headphones, test videoing or an error debugger could have helped him perform better.

D) EXAMPLE 4: MORE ADVANCED STUDENTS (CASE OF STUDENTS A, F AND S)

Students A, F and S performed better than the average. Student A (English, 40-50) tested the latest versions of the intelligent system more actively (two sessions of a total of 95 minutes) than he did in example **A)**. Student A played the guitar and had a keyboard at home, which he did not use often. The author taught student F (French, 20-30) how to play the keyboard with another teaching

method. This method relied on practising on songs of increasing difficulty and learning concepts as they appeared in the songs. However, this teaching was performed in French, with French notations:

Notes - English / French music concepts names:

- *Note names: A, B, C, D, E, F, G / La, Si, Do, Ré, Mi, Fa, Sol.*
- *Sharp - Flat / Dièse, Bémol.*
- *Note, tone, semitone, chord, interval, minor, major / Note, ton, demi-ton, accord, interval, mineur, majeur.*

Student F finished all lessons after only two sessions after a total of 127 minutes. Student S (English, 8 year old) only had one session (31 minutes) and completed half the lessons. It was noted that student S played the piano and that he also learnt it differently as some concepts were new to him.

For these three students, fast progression and as little talking as possible was required. They apparently enjoyed the challenging tasks given by later lessons and even if pressured, wanted to get more.

6.5. Chapter Discussion

This chapter described the creation of new ITS / ICAI systems for music. At the time of writing twenty-nine tests had been performed with nineteen students (aged from 8 year old to the age range 40-50). Results from testing showed that the ITS method and structure worked well if *individual parameters* could be set for better interaction.

The ITS diagram from Gerlič (shown in Figure 2-14, page 58 or next page in Figure 6-12) is now used to compare and contrast with the new Lassauniere systems and parts of the ITS diagram were used to create new prototype intelligent systems during early research work. Both diagrams are shown in Figure 6-11 and Figure 6-12.

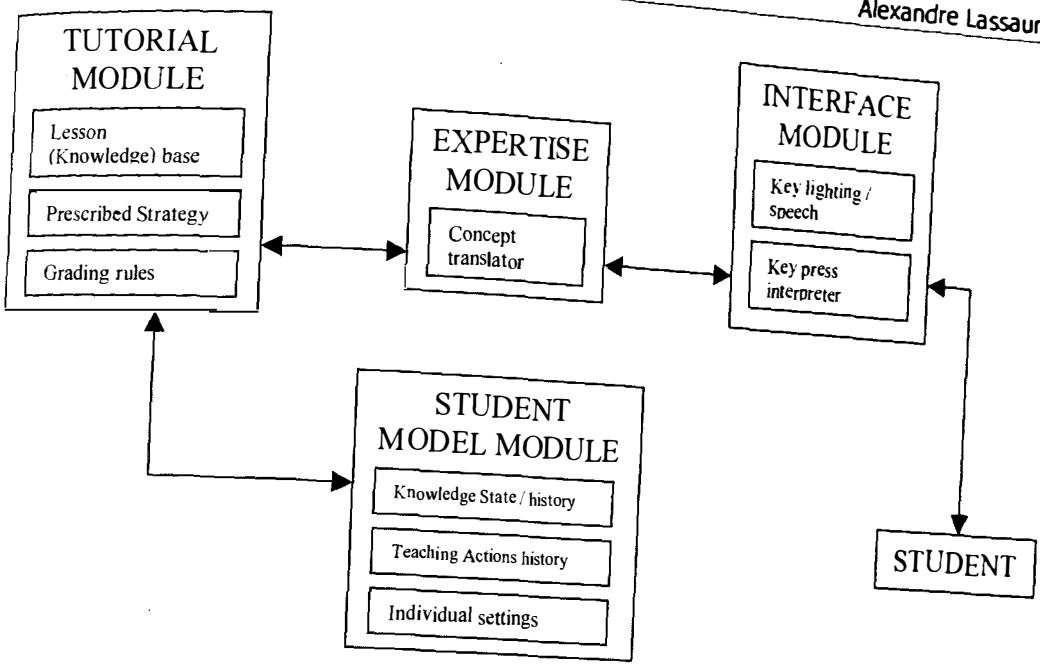


Figure 6-11: Lassauniere's ITS structure (2003)

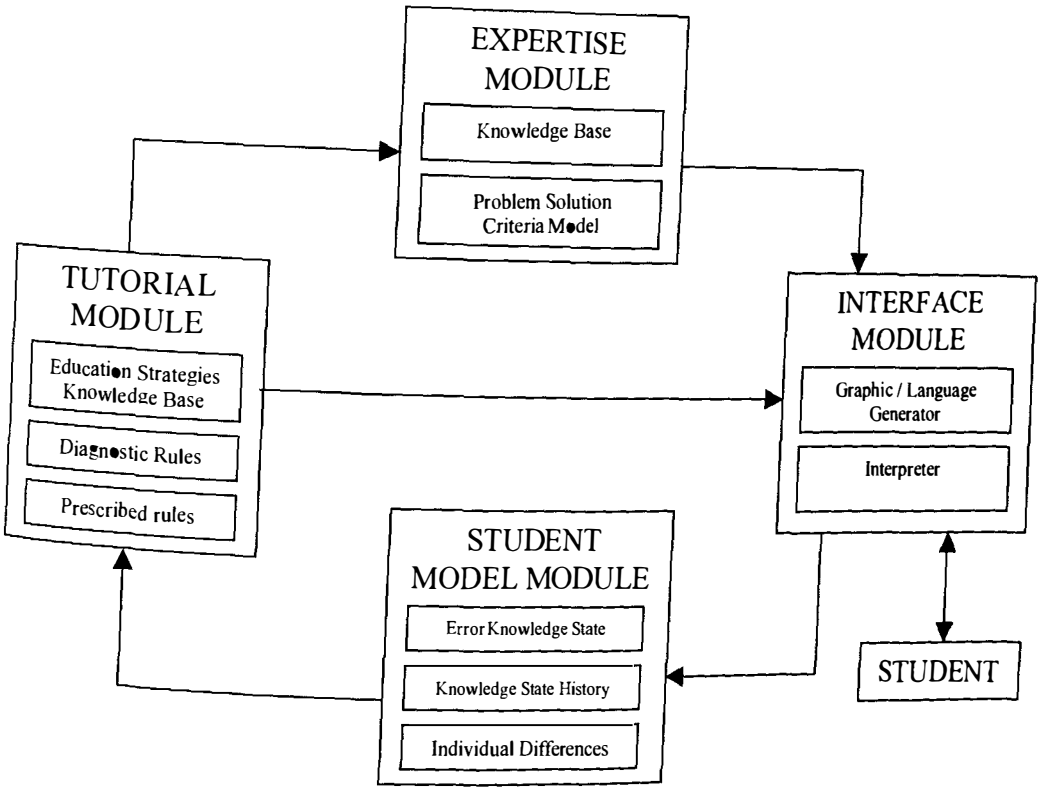


Figure 6-12: Gerlič's ITS structure

[from Gerlič (1998)]

In both cases, students interacted with the Intelligent Tutoring Systems through an *Interface Module* that allowed students to provide and receive information. In the intelligent system created by Gerlič, the *Interface Module* was an integral part of the system and was connected directly to all of the sub systems. In the new Lassauniere intelligent systems the *Interface Module* only worked to the new *System Expert*, where:

- A sequence of actions was played to the student (as speech and key lighting).
- The new *TransServer* control captured key pressed by the students.

This allowed the *Interface Module* to be updated, improved and changed without affecting the rest of the intelligent system.

In the new Lassauniere intelligent systems, the *System Expert* (expertise of concept translation, between MIDI note codes and literal note / chord names) was used to translate tasks given to students and their responses into a different format that was suitable for both the *Tutor* and *Interface Modules*:

- Lessons and tests music concepts were translated as speech and keylighting (MIDI codes sent to the keyboard).
- Pressed key MIDI codes were translated as a music concept answer to be compared with an expected answer.

The Gerlič intelligent system used an *Expertise Module* to deliver managed knowledge and problems / tasks to a student from strategies set by a *Tutorial Module*. This *Tutorial Module* was used to manage knowledge delivery from the student knowledge state and student's answers. Rules and strategies for teaching and assessing were defined to manage the knowledge delivery appropriately.

However, the new Lassauniere intelligent systems used the fact that a physical teacher was the subject expert, the lesson manager and the assessor. This was translated in the intelligent system as the *Tutorial Module*, where the knowledge base management, teaching strategies and answer assessment were performed. The *System Expert* was just used as an intermediate layer to interface with the *Interface Module*. The *Tutorial Module* was responsible for:

- Assessing current student's knowledge status with a predefined strategy, using prior information from the *Student Model Module*.
- Delivering the knowledge appropriately from the new tutoring base using the predefined strategy and current student's knowledge status.
- Training/Testing the taught knowledge with tasks from the new tutoring base, which parameters were generated in real-time.

This change in the intelligence feedback loop allowed better and faster management of the information kept in the tutoring base (for medium to longer term memory). Real-time management of short-term knowledge from tasks' answers assessment was preferred to a slower computed update from newly stored information from the *Student Model Module*. This was because it kept the interaction with the *Student Model Module* as a strict minimum. This was allowed only:

- Because there was no misconception debugging (use of a 'push' teaching method). However, this was partly planned, as a *Teaching Actions History* was stored in the *Student Model Module*, for future use if required.
- If the teaching method used the predefined strategy and knowledge grading rules.

Implementation of misconception debugging required a change in the predefined strategy to allow for bug pattern recognition. Note that this did not mean changing the interaction between modules, the *Tutorial Module* having to keep its central position for the reasons stated above. The teaching method used in the research seemed to produce good results when compared with several methods. One issue was that the implemented teaching method was unique and inflexible. Teachers needed to be given more control over the teaching method (from predefined templates or not) to use the new ITS as an assistant and not to view it as a replacement intelligent system (a common fear of people thinking that computers or robotic-entities would replace them).

Individual Settings in the *Student Model Module* were used at a later stage in the testing to store individual settings on the Interface delivery parameters (voice

font, speech speed and others). Future work could include the implementation of more individual settings to take into account in assessment.

Along with the new ITS system, new systems and methods were explained:

- New Systems:
  - New generic tutoring base and methods to teach with scripting.
  - New Interfacing system (with KAAN, keylighting and text2speech technologies).
  - New controls to display learning curves.
- New Methods:
  - Teaching Modes [Teaching, Practicing, Testing / Knowledge Assessment (KA)].
  - Knowledge and assessment grading.
  - Teaching strategies (S1, S2, S3).
  - Management of short, medium and longer term memory.

The status of the new ITS/ ICAI system for the new KAAN system was:

- Fifty-one preset music lessons.
- Nineteen students tested the intelligent system in twenty-nine sessions.
- Strategy S3 implemented.
- Automation and individual differences (speech).
- Interfacing method.
- Knowledge display.

Potential future work on the existing new intelligent system includes:

- Implement a multi-user ITS using:
  - Multithreading (available in VB.Net).
  - Better speech engine and voices.
  - Interfacing Distribution.
- Implement a strategy manager to teach against customised strategies and a lesson builder for the teacher.
- The use of individual differences in answers assessment.

CHAPTER 7. DISCUSSION AND CONCLUSIONS

This chapter describes the conclusions and recommendations from the research. The structure of the chapter is as follows:

- Summary of the research work programme.
- Resolution of research aims and objectives.
- Key research successes and contribution.
- Improvements to this research (limitations and enabling technologies).
- Suggestions for future work.
- Thesis conclusion.

7.1. Summary of the research work programme

The encompassing aim of this research was to create a new music technology system and new methods to help music teachers teach music.

The work progressed in two phases. Firstly, the creation of a new music technology system called Keyboard And Audio Network as a whole (hardware, firmware and software). This led to the creation of a new information system called Electronic Student Assessment And data Management System. The second phase of the work reviewed feedback from teachers using the new music technology and information systems and led to creation of a new Intelligent Tutoring System aimed at teaching basic music-keyboard skills to students using the new music technology system.

The research on a new music technology system was achieved by creating new Electronic Music Interfaces, which could work stand-alone or be remotely

controlled from a control device. A new communication protocol was defined for control and new control software was created to interface with the new music technology system. Requirements from teachers led to the research and implementation of a new information system within the control software. The success of this new feature led to the creation of a new teaching-centred information system.

Research on Intelligent Tutoring Systems considered the creation of a new Computer Aided Instruction system in order to validate new tutoring base and interfacing methods. The creation of a new music-keyboard skills expert and the definition of a revised version of a generic Intelligent Tutoring System structure led to the creation of a fully functional Intelligent Tutoring Systems. This was used to teach basic music-keyboard skills using the new music technology system, which was successfully tested for validation and improvements.

7.2. Resolution of Research Aims and Objectives

The specific objectives set out in the introduction are highlighted below. Beneath each of the objectives is an assessment of the work completed against the objectives.

MUSIC TECHNOLOGY SYSTEMS:

- *Collect teachers' requirements for a new music technology system.*
  A list of teachers' requirements for the new music technology system (hardware-firmware and software) was successfully collected. Each requirement was reviewed at the end of its respective chapter (chapter 3 for hardware-firmware requirements and chapter 4 for software requirements).

- *Investigate existing music technology systems and technologies.*
  Five music technology systems were found to be used in Education. More music technology systems might exist but were not found. All the music technology systems seemed to use the same types of technology as described in chapter 2. Other music networking systems were found but

not for Education. These could be investigated later, for example Mlan [Yamaha Corporation (n.d.1)] and Cobranet [Rane (n.d.)].

Existing technologies were reviewed in the fields of:

- Computer / Keyboard Aided Instruction.
- Analogue and digital audio.
- Computing and networking.
- Data communication.
- MIDI.
- Software tools, application analysis and design.
- Artificial Intelligence methods and applications.

- *Investigate and define the new music technology system design.*

  This area of work was completed successfully and the new Keyboard And Audio Network system was marketed by a collaborating company. The method to network audio was investigated and new Electronic Music Interfaces were designed. Electronic Music Interfaces could work stand alone or networked for remote control. Network configurations were also proposed to network the Electronic Music Interfaces with various control devices.

- *Investigate and define communication protocols.*

  A new communication protocol was successfully defined and based on the RS485 communication standard allowing up to 32 Electronic Music Interfaces to be networked. The new protocol was designed to control EMIs and to allow real-time propagation of MIDI messages between a portable keyboard and a control device. Each Electronic Music Interface could be reprogrammed (main program and EEPROM memory) from the control device for upgrades.

- *Investigate, define and design a control interface between the music technology system and teachers.*

  A suitable control device was found to be a computer and a software interface was successfully created. Seven prototypes were created and assessed by teachers to define the most usable control interface to control

the new music technology system. The software system was successfully designed without needing a help section. Two object oriented control structures were defined and tested in order to display an accurate status of the music technology system in real-time. The second structure was successfully implemented and allowed the creation of easy-to-use debugging tools for teachers as well as the handling of abnormal control system statuses (mobile Electronic Music Interfaces or crashes).

- *Investigate, define and design a new information system.*
  Requirements from teachers included an information system embedded within the control software for administrative purposes. A new information system was successfully created and embedded within the control interface using a database designed with teachers. The success of the new embedded information system led to the creation of a new system called Electronic Student Assessment And data Management System, based on the existing information system for any teaching field. The new information system was teaching-centred and allowed quick and efficient management of student workpieces.

- *Investigate and implement methods to improve teachers' mobility.*
  This area of work was completed successfully as two methods were defined to get teachers mobile, while using the new music technology and information systems in their department using wireless networking. The music technology system control computer was remotely controlled by the author using thin- (Win CE and windows terminal services) or thick- (Win XP Tablet PC edition) end tablet PCs. Radio headphones were successfully used to bring audio feedback to teachers. The information system was designed especially for Tablet PCs and an advanced database management system successfully allowed teachers to work with the information system at school or at home.

- *Collect feedback from teachers about the new music technology and information systems.*

  Feedback from teachers was successfully collected with various means such as surveys, observation and interviews. Feedback was used to determine if the new music technology and information systems fulfilled the roles they were designed to achieve and to resolve any remaining or new issues. Issues were determined and grouped.

- *Investigate solutions using Artificial Intelligence.*

  Each main issue was investigated separately to determine the feasibility and utility of solutions. It was found that solutions required some intelligence that could be provided by Artificial Intelligence.

  The first issue dealt with an intelligent assistant that could help perform tasks for the teacher so that the teacher could remain focused on the lesson. It was decided that a valid solution was to capture teachers' habits (for instance lesson preparation at the beginning of a class lesson or ending before the students left, etc.) in order to anticipate some of teachers' tasks. Existing agent methods were reviewed and a new habits capturer was created using a new stamping method. This was tested successfully on a separate project.

  The second issue dealt with teaching tools that could help students gain appropriate keyboard skills to follow lessons more easily in future. It was decided that a valid solution was to implement an Intelligent Tutoring System using the new music technology system to teach basic music-keyboard skills in a custom way. A new music-keyboard skills expert was created and successfully tested as well as a new interfacing method using the new KAAN system. It was decided that the tutoring method was more valuable for the research.

- *Investigate, define and design methods to deliver new knowledge to students and to assess their answers.*

  Two research plans were determined. A hybrid system was considered in a first research plan to replace a teacher after modelling the teaching. However this research plan was not studied further as a second research plan arose using existing work that could be re-arranged in the scope of the research. Eleven teaching methods for Intelligent Tutoring Systems were collected and tested. A generic Intelligent Tutoring System structure based on a 'push' teaching method {based on 'Cognitive Load' [Wilson and Cole (1996)] or 'Constructionism' theories [Cavallo *et al.* (n.d.1, n.d.2)]} was created and analysed by comparison mainly with the work from Gerlič (1998). This structure allowed appropriate customisation, knowledge delivery and assessment that could be used in the new Intelligent Tutoring System.

- *Implement a new Computer Aided Instruction system.*

  A new Computer Aided Instruction system was successfully created to teach basic music-keyboard skills using the new expert and interfacing method. A tutoring base was successfully designed, populated, linked to the Computer Aided Instruction system and tested with the help of teachers. The tutoring base was constructed around new teaching concepts. These concepts were defined so that music knowledge could be represented in Concepts divided in a hierarchy of subconcepts. Subconcepts could then be delivered with a range of lessons assessed with related tasks. At the end of the research, the tutoring bases contained 51 preset music lessons with corresponding tasks for an estimated teaching time of between 4 to 6 hours. A new assessment method was used to consider the correctness of answers as well as the time to respond. This provided appropriate results.

- *Investigate, define and design an Intelligent Tutoring System student model.*

An Intelligent Tutoring System student model module was successfully created and implemented on the new Computer Aided Instruction system to make that system an Intelligent Tutoring System. Teaching strategies were also researched and implemented to deliver knowledge to students in an effective way. The student model module first managed students' teaching actions as well as a knowledge grading history. The knowledge grading history was updated using a Knowledge Assessment session before the knowledge delivery to deliver the required knowledge. This was used along with a new plotting control, which provided plots of students' learning curve that could be used for administrative and motivation purposes.

Students' teaching actions were recorded but not used within the scope of the research as it was found that Knowledge Assessment session replaced the need for determining students' misconceptions. However, the data stored could be used in future to improve the Knowledge Assessment session and include misconception solving. The student model module also included student specific information to improve the interfacing and lesson delivery.

A new revised Intelligent Tutoring System structure was proposed at the end of the research and compared with the generic Intelligent Tutoring System structure proposed by Gerlič (1998). The tutoring base along with the student model module and teaching strategies were successfully designed to be working in any teaching field. Only the system interfacing method was specific to this research. Future work could include interfacing tools for teachers to create and link new concepts, lessons and tasks and to implement custom teaching strategies.

7.3. Key research successes and contribution

The new Intelligent Tutoring System based on the new music technology systems was considered successful.

A broad base of research was completed and a foundation platform from which further research and development could be conducted was created. In respect of the research work completed, the following were the tangible research successes.

MAIN SYSTEMS CREATED

- Electronic Music Interface.
- Keyboard And Audio Network and communication protocol.
- Electronic Student Assessment And data Management System.
- Computer Aided Instruction.
- Intelligent Computer Aided Instruction (also called Intelligent Tutoring System) with learning curves plotting.

NEW METHODS

- Control using ergonomic user interfaces.
- Object Oriented control for interfacing.
- Teaching-centred information management.
- Methods that allow teachers to move around.
- Capturing of habits (with a new stamping method).
- Music-keyboard skills expert.
- Interfacing with KAAN.
- ITS methodology.
- Generic tutoring.

The key contribution was the creation of a new Intelligent Tutoring System using the new music technology systems to assist music teachers. The utility broadening of the new systems, including the new generic tutoring base also assisted teachers from other fields.

7.4. Improvements to this research

7.4.1. Limitations of the research

There were several areas where the limitations in this research restricted the achievement of the set objectives and improvements could be made.

MUSIC TECHNOLGOY SYSTEMS:

The number of buses offered by network cables (3 or 4) as well as the complicated static management of these buses restricted the creation of improved network audio functions. Furthermore, the method used for grouping (with a relay on one bus to create sub-buses) restricted grouping configurations.

The music technology system was set in a special mode to propagate MIDI messages to record into MIDI files. This special mode prevented any other settings from being processed. Moreover, keyboard key presses scanning consumed too much bandwidth and could not be used efficiently in the background.

INTELLIGENT TUTORING SYSTEMS:

Teaching to multiple students at the same time was not possible:

- The tutoring database was an MS Access database, which was not designed for multiple users, however an intermediate server tier could be created to overcome the problem.

- The speech engine did not offer any options to set the audio output to send speech to different sound cards that could be remixed before being sent to the music technology system. However only three buses were available, meaning that either only three students could be managed in the same time, or a dynamic bus multiplexing system (to maximise the bandwidth by using unused buses for other students) had to be created for more students.

- The speech engine and VB6 did not handle multithreading, which will be required to teach to several students in the same time. An expensive speech engine could have been purchased and VB.NET could have been used to overcome the problem. Another solution could be to distribute the functionality to the students' station, either by modifying the Electronic Music Interface or by connecting a PC to them.

7.4.2. Requirements of enabling technologies

The music technology systems distributed audio processing to each Electronic Music Interfaces. The result was excellent and it appeared that to obtain even better results, some control features would need to be distributed.

MUSIC TECHNOLOGY SYSTEMS:

The restrictions set out in 7.4.1 (page 231) would either require the use of:

- New cables and connectors to increase the number of buses.
- Digital technology.

In both cases, the design of the Electronic Music Interfaces would need to be reconsidered as well as the protocol. However, as discussed, a network configuration using the current music technology systems allowed the inter-connection of the music technology system with a PC system. In that case, all MIDI features, as well as playback and recording features, could be distributed to the local PCs using client applications controlled by the main control software using DCOM or ASP.Net technologies.

INTELLIGENT TUTORING SYSTEMS:

The tutoring base could be re-developed into a SQL server to allow for multi-users. Data could come from a central Intelligent Tutoring System server teaching to many students at the same time using multithreading technology offered by VB.NET. However, the complexity of multithread programming and the number of

maximum threads offered by VB.Net could be restricting. Furthermore, a new speech engine server would be required to speak to multiple students in the same time. As the number of soundcards in a computer and the number of buses offered by the music technology system are finite and limited, this solution may not be viable.

A distributed solution may be the most appropriate. Using local PCs to generate the speech and to interface with the students' keyboard, neither multithreading, nor expensive speech engine, nor any system bus is needed. Furthermore, either DCOM or ASP.Net technologies could be used to control the local applications. ASP.Net offers a particular advantage as thinner clients are required and the main application could reside on a web server and be easily linked to the SQL server.

7.5. Suggestions for future work

This research has identified many areas where further work could be undertaken. Given the discussion presented in 7.4.2 (page 232), and throughout the dissertation, the following are areas where a significant contribution could be made:

- Have the Intelligent Tutoring System handle multiple students in the same time (locally and globally on the Internet) for:
    - Single student learning.
    - Collaborative learning.
- Have teachers building their own courseware to implement in the new Intelligent Tutoring System, using Intelligent Authoring Systems [Bierman (1988)]. Observe the change of the teacher's role in the classroom.

7.6. Thesis Conclusion

Research was undertaken in the area of music education to create new music technology systems and methods to help music teachers teach music. The research resulted in the design of the new music technology systems, which are now available as marketed products. The research also resulted in the creation of new Intelligent Tutoring System and methods to teach basic music-keyboard skills using the new music technology and information systems.

The objectives of the research were broadly achieved. Problems were encountered because of the use of limited technologies and these were explained.

The market for the new music technology systems and eventually the new Intelligent Tutoring System is expanding. The new intelligent systems bring essential teaching features for music teachers and especially help the use of portable keyboards effectively along with other instruments. Their flexibility, ease of installation and maintenance as well as reliability offer them a clear, competitive advantage over existing provision.

The new music technology and information systems were designed in collaboration with Counterpoint MTC Ltd, who now market the new systems (eleven systems sold at the time of writing).

If improvements were to be made on the new Intelligent Tutoring System, allowing for multi-students, there is little doubt the Intelligent Tutoring System could be marketed in future as a plug-in for or an embedded tool within the new music technology systems when the intelligent system is multi-users.

CHAPTER 8. APPENDICES

8.1. CAI features

Tests involving keyboard interfacing

MUSIC lessons:

- Note reading: display 4 notes on staff. Student has to play each note on the keyboard (different clefs available)
- Key signatures against major/minor keys
- Play 8 notes of a determined scale

AURALIA: Four fields with subjects:

- Intervals and scales
  - Interval comparison: play and answer with a keypress
  - Interval recognition: play interval
  - Interval singing: NO
  - Scales: play the choices and answer or play scale
  - Scale singing: NO
  - Advanced scales: play
  - Advanced scales singing: NO
- Chords
  - Cadences: play and answer questions
  - Chord recognition: play and play chord
  - Chord singing: NO
  - Chord progression: play
  - Advanced progression: play
  - Cluster chords: 1 note then another one
  - Jazz chords: play and answer question
  - Jazz chords singing: NO
  - Jazz chords progression: play
- Rhythm
  - Meter recognition: play drum and answer question
  - Rhythm dictation: NO
  - Rhythm elements: play and answer question
  - Rhythm elements dictation: NO
  - Rhythm imitation: play and tap (with metronome)
  - Rhythm styles: play and answer question
- Pitch and melody
  - Counterpoint singing: NO
  - Melodic dictation: NO
  - Note recognition: NO
  - Tapping: play and answer question

MUSITION

- Music reading
  - Advanced clefs: show and answer question
  - Chord recognition: show and answer question
  - Meter recognition: show and answer question
  - Note reading: show and answer question (play)
  - Rhythm notation: show and answer question
  - Rhythm tapping: show and play/tap
- Terms and symbols
  - Chord symbols: show and answer question (play)
  - concepts: show and answer question (play)
  - symbols: show and answer question (play)
  - terms: show and answer question (play)
- Key centres
  - intervals: show and answer question (play)
  - chords / scale relations: show and answer question (play)
  - jazz chords: show and answer question (play)
  - key signatures: show and answer question (play)
  - modulation: show and answer question (play)
  - scales: show and answer question (play)
  - scale degree: show and answer question (play)
  - scale home keys: show and answer question (play)
- Instruments
  - Drum sticks: show and tap
  - Drum styles: show and tap
  - Guitar symbols: show and answer question
  - Instrument keys: show and answer question
  - Instrument range: show and answer question
  - Instrument recognition: show and answer question
  - Transposition: show and answer question

8.2. Hardware and Firmware

8.2.1. Power block Design

The power bloc design consisted of:

A) Heat sink dimensioning

B) Power resistors dimensioning

A) HEAT SINK DIMENSIONING

A heat sink dissipated heat from the 3 voltage regulators (2*LM2940T10 + 1*7905). The calculation was performed with the worst case for 1 of them (localised power dissipation) using Equation 8-1: Heat Sink Calculation.

Equation: $$\Theta_{S\text{-}A} = \frac{Tj - Ta}{Pd} - (\Theta_{J\text{-}C} + \Theta_{C\text{-}S})$$

Equation 8-1: Heat Sink Calculation

With:

- Tj: Max Junction Temperature
- Ta: Max Ambient Temperature
- Pd : Max Power Dissipation
- $\Theta_{J\text{-}C}$: Junction to case (=3 K/W)
- $\Theta_{C\text{-}S}$: Case to heat Sink (=2K/W)

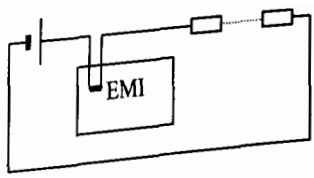
Pd = (Vin max – Vout) * Imax
= (20-10) * 1 = 10W

Tj = 125°C
Ta = 40°C

$\therefore \Theta_{S\text{-}A}$ = 85/10 – 5 = 3.5 K/W chosen 3.7 K/W

B) POWER RESISTORS DIMENSIONING

The EMI had two Printed Circuit Board (PCB) connectors to allow the daisy chaining of the power net. A reinforced power track between the two connectors has been designed to reduce the resistance of this track in order to reduce heat dissipation. The track has been tested on one EMI with low resistance power resistors (dimensioned using Equation 8-2: Equivalent Power Resistor Calculation) and a high capacity power supply.

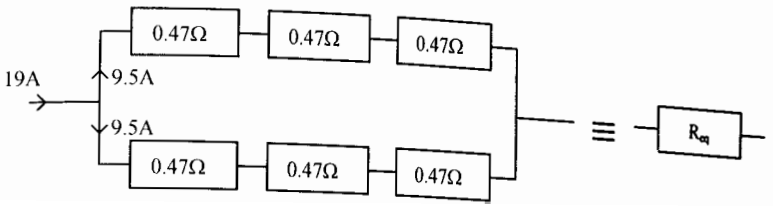


Power Supply: 13.5V (35A)
I_{max} = 19A (determined by connectors)

$\therefore P = U I = 13.5 * 19 = 256.5W = R I^2$
$\therefore R = 256.5 / 19^2 = 0.71\Omega$

Equation 8-2: Equivalent Power Resistor Calculation

The resistance required to get 19A was 0.71Ω and was obtained with:



$R_{eq} = 1/(2*1/(3*0.47)] = 0.705\Omega$

Power dissipated per resistor = 0.705 * 9.5 = 42W
⇒ 100W - 0.47Ω resistors

Leads used: $2.5mm^2$ (25A)

8.2.2. Audio and Microcontroller Block Design

The audio and microcontroller bloc design consisted of:

A) Digital potentiometer corrected values Problem 1
B) Digital potentiometer corrected values Problem 2
C) Crystal Calculations
D) Terminator Resistor Calculations

A) DIGITAL POTENTIOMETER CORRECTED VALUES PROBLEM 1

Sound followed a logarithmic law, thus a logarithmic digital potentiometer was required to measure the volume amplification of the headphones output. The problem was that only linear potentiometers were available. Thus, a conversion table was required for the microcontroller to send the correct information to control the digital potentiometers.

The selected linear digital potentiometer had a resolution R_{lin} of 256. The aim was to calculate the value to send to it from a required log value. This value depended upon the resolution R_{log} of the virtual log potentiometer (influencing the number of levels considered for the volume control).

It was assumed that the law followed the equation (Equation 8-3):

$$Y_{lin} = A.10^{B.X_{log}} - C$$

Equation 8-3: Assumed Logarithmic law

with:

Y_{lin}: Linear value to send to the potentiometer
X_{log}: Log value required to change the volume

The limit conditions were assumed to be:

1: $X_{log} = 0 \Rightarrow Y_{lin} = 0$
2: $X_{log} = R_{lin} - 1 \Rightarrow Y_{lin} = R_{log} - 1$

As there were 3 unknowns, another equation was needed. As the law was logarithmical, the gradient for the lower values could be supposed to be low, thus:

3: Xlog = 1 ⇒ Ylin = 1

From 1.: $A = C$

From 2.: $R_{lin} - 1 = A * 10^{B*(Rlog-1)} - A$

$$\therefore A = \frac{Rlin - 1}{10^{B*(Rlog-1)} - 1}$$

From 3.: $B = \log\left(\frac{A+1}{A}\right)$

$$\therefore A = \frac{Rlin - 1}{10^{(Rlog-1)*\log\left(\frac{A+1}{A}\right)} - 1} \Rightarrow \left(\frac{A+1}{A}\right)^{Rlog-1} = \frac{A+(Rlin-1)}{A}$$

Equation 8-4: Log / Lin Conversion

After solving Equation 8-4 by a drilling method, it was found:

For R_{lin}=256 and R_{log} = 64: A=C=25.952381456 B=0.0164199286

For R_{lin} =256 and R_{log} = 32: A=C=8.5522303506 B=0.04802539697

Required Log	Calculated Lin
0	0
1	1
2	2
3	3
4	4
5	6
6	8
7	9
8	12
9	14
10	17
11	20
12	23
13	27
14	31
15	36
16	41
17	47
18	54
19	61
20	69
21	78
22	88
23	100
24	112
25	127
26	143
27	160
28	180
29	202
30	227
31	255

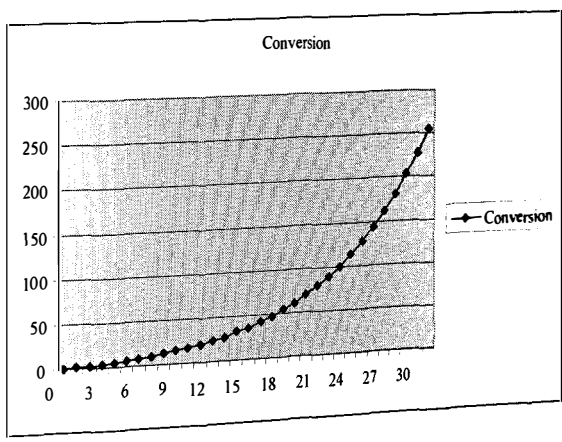


Figure 8-1: Example with R_{log}=32

B) DIGITAL POTENTIOMETER CORRECTED VALUES PROBLEM 2

The line driver was designed with headphone op amps. The signal was copied and this was inverted so that noise picked up on the bus could be cancelled at the reception (balanced audio driven on 1 twisted pair). The switchability (to have only driving EMIs attached to a bus) was designed using a MAX366 chip (signal-line circuit protector), for which power was controlled with mosfets (low gate powered transistors).

The receiver was also designed with headphone op amps. The signals were added in order to cancel any carried noise that would have been received by both wires of the twisted pair. A problem occurred in the testing stage as the received signal decreased with the number of EMIs on the bus. This occurred as it was found that the MAX366 chip could be considered to operate as a resistor. Therefore, Ohm's law had to be applied for each EMI added on the bus and the corrected 'loading' of a bus had to be calculated to compensate the effect of the Max366. As this loading correction had to be performed anyway, it was decided to control the volume of the received signal coming from a bus as well.

The same digital potentiometers as for the volume control were used to control the bus input volume/loading. A new table was calculated to account for both number of EMIs attached to the bus (or 'bus loading') and a 'bus volume'. Table 8-1 shows the value the microcontroller has to send to a receiver's digital potentiometer to set the right volume for a determined bus load (set by the number of EMIs attached to a bus). The values are calculated to follow a logarithmic law using a linear potentiometer.

Volumes have to be read in columns (0-15) / Loads have to be read in rows (1-16).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48
4	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
5	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96
6	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128
7	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144
8	11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176
9	13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208
10	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	**255**
11	18	36	54	72	90	108	126	144	162	180	198	216	234	252	**255**	**255**
12	21	42	63	84	105	126	147	168	189	210	231	252	**255**	**255**	**255**	**255**
13	24	48	72	96	120	144	168	192	216	240	**255**	**255**	**255**	**255**	**255**	**255**
14	27	54	81	108	135	162	189	216	243	**255**	**255**	**255**	**255**	**255**	**255**	**255**
15	31	62	93	124	155	186	217	248	**255**	**255**	**255**	**255**	**255**	**255**	**255**	**255**

y = A (10^BX - 1) with R1=12K gain of 1.01 if value = 31
A = 10.320840396
B = 0.04016363056

Table 8-1: Bus loading table

C) CRYSTAL CALCULATIONS

Calculations were performed to determine the best crystal to use to clock the microcontroller as the microcontroller had to handle three different protocols running at different baud rates:

- MIDI at 31250 bauds ±5%
- RS232 at 38400 bauds ±5%
- RS485 at around 38400 bauds

The baud rate was a function of the frequency of the crystal and a microcontroller parameter (integer) called UBR. The relationship was (Equation 8-5):

$$\text{Baud rate}_{(\text{baud})} = \frac{Frequency}{16(UBR+1)}$$

Equation 8-5: Crystal Frequency Calculation

As UBR was an integer, there could be an error between the calculated and the required Baud rates. RS232 and MIDI were used to communicate between EMIs and a computer or a keyboard. The error for these standards had to be minimal and within the tolerances to have correct communication. RS485 was used to communicate between EMIs, thus any error was acceptable, as every EMI would get the same error. Therefore the crystal frequency had to be calculated with the use of MIDI and RS232 simultaneously. Furthermore, the crystal frequency had to be below 8 MHz (microcontroller specification).

Crystal Frequency (MHz)	MIDI		RS232	
	UBR	Error (%)	UBR	Error (%)
1.8432	3	7.8	3	7.8
2	3	0	3	0
2.4576	4	1.7	4	1.7
3.2768	6	6.4	6	6.4
3.579545	6	2.3	6	2.3
3.6864	6	5.3	6	5.3
4	7	0	7	0
4.194304	7	4.9	7	4.9
4.433619	8	1.5	8	1.5
4.608	8	2.4	8	2.4
4.9152	9	1.7	9	1.7
5	9	0	9	0
5.0688	9	1.4	9	1.4
5.24288	10	4.7	10	4.7
6	11	0	11	0
6.144	11	2.4	11	2.4
6.4	12	1.5	9	4.2
6.5536	12	.8	10	3
7.3728	14	1.7	11	0
7.86432	15	1.7	12	1.5
8	**15**	**0**	**12**	**.2**

Table 8-2: Crystal Frequency Errors for MIDI and RS232

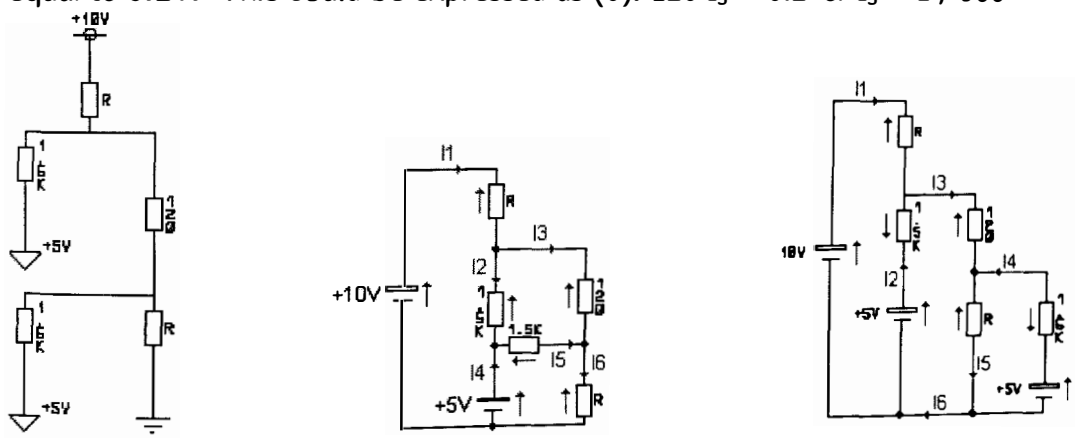
D) TERMINATOR RESISTOR CALCULATIONS

Communication with PC (RS232): communication problems occurred with the PC due to digital level latency:

Figure 8-2: RS232 Signals causing communication problems

The network state had to be defined for the instance when nobody talked to prevent level stabilisation latency and therefore communication problems. This needed to be set in only one box, with pull up and pull down resistors. The value of the pull up and down resistors of the network terminator had to be calculated accurately as some EMI would not communicate properly:

The terminator had to be calculated so that the voltage at the 120R resistor was equal to 0.2V. This could be expressed as (0): $120\ I_3 = 0.2$ or $I_3 = 1 / 600$



Figures 8-3: Network termination Circuitry Representations

From the last representation, the relationships between the currents (in red) could be written:

(1) $I3 = I1 + I2$

$I5 = I3 + I4 \ \therefore$ (2) $I5 = I1 + I2 + I4$

$I5 = I4 + I6 \ \therefore$ (3) $I4 = I5 + I6$

$I1 = I6 - I2 \ \therefore$ (4) $I6 = I1 + I2$

$\therefore$ (5) $I4 = I5 - I1 - I2$

The relationships between the voltages (in green) could be written:

(A) $10 = 5 - 1.5K\ I2 + R\ I1$

(B) $5 = 1.5K\ I2 + 120\ I3 + R\ I5$

(C) $5 = 1.5K\ I4 + R\ I5$

From (A) and (1): $5 = R\ (I3 - I2) - 1.5K\ I2 \ \therefore$ (i) $I2 = \dfrac{RI3 - 5}{R + 1.5K}$

From (C) and (2): $5 = 1.5K\,(I5 - I3) + R\,I5$ $\therefore$ (ii) $I5 = \frac{5+1.5KI3}{R+1.5K}$

From (B), (i) and (ii): $5 = 1.5K\left(\frac{RI3-5}{R+1.5K}\right) + 120\,I3 + R\left(\frac{5+1.5KI3}{R+1.5K}\right)$

Which gives: $15K = I3\,(3120\,R + 180K)$

From (0): $I3 = 1/600$ $\therefore R = \frac{15K*600-180K}{3120} \approx 2827\Omega$

Chosen Value: R = 2.7K (existing commercial value)

8.2.3. Metal enclosure

Initially, the idea was to have each EMI fixed on top of the students' desks. The fixed cables were connected on the bottom of the EMI and the input and output stereo sockets were available on the front panel of the EMI, such as shown in Figure 8-4.

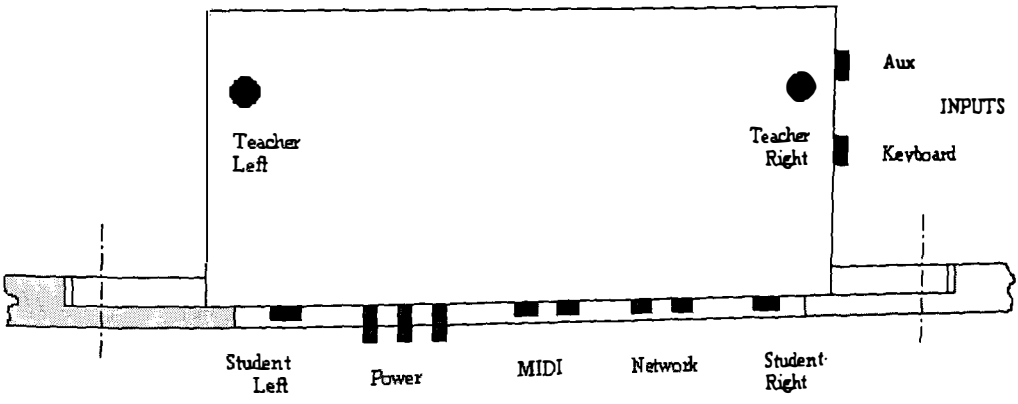


Figure 8-4: First Idea

This design was rejected because:

- It did not look professional because of its shape and assembly
- The space taken on the desktop surface was too large
- A hole had to be made in the desktop and it could only be installed on desks
- Multiple connector directions and connected to the box separately involved the design of multiple small PCBs connected together with ribbon cables

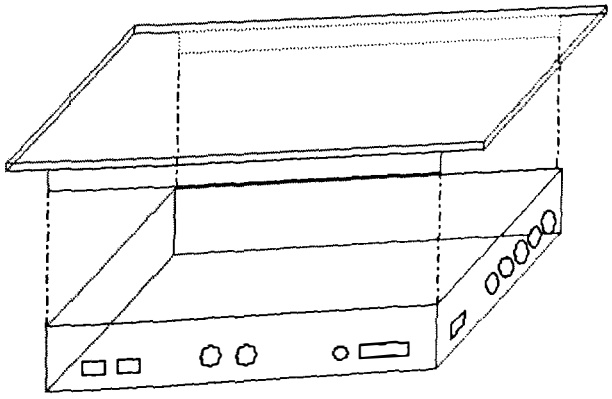


Figure 8-5: Second Idea

A second idea emphasised the use of simple shaped, easily manufactured bases and covers as shown in **Figure 8-5**. The PCB would be screwed onto the base and the cover would close the box and eventually be used as brackets to fix the box anywhere. Also protruding connectors (RS232 serial port, and audio sockets) had to be gathered on the same side and the other connectors set on a perpendicular side to ease the installation of the PCB on the base.

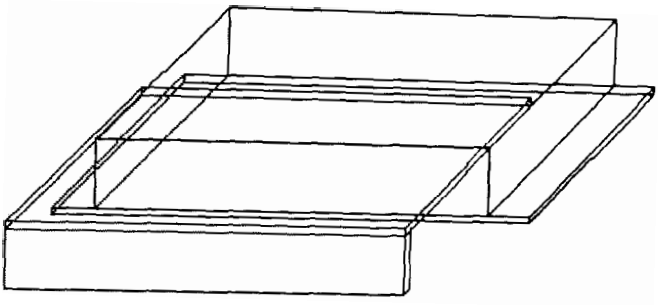


Figure 8-6: With cable hider

The EMI could be installed under a desk or on a wall. To secure the power and network connectors from the students, a cable hider could be designed to fit onto the base or the cover as shown in Figure 8-6: With cable hider.

The final design was based on the second idea and consisted of a base to which the main PCB was screwed. Holes allowed the audio and RS232 (serial port) sockets and a debug tricolour LED to protrude. Threaded bushes were also added to screw brackets on the base to fix the EMI on a wall or under a desk. The metal enclosure also comprised a cover screwed on the base. Slots allowed the power (2 network and 1 out), MIDI (in and out) and network (in and out) connectors to be kept plugged in when the cover needed to be removed for maintenance.

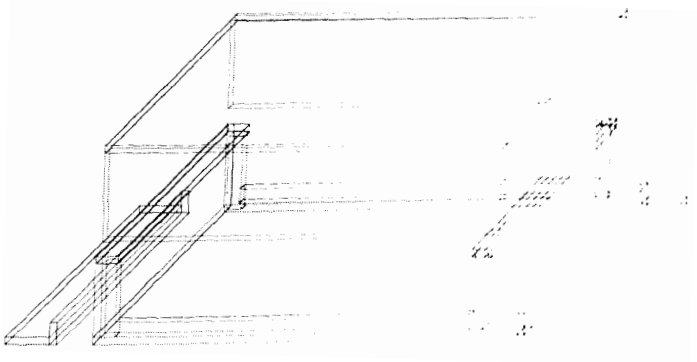


Figure 8-7: Final design

All parts were made out of punched stainless steel and painted in black. A print screen was designed for the cover top to show the connectors' name. The metal enclosure was grounded from the serial port for safety reasons and to reduce Electro-Magnetic Interferences. Punched holes were added on the cover for the heat sink, but small enough to prevent anything being poked into the box that could cause short circuits.

Improvements were:

- *Brackets*: A pair of bracket was preferred than four small ones, which were fixed with countersunk screws.
- *Cover*: Slots were punched instead of holes for the connectors for easier maintenance (no need of disconnecting the cables).
- *The entire enclosure* itself was designed bigger for easier insertion of the PCB.

8.2.4. Firmware: Protocol of communication

This section consisted of:

A) Commands
B) Commands Usage

A) COMMANDS

Below is the full documentation of the communication protocol commands. These commands are arranged in order of databytes to follow, the first byte being constructed as follows:

G	x	x	x	x	x	F	F

With:

FF: Bytes to follow
xxxxx: Address or Command
G: Addressable Mode

FF	Description	Notes
00	0 Byte to follow	
01	1 byte to follow	
10	2 bytes to follow	
11	3 bytes to follow	

G	Description	Notes
0	Message addressed to every EMI	First Byte called First Command Byte
1	Message addressed to only 1 EMI	First Byte called Address Byte

Table 8-3: First Byte description

PING

Code	Addressee	Command
1**AAAA**00	Defined by AAAAA	Ping

COMMAND BYTE ONLY (NO DATA BYTES)

COMMANDS 0 TO 31 (Global and single addressing modes)

Code	Description	Notes
0	Response to master?	(Locals are alwa
1	*Mute Everything (Disabled)*	*(Left,Right, Bus1,2,3,4)*
2	Mute Left and Right	
3	Un-Mute Left and Right	
4	Increment Net Address	(only valid in net setup mode)
5	*Master Ignore next Message*	(Internal use only)
6	*Reset User Keyboard Settings	Reset the internal Pan and Volume settings on Midi
7	*Exit Net Addressing Mode	(only valid in net setup mode, and only as a global msg)
8 .. 25		
26	*Read Network Address (This will reply with the address)	Note this should only be sent in global mode if there is only one box attached to the network.
27	Reset (Software Restart)	
28	Clear Network Counters	Clears ChecksumErr /Sent /Rec
29	Restore Factory Settings	
30	Receive Raw Data	Rec. C0, C3 = End Receive
31	Ignore Data	C0, C0 = C0 (char) C0, C1 = Start block C0, C2 = End block C0, C3 = End "Ignore Data"

COMMAND BYTE ONLY (NO DATA BYTES)

COMMANDS 32 TO 63 (single addressing mode)

Code	Description	Notes
32		
33	Read NetSplit Value	
34	Read Headphone Volume Value	Reads both Left and Right
35	Read Left,Right Mix Switches	Returns 2 bytes
36	Read Bus 1, Bus 2 Mix Switches	Returns 2 bytes
37	Read Bus 3, Bus 4 Mix Switches	Returns 2 bytes
38	Read Headphone Switches	Returns 1 byte
39	Read Student and Teacher Switch	Returns 2 bytes It returns what was sent, which may be illegal values
40	Read Mute Status	Returns 2 bytes, LR, Bus1-4
41	Read Bus Volume Settings	(2 Bytes) Stored in Nibbles
42	Read Bus 1 & 2 Loading Setting	(2 Bytes) Bus1, Bus2
43	Read Bus 3 & 4 Loading Setting	(2 Bytes) Bus1, Bus2
44	Read Net-Split SubGroup	0 = No Netsplit, 1-254 = Group address, 255=Address not defined yet
45	Read LED status	Led Colour (1 byte)
46	*Send All Keys Forced	This returns 3 global packets (with command 0, and 3 Data bytes) the data returned is Key Change byte. Followed by the data bytes
47	*Send Centre Keys	Incremental Report only on keys 57 to 72
48	*Send Left Keys	Incremental Report only on keys 33 to 64
49	*Send Right Keys	Incremental Report only on keys 65 to 96
50	*Send Any keys	Incremental Report only on keys 33 to 96
51	*Send Key Status	Returns the Key Status
52	*Read Ticks per Deci Second	Returns Maximum Ticks (low, Hi)
53	*Read User Status The changes flags are cleared after this statement	High nibble = Filter, (0,T,R,L) Low nibble = Changes, (0,T,R,L)
54	*Read Eeprom Byte This reads one byte from the user – use N1:21 to set read address	Byte Read from Eeprom
55	*Read MIDI Patch	2 bytes sent (left/right)
56	*Read MIDI TPFlag (Bits 0-1 = Transpose (0:off, 1:down, 2:0, 3:up), Bits 2-7 = Volume)	2 bytes sent (left/right)
57	*Read MIDI Listen Channel	Channels to listen to (1 in each nibble)
58	*Read MIDI FIFO Filter Messages	2 Bytes (System, Voice)
59	*Read MIDI FIFO Filter Channels	2 Bytes (ch 0-7,ch 8-15)
60	*Read MIDI FIFO Filter Keys	2 Bytes (low, high)
61	*Read MIDI THRU Filter Messages	2 Bytes (System, Voice)
62	*Read MIDI Transpose channels	2 Bytes (left, Right / nibble)
63	*Read MIDI Transpose Low Value	Left Channels

COMMAND BYTE ONLY (NO DATA BYTES)

COMMANDS 64 TO 95 (single addressing mode)

Code	Description	Notes
64	*Read MIDI Transpose High Value	Right Channels
65	Return Software Version	(2 Bytes) Maj, Minor
66	Check Sum Error Count	The count is reset after a read
67	Network Packets Received	(2 Bytes) low first
68	Network Packets Sent	(2 Bytes) low first
69	Resend Last Message	If no msg has been sent or counters have been cleared then sends a 'blip'
70	Buffer Overrun Status	1: U0Tx Overrun 2: U0Rx Overrun 4: U1Tx Overrun 8: U1Rx Overrun 16: 32: 64: FIFO Protected Overrun 128:FIFO Overrun
71	MIDI Buffer Overrun Count	
72	Checksum Value	2 bytes(Low – High)
73 .. 80		
81	*Read EEPROM Address	Current address of the EEPROM pointer
82	*Read MIDI Patch Range	2 bytes exactly was has been set
83	*Read Duet Mode Pan	1byte, 2 nibbles (Right | Left)
84 .. 89		
90	Dbg: Read Bus Pot 1,2 value	Actual pot value is returned (bus1,bus2)
91	Dbg: Read Bus Pot 3,4 value	Actual pot value is returned (bus3,bus4)
92 .. 249		
250	Dbg: Read Bus Pot 1,2 value	Actual pot value is returned (bus1,bus2)
251	Dbg: Read Bus Pot 3,4 value	Actual pot value is returned (bus3,bus4)
252	Debug	(internal use only)
253	R Stack Space	Amount of stack space that hasn't been used since power up (2 Bytes) low first
254	C Stack Space	
255		

COMMAND + 1 DATA BYTE

COMMANDS 0 TO 31 (Global and single addressing modes)

Code	Description	Data 1
0	Response to master?	
1	Set NetSplit (only works hen in netsplit mode see [N1:15])	Split=1 / Not split=0
2	Set Headphone Volume Left	Left Volume (0..31)
3	Set Headphone Volume Right	Right Volume (0..31)
4	Set Left Mix Switches	Switch Settings B0 (1) = Aux2 B1 (2) = Aux1 B2 (4) = Keyboard Right B3 (8) = Keyboard Left B4 (16) = Bus 4 B5 (32) = Bus 3 B6 (64) = Bus 2 B7 (128) = Bus 1
5	Set Right Mix Switches	
6	Set Bus 1 Mix Switches	
7	Set Bus 2 Mix Switches	
8	Set Bus 3 Mix Switches	
9	Set Bus 4 Mix Switches	
10	*Set Headphone Switches (Disabled)*	*HP Switch Settings*
11	Set Student Headphone Switch	0 = Solo, else: Split(1)/Duet(2)
12	Set Teacher Headphone Switch	0=Off, 1=Left, 2=Right 3=Stereo, 4=Mix (else Mix)
13	Set LED Colour	0=Off, 1=Green, 2=Red, 3=Orange
14	Clear Bus (set loading=0, output=off)	Bus (0.3)
15	Bus 2 Split Mode	0 = Turn Split mode off 1 = Turn Split mode on
16	*MIDI Send 1 Char If used in conjunction with MIDI_Send_1_Plus_Char then char is the data else char is the status and must be between (80→FF) if extra bytes are needed to complete the midi message 0's are used	Char
17	*MIDI Send 1 Plus Char If this is set it will be prefixed to the next MIDI_Send_1_Char or MIDI_Send_2_Char, if a value < 80 hex is sent then it is cancelled	Char (80→FF)
18	*MIDI Running Status This sets the running status status byte	Status Byte (80→FF)
19	Set User Filter This controls if a left User, Right User, or Teacher can control the box from the centre keys	High nibble = Filter, Low nibble = Changes
20	Set Keyboard Splitting. This command sets all the switches and sends appropriate MIDI data to the keyboard	0 = Solo, 1 = Split, 2=Duet
21	Set EEPROM Start Address	Address (0..255)
22	EEPROM Write One Byte This auto increments the Address pointer. (Address pointer goes from 00 to FF then back to 00 !!)	Byte to store
23	*MIDI Listen Channel this is used to filter channels for MIDI transpose/splitting and also for the keyboard status	Channels to listen to (1 in each nibble)
24	*Set Duet Mode pan	1byte, 2 nibbles (Right | Left)
25 .. 31		

COMMAND + 1 DATA BYTE

COMMANDS 32 TO 255 (single addressing mode)

Code	Description	Data 1
32		
33	Specify Net-Split Sub-Group	Group (1..255)
34	Attach / Detach from NetSplit Subgroup	1=Attach 0=Detach
35	*Send Keys (incremental)	Mask (Keys) 1 (33-40) | 2 (41-48) | 4 (49-56) | 8 (57-64) | 16 (65-72) | 32 (73-80) | 64 (81-88) | 128 (89-96)
36 .. 255		

COMMAND + 2 DATA BYTES

COMMANDS 0 TO 31 (Global and single addressing modes)

Code	Description	Data 1	Data 2
0	Response Message to Master		
1	Set Headphone Volume	Left	Right
2	Set Headphone Switches	Student (0=solo) Else – Split(1)/Duet(2)	Teacher (0=Off, 1=Left, 2=Right 3=Stereo, 4=Mix)
3	Mute *(warning this should not be used to mute the bus's as it only updates the local bus loading parameter – all other box's will not be updated and therefore their volume will be wrong! Use [N1:33,34])*	0=Left, 1=Right, (2=Bus1, 3=Bus2) (4=Bus3, 5=Bus4)	0 = No Mute, 1 = Mute <all other values> = Mute
4	Bus Volume	Bus (0..3)	Vol (0..7)
5	Bus Loading	Bus (0..3)	Load (0..255) (only 0..16 are processed)
6	NetSplit Sub-Group Loading (this will only change the load setting on Bus2 if the group matches the box's group)	Group (1..255)	Load (0..255) (only 0..16 are processed)
7	*MIDI FIFO filter messages (the parameter values shown can be or'd together). This is a low level parameter, (Real Time is >= 0xF8, System is 0xF1 → 0xF6, Sys Ex is 0xF7, 0xF0).	System (1 = Sys Ex | 2 = System Common | 4 = Real Time)	Voice (1=Note off | 2=Note on | 4=Poly Aftertouch | 8=Control Change | 16=Program Change | 32=Pressure | 64=Pitch Bend | 128=Chan Mode)
8	*MIDI FIFO filter Chan (setting the bit to 1 will filter out the appropriate channel)	Channel 0-7 0(LSB) → 7(MSB)	Channel 8-15 8(LSB) → 15(MSB)
9	*MIDI FIFO filter Keys	Low Keys (will filter all keys below this value)	High Keys (will filter all keys above this value)
10	*MIDI Thru filter Messages (the parameter values shown can be or'd together). This is a low level parameter,	System (1 = Sys Ex | 2 = System Common | 4 = Real Time | 128=Auto Split +Transpose)	Voice (1=Note off | 2=Note on | 4=Poly Aftertouch | 8=Control Change | 16=Program Change | 32=Pressure | 64=Pitch Bend | 128=Chan Mode)
11	*Split Transpose Channel Definition	Left Channels (one in each nibble)	Right Channels (one in each nibble)
12	*Split Transpose Low Values	First Chan Transpose (Specified in semi-tones)	Second Chan Transpose (Specified in semi-tones)

Code	Description	Data 1	Data 2
13	*Split Transpose Hi Values	First Chan Transpose (Specified in semi-tones)	Second Chan Transpose (Specified in semi-tones)
14	*MIDI Send 2 Char If used in conjunction with MIDI_Send_1_Plus_Char then char 1 & 2 are the data else char1 is the status byte and must be between (80→FF) if an extra byte is needed to complete the midi message 0 is used	Char1	Char2
15	*MIDI Send 2 Running Status If used in conjunction with MIDI_Running_Status then char 1 & 2 are the data If the running status has not been defined then the command does nothing extra byte is needed to complete the midi message 0 is used	Char1	Char2
16	Chain Send Key Stat (if Box Number +1 = myNetAddress then this command responds by generating the same global message with its Box number (which will cause the next box, myNetAddress +1 to respond). It also sends the Key status data as the parameter. – The master box sends all this data to the host computer which simply listens to all the boxes reports then address them all seperatly once they have finished).	Box Number - 1	Ignored (but contains the status for the relevant box number) High: Filter, Low: changes xTRL
17	Chain User Stat (if Box Number +1 = myNetAddress then this command responds by generating the same global message with its Box number (which will cause the next box, myNetAddress +1 to respond). It also sends the User status data as the parameter. – The master box sends all this data to the host computer which simply listens to all the boxes reports then can interrogate them all seperatly once they have finished. The User status is reset after this command)	Box Number - 1	Ignored (but contains the status for the relevant box number)
18	EEPROM Write Two Bytes This auto increments the Address pointer. (Address pointer goes from 00 to FF then back to 00 !!)	First Byte to store	Second Byte to store
19	EEPROM Send Block To MIDI This will send a block of data Directly to the MIDI Port	Block Pointer Address The address of a location in the Eeprom that points to the data to be transmitted	UnLock Byte = (0xC0)
20	Set MIDI Split Patch	Bit 1: left, Bit 2: Right	Patch [0..127]
21	Set MIDI Split Volume	Bit 1: left, Bit 2: Right	Volume [0..127] the keyboard is set to this value but the EMI stores the nearest 4 index
22	Set MIDI Patch Range if bit 7 = 0 then the range is set to both channels (layers) to the selected range [lower..higher]. If the 2 given values are the same, it will restrict to only 1 sound. If bit 7 = 1 then the given values are respectively the restricted patch for the 1st and 2nd layers. An undefined mode occurs if bits 7 differ.	Lower patch(0-127)(bit7=0) /Channel (layer) 1 (0-127) (bit7=1)	Higher patch(0-127)(bit7=0) /Channel (layer) 2 (0-127) (bit7=1)
23	Slave Ignore ReProgram	Lock Byte 1 = (&HAA=170)	Lock Byte 2 = (&C0=192)
24 .. 31			

COMMAND + 2 DATA BYTES

COMMANDS 32 TO 255 (single addressing mode)

Code	Description	Data 1	Data 2
32			
33	Attach/Detach from Bus (will return a message, either a blip or a global [N2:5])	Bus (0..3)	0 = Detach 1 = Attach
34	*Set Net Address (used to reallocate a box's address)	Lock Byte = (0xC0)	Address = 0..31 Or 255 = Undefined
35 .. 255			

COMMAND + 3 DATA BYTES

COMMANDS 0 TO 31 (global addressing mode)

Code	*Description*	Data 1	Data 2	Data 3
0	Response Message to Master			
1	*Configure Network Command (each box enters a special wait state and will not respond to any other network messages apart from N0:4)	Lock Byte 1 (0xAA=170)	Lock Byte 2 (0xC0=192)	Lock Byte 3 (0xAA=170)
2	*Set Single Unallocated Net Address (Should only be used if there is only slave 1 box on the network)	Lock Byte 1 (0xC0=192)	Lock Byte 2 (0xAA=170)	New address (0..31)
3	Save Status to Non-Volatile-RAM	-	-	-
4	Restore Saved Status (only works if status has been previously saved)	-	-	-
5	Save Split-Net Status	-	-	-
6	Restore Split-Net Status	1=Restore Volume	1=Restore Switches	1=Restore Loading
7	MIDI Raw Data	Raw 1	Raw 2	Raw 3
8	EEPROM Write Three Bytes This auto increments the Address pointer. (Address pointer goes from 00 to FF then back to 00 !!)	First Byte to store	Second Byte	Third Byte
9	Re-Program Master	Lock Byte 1 (0xAA=170)	Lock Byte 2 (0xC0=192)	Lock Byte 3 (0x5A=90)
10	Master Pass - Through	Lock Byte 1 (0xC0=192)	Lock Byte 2 (0xAA=170)	Lock Byte 3 (0x5A=90)
11	Re-Program Slaves	Lock Byte 1 (0xAA=170)	Lock Byte 2 (0xC0=192)	Lock Byte 3 (0x55=85)
12 .. 31				

B) COMMANDS USAGE

Below are some tips on how to use the above protocol commands. The terms written in brackets such as [N3:01], refer to commands. The first number indicates the number of databytes to follow, the second one indicates the command number in that category.

CONFIGURE THE NETWORK ADDRESSES

Each EMI was assigned a unique address to respond to local commands. This address could be set in many ways:

The global [N3:01] Configure_Network_Command needed to be sent, with the appropriate unlock bytes (AA, C0, AA hex) (each box then entered a special wait state and would not respond to any other network messages). An address counter was set to 1. If the box received [N0:4] Increment_Net_Address, the address counter was incremented by 1. If the box received a MIDI message (from the keyboard) then the Address Counter became the net address for the box. The box then transmitted a global [N0:4] Increment_Net_Address. It then exited and resumed normal operation.

A box's network address could be changed by directly addressing it with: [N2:34] Set_Net_Address (C0, NewAddress hex). If a SINGLE new box (non addressed) was added to an existing network it could be given a network address using [N3:2] Set_Single_Unalocated_net_address (C0, AA, Address hex).

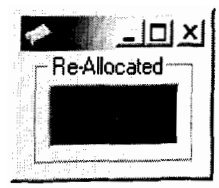


Figure 8-8: New unallocated address

This utility allowed an address to be set to any unallocated EMI in the network. It first pinged all EMIs from 1 to 31 and then allocated the last address to where no EMI pinged back. It then pinged back this address to check if the new address was set correctly. The background turned red if not.

TALK ON THE AUDIO NETWORK

- **On a standard (non split Bus):**

The global [N1:14] Clear_Bus could be sent in order to mute all outputs to the bus, and reset the load to Zero.
[N2:33] Attach_Detach_from_Bus was sent individually to each box that had to send audio to the bus (the command responded with a global [N2:5] Bus_Loading command stating the new loading factor for the bus). If the connection did not change (ie Attach command was sent and the box was already attached) then a

'Blip' was sent back to the master. The global [N1:14] Clear_Bus was sent again when finished.

- **Use of the Split Bus (Bus2):**

[N1:15] Bus2_Split_mode(true) was sent to mute all outputs to bus2, and reset the load to zero, to reset the netsplit relay, and to set the box in group mode – 'All belongs to Group 1'. Appropriate individual commands to split the net [N1:1]Net_Split(true) then needed to be sent. Each box had to be informed which sub-group it belonged to with [N1:33] Specify_Net_Split_Sub_Group (1..255). Each box that would send audio to the Split Net needed to receive [N1:34] Attach_Detach_from_Sub_Group individually (the command responded with a global [N2:6] Sub_Group_Loading command stating the new loading factor for the bus). If the connection did not change (ie Attach command was sent and the box was already attached) then a 'Blip' was sent back to the master. When finished, [N1:15] Bus2_Split_mode(false) was sent to mute all outputs to bus2, and to reset the load to zero, to reset the Netsplit relay, and to set the box in non-grouped mode.

Note: to read if a box was physically attached to a bus [N0:40] Read Mute status could be used. (Also [N0:44] was useful for determining the net Sub-group for the net-split option)

CONTROLLING THE MIDI

- **Sending a Basic Message:**

Basic messages could be sent using the 'MIDI Send' functions. Three functions could be used as follows:

'MIDI_Send_1_Char' :- This would send one character on MIDI (note this had to be a value higher than 80 hex) the single value would be used as the status byte in the midi message, and if the message required more bytes then 0's were appended.
'MIDI_Send_2_Char' :- This would send two chars on MIDI (note, first data parameter had to be a value higher than 80 hex) the first data value would be used as the status byte in the midi message, and the second as regular data. If the message required more bytes then a 0 was used. (This defaulting to 0 could be useful for generating note off messages).

'MIDI_Send_1_Plus_Char' with 'MIDI_Send_1_Char' or 'MIDI_Send_2_Char':- This allowed an extra byte to be sent with the 'MIDI_Send_1_Char' or 'MIDI_Send_2_Char'. The byte specified with 'MIDI_Send_1_Plus_Char' was used as the status byte and had to be greater than 80 hex. If a lower value was specified then the function was cancelled. Once the status byte was defined, it was used with the next 'MIDI_Send_1_Char' or 'MIDI_Send_2_Char' routine, then it was cancelled. The data supplied in the following function was used for the rest of the midi message, and if there was not enough data supplied then 0 was used.

- **Sending a message using a Running Status commands:**

'MIDI_Running_Status' message was used to set up the running status byte. The 'MIDI_Send_2_Running_Status' command was then used to send the message. This command could be called repeatedly using the same running status byte. The running status commands worked independently of the other send commands and could be interspersed amongst them. Note the controller did not actually use the MIDI running status when communicating with the keyboard, but actually sent the status byte every time.

This utility was used to test the sending of MIDI messages to a real keyboard.

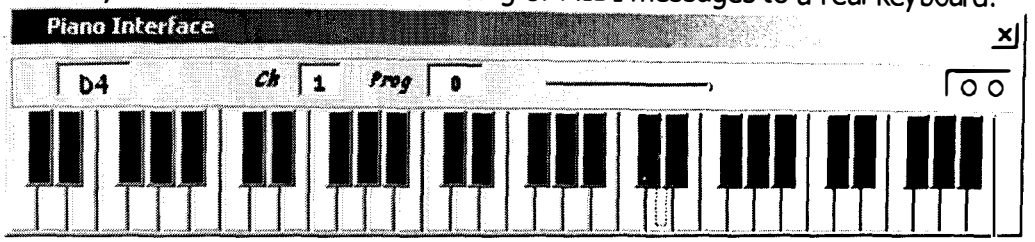


Figure 8-9: Piano interface

- **Sending and Receiving Raw Midi Data:**

The 'MIDI_Raw_Data' message was used to put the EMI into a Receive / Transmit Midi Raw Data mode. When in this mode, the protocol changed and the EMI would only receive data which could be passed onto the keyboard until C0, C3 hex were received. The 3 bytes of data that were the parameters of this instruction were used as the first three bytes of raw data and comply with the new protocol described below. However, these three bytes must not cancel the command i.e. they must not contain an end of transfer (C0, C3 hex). The protocol was as follows after the sending of the 'Receive_MIDI_Raw_Data' message:

The EMI was initially in 'Receive address' mode. It stayed in this mode until a start block was received (C0, C1 hex). In between, it could be set with the flags shown in Table 8-4: MIDI raw Mode flags.

0000 0000	0	own specific address to be received to indicate if the Data stream was to be passed on to the MIDI out. [32]
0001 1111	31	
0010 0000	32	32 + cc – (NOTE: cc is between 0 and 47) - cccccc was the number of F8's to skip between each F8 timestamp. [48]
0100 1111	79	
0101 0000	80	Unused [16]
0101 1111	95	
0110 0000	96	011a aaaa – Take control of bus (device to take control = aaaaa). The device would confirm control was taken by immediately issuing a start block command (C0 C1 hex). At the end of its allocated time period the box would send an end block (C0,C2 hex). Whilst it had control any midi data received in the FIFO buffer would be forwarded to the network. [32]
0111 1111	127	
1000 0000	128	10tt tttt – Set the time the Take control mode would take control for (0 = ½ second(default), or tttttt seconds. The timer value was only reset to 0 when the 'MIDI_Raw_Data' command was entered. [64]
1011 1111	191	
1100 0000	192	C0 - Reserved (could be used by sending C0,C0 ! – But not recommended)
1100 0001	193	Unused [47]
1110 1111	239	
1111 0000	240	F6 hex – Exit upon receipt of hF7

1111 1111	255	F7 hex – Do not Exit upon receipt of hF7 F8 hex – Send Start of block 'Timestamp' (Note this was reset at end of Block!) F9 hex – Do not Send a 'Timestamp' for note on/off, AND F8 midi words (Default) FA hex – Send a 'Timestamp' for note on/off midi words FB hex – this would cancel the Midi FIFO buffer clear on exit - Not Recommended! FC hex – Clear the MIDI receive FIFO Buffer FD hex - Do nothing with this (could be used to pad out the parameters sent with the 'Receive_MIDI_Raw_Data' message) FE hex - Reset address ie. Set to 'ignore mode' FF hex (for all boxes) indicated if the Data stream was to be passed on to the MIDI out.

Table 8-4: MIDI raw Mode flags

When a start block (C0, C1 hex) was received, this put the box into either 'send on' mode, or 'ignore' mode depending on the results of the 'Receive Address' mode. In 'send on' mode, all data received was sent out on MIDI.

If end block (C0, C2 hex) was received, then the box was put back into the 'Receive Address' mode and all addresses were cleared (ie to pass on the next block then it had to receive its address or FF hex again).

When C0,C3 hex was received, then normal network operation resumed. If control of the network was handed over to the device then the device's MIDI receive FIFO buffer was automatically cleared on exit. If during 'send on Sysex' mode another start block was received (C0,C1 hex) then this was ignored. If during 'send on Sysex' mode an end of transfer (C0,C3 hex) was received then the current transfer was ended. Note if C0 hex needed to be sent then it had to be prefixed with another C0 hex (ie C0 = C0, C0).

Notes: When using 'MIDI_Raw_Data' to pass control over to an EMI, it was recommended that unused raw data bytes were padded with FD hex, e.g. MIDI_Raw_Data(TimerVal, FD, 011a aaaa).

When a device was receiving raw MIDI data then Midi Messages from the keyboard were not processed.

The transmission rate of data across the network had to be slower than the transmission rate over MIDI (31.25 kbps). If it was faster, then the internal buffer would overflow causing data loss and unpredictable behaviour. The network operated at approximately 38.4 kbps, and when transmitting data to the device, some wait cycles had to be added. There shouldn't be a problem receiving data from the midi since the midi rate was slower than the network. However if a large number of 'C0' bytes were transferred from the midi then overrun could theoretically happen (each 'C0' byte took twice as long to send because C0, C0 was sent). However C0 shouldn't be received during Sysex Dumps, and normal playing of the keyboard would not produce a problem.

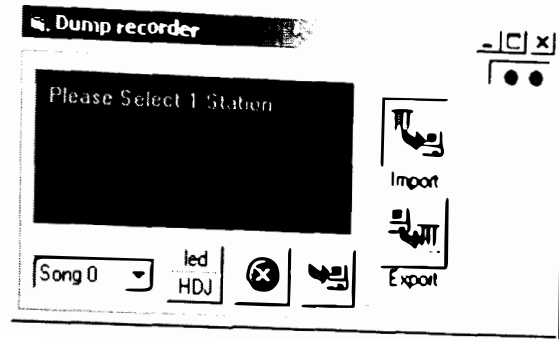


Figure 8-10: MIDI Dump handler

This utility tested the MIDI Sysex Dump handling (import / export) from different models of keyboards.

- **Recording MIDI:**

The Midi Clock (F8) Timestamp information could be turned on by sending a single byte of 32 + cc where cc was between 0 and 47). If cc was greater than 47 then this would cause problems, therefore this value had to be limited. The value of ccccc determined how many F8's were ignored between the timestamp info being sent (0 = send all F8 timestamp info, 47 = send the 48th F8 = which equates to approximately every other beat). The First F8 received once send mode was entered was always timestamped and sent.

Note: It was necessary to setup the MIDI FIFO buffer filters appropriately to allow the F8 messages to reach the FIFO Buffer. If the filters ([N2:7]) were set to filter out Real Time System messages (data1, 0x04) then no F8's would be received.

A start of block Timestamp could be turned on (F8). This was useful for synchronising the PC clock with the clock in the EMI. The timestamp was recorded as soon as the box was instructed to take control, and was sent to the pc after the C0, C1 had been sent. Note that after an End block command (C0, C2) this flag was reset, so a F8 command would need to be re sent if the next start block needed to be timestamped.

Once the time stamp info had been turned on (using FA / F8 or timestamp F8's (192 to 239)] the TimeStamp information was sent as follows:

<Packet ID> [<Seconds> [<Tenths of Second> [<Ticks High>]]] <Ticks Low>

Where Packet ID has the following values (Hex):

D3: Start of block Timestamp (always 4 trailing bytes)
D4: 'F8' Incremental Timestamp (D4 = 1, D5 = 2, D6 = 3, D7 = 4 Trailing Bytes)
D8: Note on/off Incremental Timestamp (D8 = 1, D9 = 2, DA = 3, DB = 4 Trailing Bytes)

The D3 packet was a full timestamp. The Incremental timestamp was the value since the last incremental timestamp or the start of block timestamp (or 0 if no previous timestamp).

For Note on / Note off, the timestamp was sent before the midi note on or note off char (80 → 9F hex). For Start of Block, the timestamp was sent after the C0, C1. In the F8 timestamp mode, no F8's was passed on however, the timestamp was sent after the correct number of F8's had been counted.

This utility was a MIDI recorder used to test first the MIDI transfer between one EMI and the PC and second to test the creation of understandable MIDI files. MIDI files were created from the received raw MIDI bytes stored in a temporary file and then filtered. The reception of notes timestamps was compulsory to determine the length and position of each played note. The reception of F8 timestamps allowed determination of the tempo of the song played.

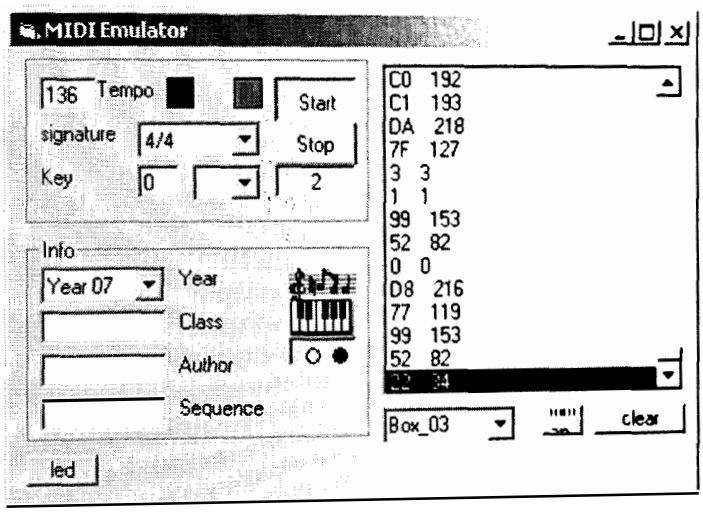


Figure 8-11: MIDI Recorder

Note that to find out how many ticks there were per tenth of a second (Deci-Second) [N0:52] Ticks_Per_Deci_Second function was used. This was dependant on the hardware and software in the EMI and could vary from one release to another.

READING THE STATE OF THE KEYS ON THE KEYBOARD

The 'Send_All_Keys_Forced' message was used to force the junction box to send the state of all its keys. The message would cause 3 separate messages to be returned. These were Global messages addressed to Master, with three data bytes.

The 'Send Keys' message would allow a mask to be specified that would be compared to a register that marked the banks of 8 keys that have changed. Only banks that have changed would be returned. If no banks of keys had changed then a blip (or 'pong') was returned. The number of packets returned depended on the number of banks that had changed. The first data byte returned was the masked value of the key change data, and could be used to determine how many packets of data would be returned. Other commands using the same format were 'Send_Centre_Keys', 'Send_Left_Keys', 'Send_Right_Keys', 'Send_Any_Keys'.

The N2:16 'Chain_Send_Key_Stat' message would cause a box to respond by generating the same message addressed to the next box (myNetAddress +1). The box would also send its Key status data as the parameter (which was ignored by the next box). A master device could simply listen to all the boxes reporting their key status and then address any that need addressing separately once the chain of responses had finished.

Finally 'Send_Key_Status' would return a global message to Master with one data byte containing the Key status. This could be used to determine if any of the banks of keys had changed value since they were last read.

This utility allowed to detect pressed keys on a portable keyboard attached to a determined EMI.

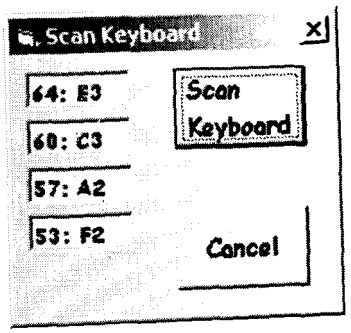


Figure 8-12: Keyboard keypress scanner

USING THE INTERNAL EEPROM

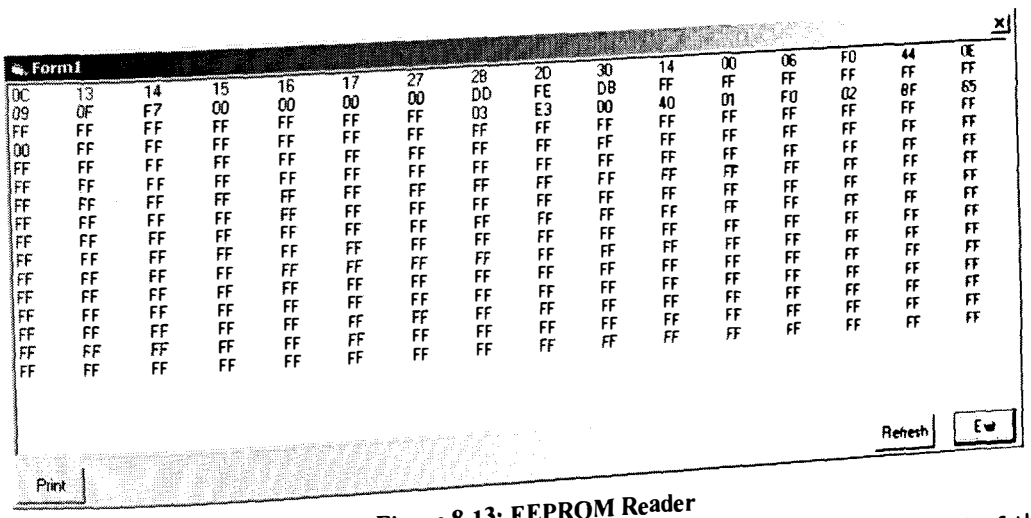


Figure 8-13: EEPROM Reader

The internal EEPROM had a 256 byte user area which was under the control of the master PC program. This area could be used to store Raw MIDI data that was often sent to a keyboard. The structure was such that it was easy to customise it against the keyboard connected to an EMI.

This utility was used to control that the custom area of the EEPROM was correctly written. The EEPROM was written from an EEPROM file where details on a MIDI keyboard were stored. Please refer to 8.2.5 for further details about this process.

- **Reprogramming the Master:**

Re-Program Master [N3-9] had to be sent first to put the master in reprogram mode. Then the bytes would be sent page per page as follows: page number, encrypted 128 bytes of the page, checksum 1 and checksum 2. After each page had been sent, FF hex was sent to exit this mode. If an error occurred, the master entered an error mode (orange LED). To exit this mode, FF hex was sent, then 256 time 00 hex. To avoid crashing an EMI because of a programming error, the first page could be replaced by a safe one and written first. The correct first page was then reprogrammed at the end.

- **Reprogramming the Slaves:**

Re-Program Slaves [N3-11] needed to be sent first to put the slaves in reprogram mode. This automatically set the Master in Master Pass-Through mode [N3-10]. Then the bytes were sent as for the master. When all the bytes were sent, the Master needed to exit the Pass through mode. This was done by sending 70 time 80, D5 hex. If an error occurred, the slave entered an error mode (orange LED). To exit this mode and / or to avoid crashing the EMIs, the same recovery routine as for the master had to be performed.

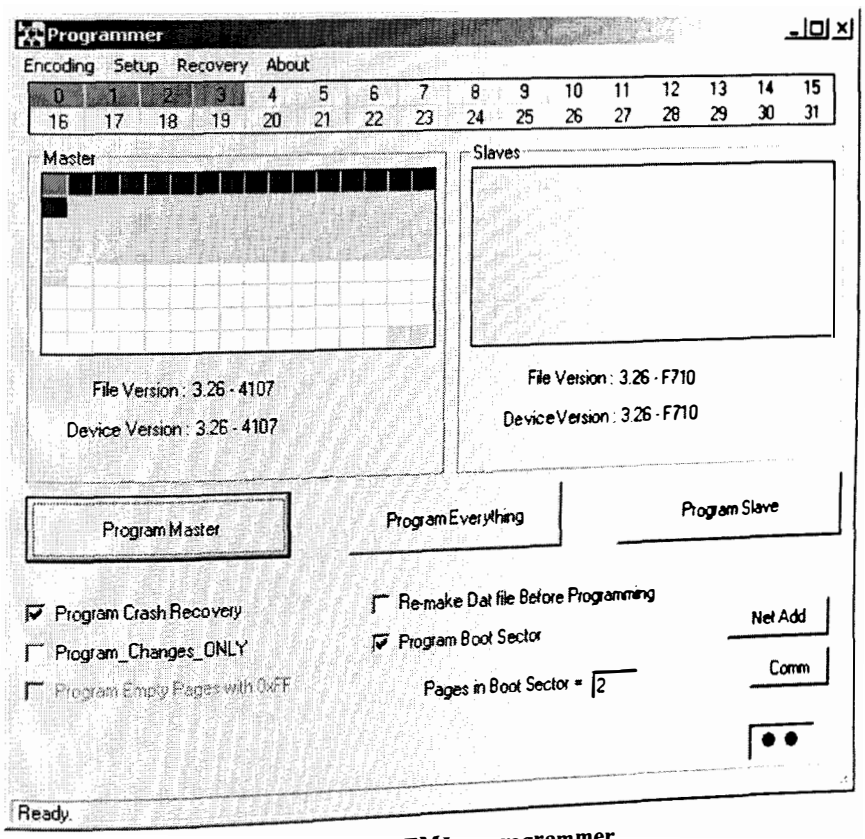


Figure 8-14: EMI re-programmer

This last utility allowed the encoding and encryption of a hex Intel file generated from the microcontroller source code compiler. Then it could check the current version of the firmware of the Master and all the slaves against the last encoded version. Finally, it managed to reprogram any EMI with different flag settings.

8.2.5. EEPROM file creation

This section consisted of:

A) Method

B) Example of an EEPROM file

A) METHOD

EMIs can process MIDI messages from an attached keyboard in real time. This allows to record MIDI and to control the keyboard for the split and duet modes. Kaan specification allowed the use of any keyboard model with the system. As all model operate differently, we use part of the EEPROM memory to store information relating to the attached keyboard to customise its control. The EEPROM can be programmed from a simple file called EEPROM.txt. The documentation below describes how to create this EEPROM file for a new keyboard.

STAND ALONE USAGE COMPATIBLE EEPROM FILE CREATION:

- *Solo/Split mode:*
  - Local on/off from a MIDI message, or MIDI mute accepted (Bn,07,00: MIDI volume on channel n = 0; B0,7A,0: Local Off).
    Note: Yamaha tends to have a special local channel that cannot be muted. Only local off can work. Some of these Yamaha cannot accept a local off message, and some others cannot be set off even from the front panel.
  - General MIDI on (sysex depending on manufacturer and models)
  - Panning OK (Bn,0A,00 for left, Bn,0A,7F for right)

MIDI messages from Reverb or Chorus functions give problems and have to be switched off:

- Reverb off (sysex depending on manufacturer and models)
- Chorus off (sysex depending on manufacturer and models)

The EEPROM consists of several fields for the different system configuration and must contain the MIDI information required in these modes:

EEPROM fields:

- [Generic]: for all modes
- [Solo] Solo mode only
- [SplitDuet Generic] Split and Duet modes
- [Split] Split mode only
- [Duet] Duet mode only

- *Check unpanning patches: Some patches cannot be panned correctly so that a right student may hear what a left student plays in split mode. These patches can be filtered.*

EEPROM fields:

- [Disabled Patches]: For patches with no panning facility. 16 bytes = 128 patches. Bytes are written as:

7-0,15-8,23-16,31-24,39-32,47-40,55-48,63-56,
71-64,79-72,87-80,95-88,103-96,111-104,119-112,
127-120. They are created bitwise (0 = off, 1 = on).
For example:
FF,FF,FB,FF,FF,FF,7F,6F,FF,FF,9C,EB,EB,FF,FF,B7

Method:

- First put FF for the 16 bytes (no filter) and reprogram the EEPROM of the EMI linked to the tested keyboard. Then, in split mode, either listen to the Left and change patches on the Right side from the keyboard keys or vice versa and write down the patches that can be heard (and need to be filtered).
  Note: The first patch is 0, a first change is patch 1.

- When all these patches have been found, the filtering bytes have to be written. For example patches 18 and 21 have to be filtered: both are in the third byte [23-16].

23	22	21	20	19	18	17	16	
1	1	0	1	1	0	1	1	
	D				B			= hDB

- Write the new filtering bytes in the file and reprogram the EMI. Retest all patches to check if OK.

- *Check with a MIDI message manager program such as MIDI-OX all the 'reset' messages sent by the keyboard when any of the panel key is pressed. From this define the few bytes that are common to the entire set of reset patterns and that is not form common use. When an EMI receives one of these bytes, it will know that a button has been pressed on the keyboard and that it will need to reset itself and the keyboard back to normal mode. Four patterns can be recognised.*

EEPROM fields:

- [Capture Pattern 1]
- [Capture Pattern 2]
- [Capture Pattern 3]
- [Capture Pattern 4]

Method:

- Get all patterns and write as few as possible in the above fields. Then Reprogram EMI and put it in Split mode. Press all the buttons of the panel and check their action. Sometimes the EMI cannot fully control the keyboard and some buttons need to be disabled. Usually, it is achieved by cutting tracks from the front (CTK651) or by removing the button and sticking it back in (all new Casio).

- *[Power On Delay] is used to tell the EMI after how long after it has been powered in it can send the reset MIDI messages to the keyboard, as this one has a long reset startup process. It is expressed in 1/10th seconds (and written in hexadecimal). 0 means do it ASAP.*

STAND ALONE USAGE AND NETWORK COMPATIBLE EEPROM FILE CREATION:

- *[Kbd Fct Flags] is a bitwise flag telling the EMI MIDI information about the attached Keyboard (for MIDI dump). The information is available from the MIDI implementation chart of the keyboard user guide.*

  - bit 0: Kbd supports local On/Off (1=true)
  - bit 1: Kbd supports initiate receive bulk dump
  - bit 2: Kbd supports initiate send bulk dump

B) EXAMPLE OF AN EEPROM FILE

```
[Generic]
;  Reverb Off and Chorus Off
F0,44,11,01,7F,01,19,06,00,00,00,00,F7
F0,44,11,01,7F,01,11,06,00,00,00,00,F7

[Solo]

[SplitDuet Generic]

[Split]

[Duet]

[Disabled Patches]
; For patches with no panning facility. 16 bytes = 128 patches
FF,FF,FB,FF,FF,FF,7F,6F,FF,FF,9C,EB,EB,FF,FF,B7

[Capture Pattern 1]
B0,20

[Capture Pattern 2]
B4,20

[Capture Pattern 3]

[Capture Pattern 4]

[Power On Delay]
;Not a pointer to a block
;specified in 1/10th seconds
;FF: does not send the reset MIDI messages
;0: doest it ASAP
14

[Kbd Fct Flags]
;Not a pointer to a block
;bit 0: Kbd supports local On/Off (1=true)
;bit 1: Kbd supports initiate receive bulk dump
;bit 2: Kbd supports initiate send bulk dump
6
```

8.3. Base Software

8.3.1. Study on Exiting User Interfaces

Program	Field	Nb of fcts	Nb of Menus	Nb of S/menus	Nb of Buttons	Ease of use Beginner	Ease of use Prof	Use of Kbd	Use of Mouse	Use of Others	Use	layout	Help
autocad	CAD/E	4	9	34	#	1	3	3	3	0	A	1 sheet	F1 - Text
ranger	CAD/E	4	16'	6'	25'	1	3	1	3	0	A	upon fct	F1 - Text
explorer	Explorer	1	5	3	13	3	4	1	3	1	B	2 windows	F1 - Text + Images
internet explorer / netscape	Explorer	2	6	8	13	3	4	1	3	1	B	HTML page	F1 - Text + Images
my computer	Explorer	1	4	1	13	3	4	1	3	1	B	1 form	F1 - Text + Images
cubase	Interfacing	4				1	3	1	3	1	C	MDI Children	
sound forge	Interfacing	3				2	3	1	3	1	C	MDI Children	
avr isp (external I/F)	Interfacing	2	7	12'''	14	3	4	2	3	1	C	MDI Children	
cd player / mixer	Interfacing	1	4	0	15	3	4	0	4	1	B	1 Form + controls	F1 - Text + Images
dv studio	Interfacing	1				3	4	0	4	1	C	MDI Children	
tv32	Interfacing	2				2	4	0	4	1	B	Active page + MDI children	F1 - Text + Images
encarta atlas	Knowledge	2				3	4	2	3	0	A	HTML page	
encarta autoroute	Knowledge	2				2	4	2	3	0	A	Maps	
encarta encyclopaedia	Knowledge	2				3	4	2	3	0	A	HTML page	

Program	Field	Nb of fcts	Nb of Menus	Nb of S/menus	Nb of Buttons	Ease of use Beginner	Ease of use Prof	Use of Kbd	Use of Mouse	Use of Others	Use	layout	Help
access	Office	3	7	12	#	1	3	4	2	0	A	Tables	office assistant
excel	Office	3	10	19	#	2	3	4	2	0	A	Spreadsheets	office assistant
power point	Office	3	9	10	#	2	4	3	3	0	A	Slides	office assistant
word	Office	3	10	9	#	3	3	4	2	0	A	Pages	office assistant
3d studio	Pictures	4	@			1	3	2	3	0	A	4 viewing frames	
Photoshop / paintshop pro	Pictures	4	9@	30	27	1	3	1	3	0	A	MDI Children	F1 - Text + Images
vb6	Programming	4	13@	18	#	1	3	4	4	0	A	MDI Children + Menus	MSDN / Wizard
winzip	Utility	2	4	1	8	2	4	0	4	0	A	1 Window	F1 - HTML
EMI I/F (external I/F)	**Interfacing**	**3**				**3**	**4**	**0**	**4**	**1**	**C**	***Classroom***	**Wizard**

Table 8-5: Existing types of Interfaces

Legend:

Range: 1 ← → 4
Low High
Difficult Easy

Symbols:

- [0..4]: 0: less, 4: more
- #: user dependant
- n': function dependant
- n": Right click only
- n"': sub menus = forms
- n@: tool menu / form
- A: data Handling
- B: I/F with H/W & NO data Handling
- C: I/F with H/W & data Handling

8.3.2. Kaan Database System Analysis

This section consisted of:
A) Requirements
B) Top / Down Analysis
C) Database Design – Tables Relationships

A) REQUIREMENTS

- Management of **Classes**:
  - Class details: Defined by name, year group and Academic Year it started
- Management of **Students**:
  - Student details
  - Set with classes
  - Active class to set student's current class (could have been in many)
  - Imported results
- Management of **Workpieces**:
  - General details with marking scheme
  - Set with students ➔ student specific details + group comments
  - Set with Lessons ➔ set lesson marking schemes + settings + resources
  - Set with a unit
- Management of **Teachers**:
  - Teacher details
  - User policy
  - Set with classes
  - Set with Student Activities
  - Set with tutorGroups
- Management of **Resources:**
  - Resource details
  - Instruments / family
- Management of **ClassroomLayouts**:
  - Stations details and location
  - Set Student placing
  - Multiple rooms
- Management of **Lessons**:
  - Lesson details (date)
  - Set with a Class
  - Set with a room
  - Set with a Workpiece (to get test marking scheme + resources)
  - Registration
- Management of **MarkingSchemes**:
  - Details (name + intermediate grades)
  - Marking grades
- Management of **Units**:
  - Unit details
  - Set with class
- Management of **Activities**:
  - Activity details
  - Set with Students
  - Set with Teachers

B) TOP / DOWN ANALYSIS

ENTITIES:

- SubjectClass
- Student
- StudentClasses — a student in a class
- ImportedResults
- Workpiece
- MarkingScheme
- StudentWorkpieces — workpiece set to a student with specific details
- GroupWork — collection of StudentWorkpieces
- Lessons
- Resource
- Unit
- Teacher
- Activities
- TutorGroups
- InstrumentFamily
- ClassroomLayout — Station on which can sit students
- Register — student status for a lesson

ENTITIES RELATIONSHIPS:

• 1	Student	can be with	MANY	StudentClasses
• 1	SubjectClass	can set	MANY	StudentClasses
• 1	ClassroomLayout	can be set with	MANY	StudentClasses
• **MANY**	**Teachers**	**can teach to**	**MANY**	**SubjectClasses**
• 1	Workpiece	can create	MANY	StudentWorkpieces
• 1	Student	can be set with	MANY	StudentWorkpieces
• 1	GroupWork	can be created by	MANY	StudentWorkpieces
• **MANY**	**Resources**	**can be set with**	**MANY**	**Workpieces**
• 1	MarkingScheme	can be set with	MANY	Workpieces
• 1	Unit	can be set with	MANY	Workpieces
• **MANY**	**Units**	**can be set with**	**MANY**	**SubjectClasses**
• **MANY**	**Workpieces**	**can be set with**	**MANY**	**Lessons**
• 1	Lesson	can generate	MANY	Registers
• 1	Class	can set with	MANY	Lessons
• 1	Student	can generate	MANY	Registers
• 1	Student	can have	MANY	ImportedResults
• 1	Student	can have	MANY	Activities
• 1	Teacher	can participate in	MANY	Activities
• 1	TutorGroup	can gather	MANY	Students
• 1	Teacher	can teach	MANY	TutorGroups

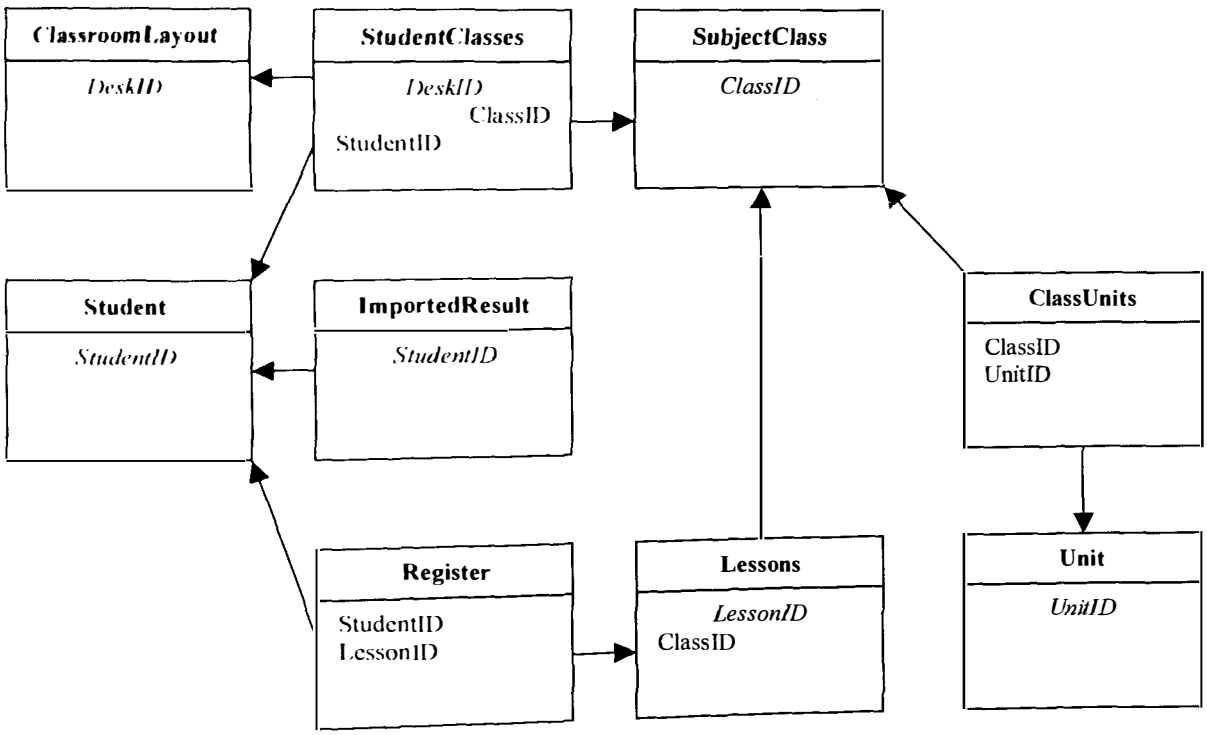
All **MANY to MANY** relationships (in bold) will generate extra tables:

- TeacherClasses
- WorkpieceResources*
- ClassUnits
- LessonWorkpieces

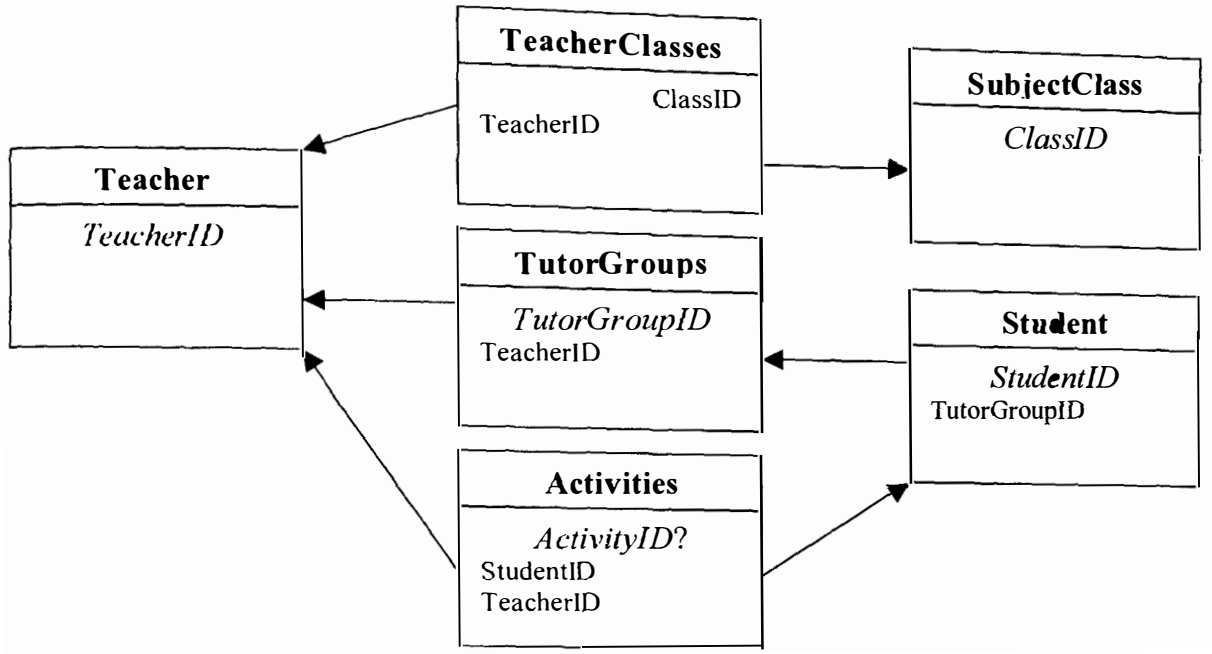
*: This table was not created as resources were initially considered to be always attached to 1 workpiece.

TABLE RELATIONSHIPS:

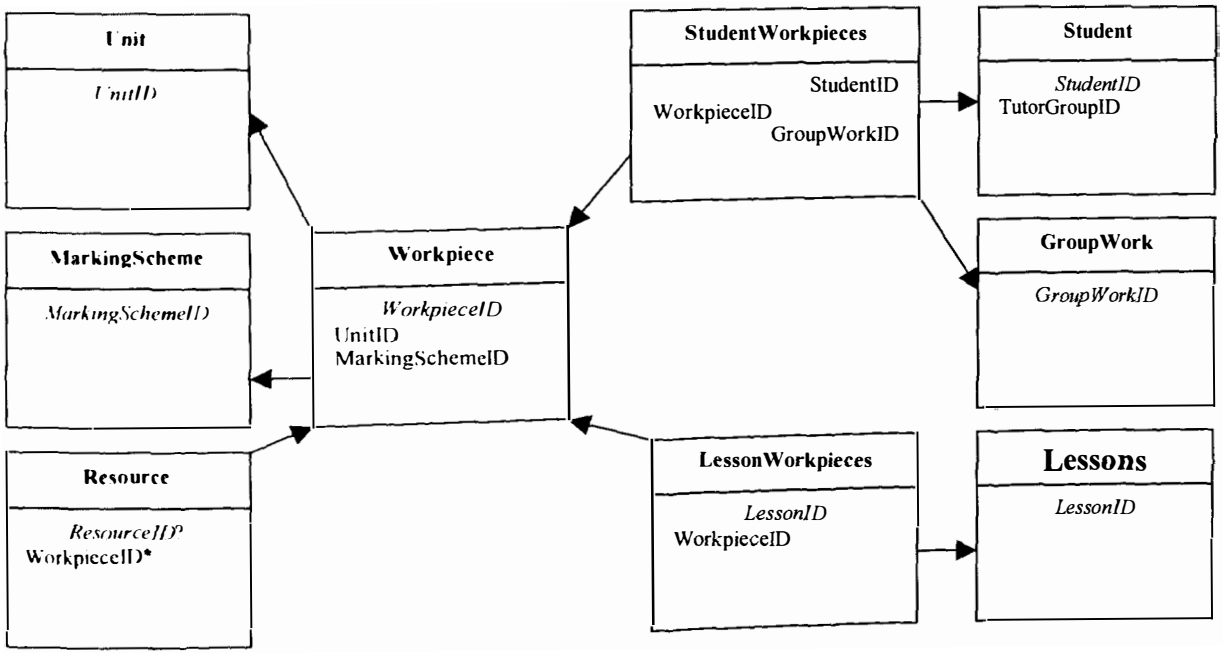
• 1	Student	can be with	MANY	StudentClasses
• 1	SubjectClass	can set	MANY	StudentClasses
• 1	ClassroomLayout	can be set with	MANY	StudentClasses
• **MANY**	**Units**	**can be set with**	**MANY**	**SubjectClasses**
• 1	Lesson	can generate	MANY	Registers
• 1	Class	can set with	MANY	Lessons
• 1	Student	can generate	MANY	Registers
• 1	Student	can have	MANY	ImportedResults



• **MANY**	**Teachers**	**can teach to**	**MANY**	**SubjectClasses**
• 1	Student	can have	MANY	Activities
• 1	Teacher	can participate in	MANY	Activities
• 1	TutorGroup	can gather	MANY	Students
• 1	Teacher	can teach	MANY	*TutorGroups*



- 1 Workpiece can create MANY StudentWorkpieces
- 1 Student can be set with MANY StudentWorkpieces
- 1 GroupWork can be created by MANY StudentWorkpieces
- **MANY Resources can be set with MANY Workpieces**
- 1 MarkingScheme can be set with MANY Workpieces
- 1 Unit can be set with MANY Workpieces
- **MANY Workpieces can be set with MANY Lessons**



*: Resources were first thought to be always attached to 1 workpiece.

ENTITIES ATTRIBUTES:

*: Primary keys.

- SubjectClass
  - SyClassID — AutoNumber
  - SyYearGroup* — Number
  - SyClass* — Text
  - SyYearStart* — Text
  - SyNotes — Memo
- Student
  - SstudentID* — AutoNumber
  - StutorgroupID — Number
  - Ssurname — Text
  - Sfirstname — Text
  - Smiddlenames — Text
  - Snickname — Text
  - Sgender — Enum (Male, Female, Undefined)
  - Sdateofbirth — Date/Time
  - Sregno — Text
  - Spicture — Text
  - SSEN — Memo
  - Snotes — Memo
- StudentClasses
  - scstudentID* — Number
  - scclassID* — Number
  - scdeskID — Number
  - scactive — Boolean
  - scside — Enum (None, Left, Right, Both)
- ImportedResults
  - IrstudentID* — Number
  - IrTestname* — Text
  - IrDate* — Date/Time
  - Irmark — Text
- Workpiece
  - WworkID* — AutoNumber
  - WunitID — Number
  - WmarkingschemeID — Number
  - Wworktitle — Text
  - Wskills — Enum (other, composing, performing, listening)
  - Wothername — Text
  - Wsettings — Memo (script)
  - Wdescription — Memo
  - Wassociatedwork — Text
  - WNClinks — Text
- MarkingScheme
  - MsschemeID* — AutoNumber
  - Msschemestring — Memo (script)
- GroupWork
  - GwgroupID* — AutoNumber
  - Gwcomment — Memo

- StudentWorkpieces
  - SwstudentID* Number
  - SwworkID* Number
  - SwgroupID Number
  - Swmark Text
  - Swcomments Memo
  - Swattachmenttypes Memo
  - Swattachmentfiles Memo
  - Swdate Date/Time
- Lessons
  - LlessonID* AutoNumber
  - LclassID Number
  - Lroom Text
  - Lstartdatetime Date/Time
  - Lduration Text
- Resource
  - rresourceID* AutoNumber
  - rworkID Number
  - rtype Text
  - rfilename Memo
  - rtitle Memo
  - rauthor Text
  - rartist Text
  - rinstrument1 Text
  - rinstrument2 Text
  - rgenre Text
  - rstyle Text
  - ryear Text
  - rcomments Memo
- Unit
  - UunitID* AutoNumber
  - Uname Text
  - Uyeargroup Date/Time
  - Uunitnum Text
  - Uunittype Number
  - Unotes Memo
- Teacher
  - TteacherID* AutoNumber
  - Tsurname Text
  - Tfirstnames Text
  - Tinternal Boolean
  - Taddress Memo
  - Ttel Text
  - Tmobile Text
  - Tnotes Memo
  - Tpassword Text
  - Texpires Date/Time
  - Tchangenext Boolean
  - Tusername Text
  - TloggedON Boolean
- TutorGroups
  - Tgtutorgroup* Text
  - TgteacherID Number

- Activities
  - AactivityID — AutoNumber
  - AstudentID* — Number
  - AteacherID — Number
  - Ainstrument* — Text
  - Aactivity* — Text
  - Agrade — Text
  - Ainternal — Text
  - Adate — Text
  - Anotes — Memo
- ClassroomLayout
  - CldeskID* — AutoNumber
  - Clroom — Text
  - ClEMInumber — Number
  - Cltop — Number
  - Cleft — Number
  - Clorientation — Enum (Top, Left, Bottom, Right)
  - ClmaininputL — Number
  - ClmaininputR — Number
  - ClAuxL — Number
  - ClAuxR — Number
- Register
  - RgLessonID* — Number
  - RgStudentID* — Number
  - Rgstatus — Enum (Unregistered, Present, Absent, Late)
- TeacherClasses
  - TcteacherID* — Number
  - TcclassID* — Number
- ClassUnits
  - CuClassID* — Number
  - CuUnitID* — Number
- LessonWorkpieces
  - LwLessonID* — Number
  - LwWorkpieceID* — Number
- InstrumentFamily?
  - IfFamilyID* — AutoNumber
  - IfInstrument — Text

C) DATABASE DESIGN – TABLES RELATIONSHIPS

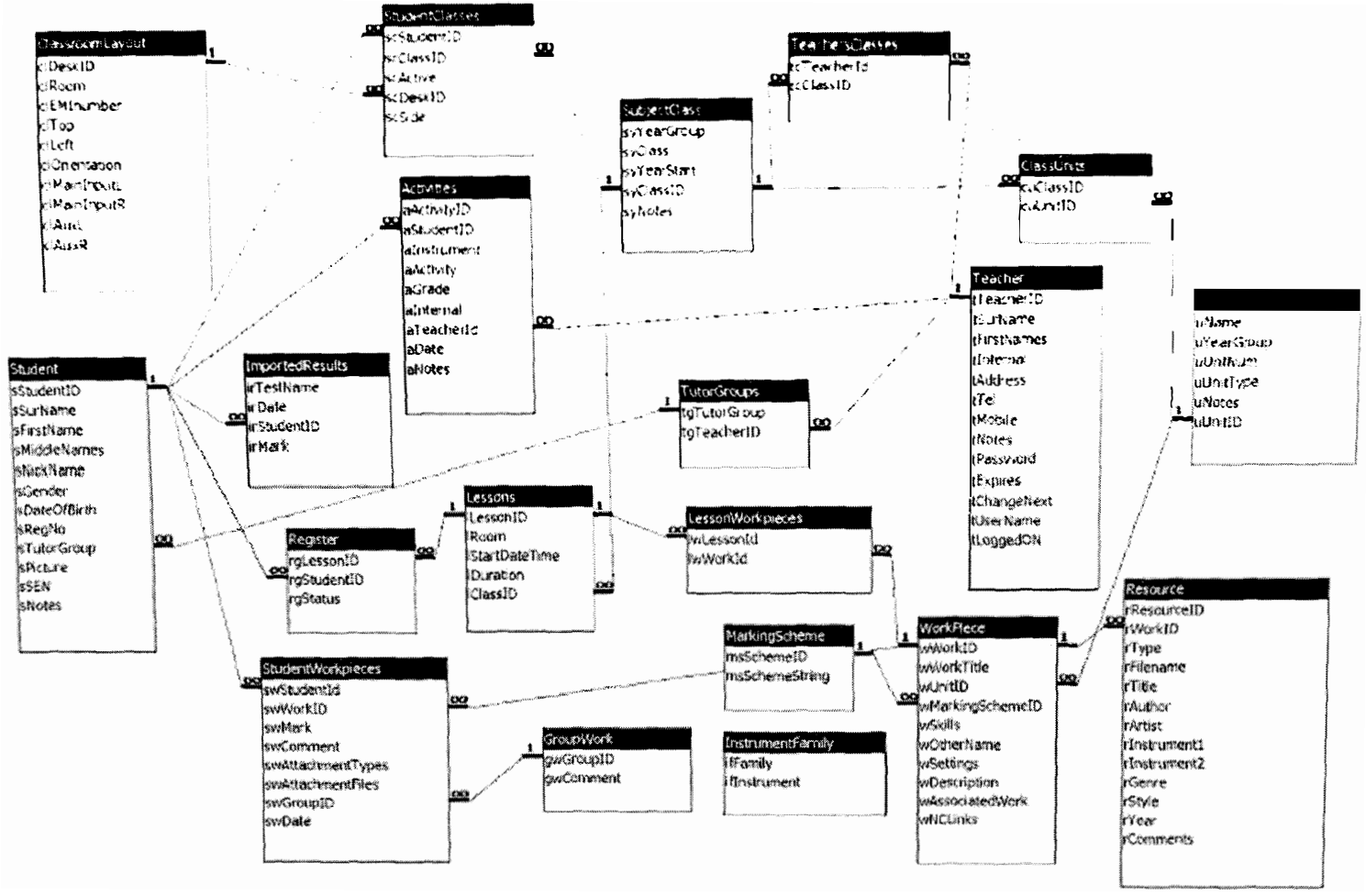


Figure 8-15: ESAAMS - KAAN Database: Tables relationships

8.3.3. KAAN - ESAAMS Documentation (release 3)

This section consisted of:

A) UML System Analysis (SFG / SF / UC relationships)
B) Objects Relationships

A) UML SYSTEM ANALYSIS (SFG / SF / UC RELATIONSHIPS)

SFG	SF	UC
ESAAMS Database Interfacing	Manage classes	Load Class
		Close Class
		New Class
		Import Class and students from SIMS
		Add / import Students to Class List
		Edit Academic year classes
		Edit Class Details
		Generate Class students unique nickname
		Remove Student from Class list
		Delete Class
	Manage Students+Placing	New student
		Edit Student Details
		Find student and edit details
		Load student picture from file / no picture
		Capture student picture
		Delete Student
		Set student name view
		AutoPlace all
		Place / unplace 1 student
		Lock / Unlock Student Placing
		Unplace All Students
	Manage Teachers/Users	Change Current User Password
		New User
		Edit Users
		Delete Users
	Manage Lessons/Registration	AutoRegister
		Show registration
		New registration
		Update registration
		Set Registration display
	Manage Workpieces	Set workpieces manager display
		New Marking Scheme
		Edit Marking Scheme Details
		Delete Marking Scheme
		Store Workpiece and create Record in database
		Store Workpiece as a file only
		Playback Just Recorded Workpiece
		Playback Workpiece in Student Card
		Set Playback Volume (mini player)
		Send Workpiece to Player or default application
		Send workpiece to Dump Handler
		Import Student Workpiece
		Export Student Workpiece
		Edit Student Workpieces
		Find Workpiece file
		Edit Class workpieces
		Edit Workpiece details (with multiple students and files)

		Get grouped students / workpiece + jump2Student
		Delete Workpiece
	Manage Rooms/layouts	Edit default room (change layout only)
		New room
		Delete room
		Set default room
		Rename room
	Reporting	Class report
		Student report
	Manage Resources	Refresh resources
		Edit and Filter Resources
		New resource playlist
		Load resource playlist
		Save resource playlist
		Send temp resource playlist to player
	Manage Database and data	Password forgotten
		Startup (prechecks prior to login)
		Startup (database caching after login)
		New academic year checking and routine
		Database location checking and setting
		User login with expiring date
		Administrator Account setup with licensing and EULA
		Database Locking Check
		Database Offline working
		Database Offline restoring
		Change academic year folder location
		load external database
		Close external database
		End and terminate services properly
		Backup
		Set AutoBackup
		Set Backup Folder
		Offline work setting
		Offline work asking
		Offline work preparing
		System Locking / unlocking
KAAN-PC system Interfacing	Playback	Play CD to selection
		Play File to selection (any audio-video file+Playlist)
		Set Player Volumes (main and others) + mutes
		Set / reset Playback loop
		Play in bigger resizable window
		play full screen
		Set start and stop markers
		change frame rate
		zoome in/out
	Record	Record audio
		Record video
		Set recorder settings(audio/ video source and quality)
		Set Recorder Volume
	Other / settings	Hide / Show Clock
		Hide / Show popup window
		Maximise / minimise popup window
		Lock / unlock popup window
		Lock Interface
		Select stations / students
		Show about
		Show Notepad in popup window
		Show Browser in popup window

	Show Resource Collection Manager in popup window
	Show Settings in popup window
Manage system status	Disable EMI
	Find System
	Scan / refresh and cash system status
	Test communication with EMI
	Work Offline
	Work Online
	Address Kaan system
	Check for Mobile EMIs
Manage control/setting functions to selected stations	Mute All
	Mute none
	Reset ALL to Current Settings
	Reset All to Startup Setting
	Reset EMI to Current Settings
	Reset EMI to Startup Settings
	Set Keyboard Instrument to selection
	Set Keyboard Local Student functions (lock) to selection
	Set Keyboard Mode to selection
	Set Keyboard Mute to selection
	Set Keyboard Transpose Level to selection
	Set Keyboard Use of Rhythm to selection
	Set Local Teacher Functions (lock)
	Show Selection settings
	Show setting window
Reset to preset or current settings	Initialise System
	Reset ALL to Current Settings
	Reset All to Startup Setting
	Reset EMI to Current Settings
	Reset EMI to Startup Settings
	Reset to Last Class Settings
	Reset to saved Default Class Settings
	Save Default Class Settings
	Save Startup Settings
Group selected stations	Autopairing (autogrouping)
	Group selected stations
	Ungroup ALL
	Ungroup selected group
Manage MIDI Dump/Sequence	Download MIDI Dump from Selection
	Upload MIDI Dump to Selection
Select+Listen to selected stations	Listen to Selection
	Select Stations
	Lock / Unlock Selection
Speak to selected stations	Set Speak Automuting
	Set Speak Muting (when teacher speaks)
	Speak to ALL
	Speak to selection
	Set Sound cards
	Set Mike to Speak
	Set Speak Mike/Teacher Keyboard Volume
	Set Teacher - Student Volume balance
	Set Teacher Keyboard to Speak
Selected stations perform to the class	Set Perform to selection

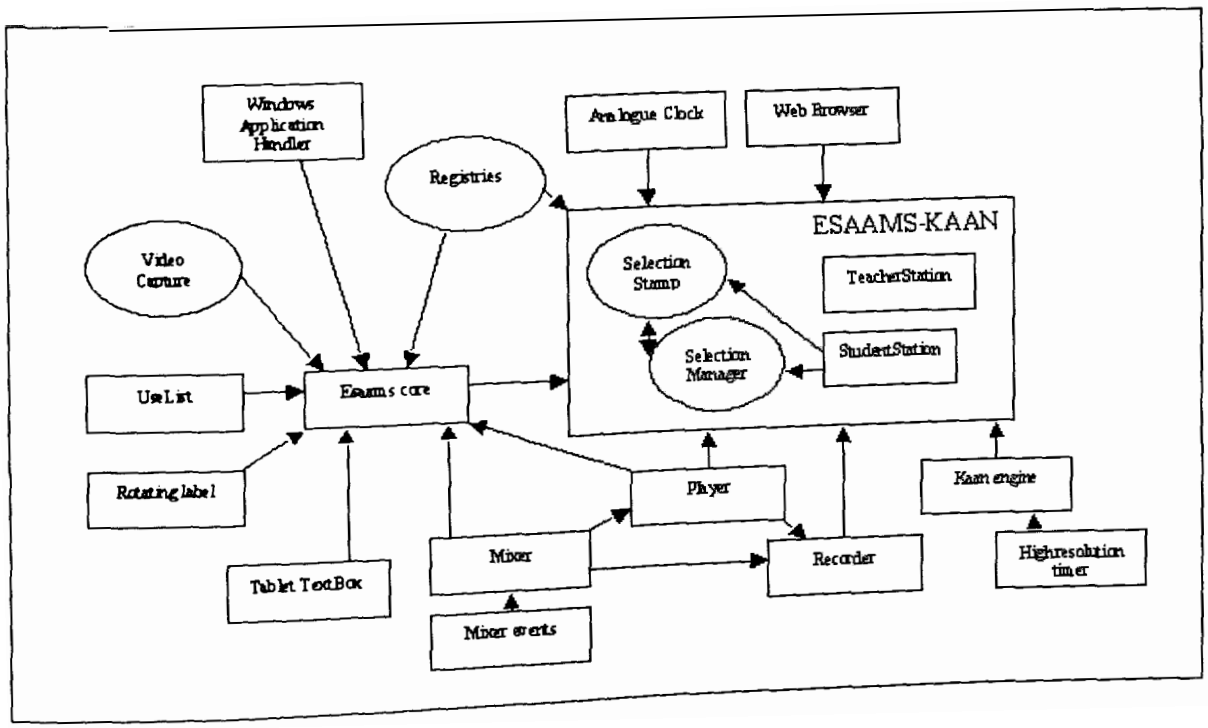
B) OBJECTS RELATIONSHIPS

MAIN (EXE), COM (DLL, OCX) AND OTHER OBJECTS RELATIONSHIPS

Contained Entities:

• ESAAMS-KAAN	Exe	Main application
• StudentStation	Public User control	User Control
• StudentStation	Public User control	User Control
• SelectionManager	Public	Class
• SelectionStamp	Public	Class
• Analogue Clock		ocx
• Esaams Core		ocx
• High Resolution Timer		ocx
• Kaan Engine		ocx
• Mixer		ocx
• Mixer Events		ocx
• Player		ocx
• Recorder		ocx
• Rotating Label		ocx
• Tablet TextBox		ocx
• UseList		ocx
• Web Browser		ocx
• Windows Application Handler		ocx
• Registries		dll
• Video Capture		dll

Concept Diagram:



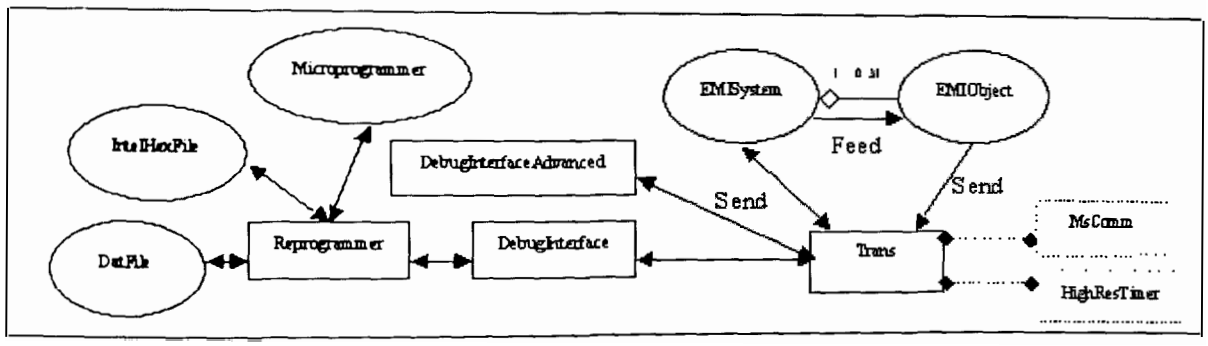
OBJECTS RELATIONSHIPS FOR KAAN ENGINE OCX

Description: Contains the objects to interface with Kaan system (communication) and store system status.

Contained Entities:

• Trans	OCX display / Public	User Control
• DebugInterface	Public User control	User Control
• DebugInterfaceAdvanced	Public User control	User Control
• ReProgrammer	Public User control	User Control
• EMIObject	Public	Class
• EMISystem	Public	Class
• DatFile	Public	Class (not to be used)
• IntelHexFile	Public	Class (not to be used)
• Microprogrammer	Public	Class (not to be used)
• KaanEnginePublic		Module

Concept Diagram:



OBJECTS RELATIONSHIPS FOR ESAAMS CORE OCX

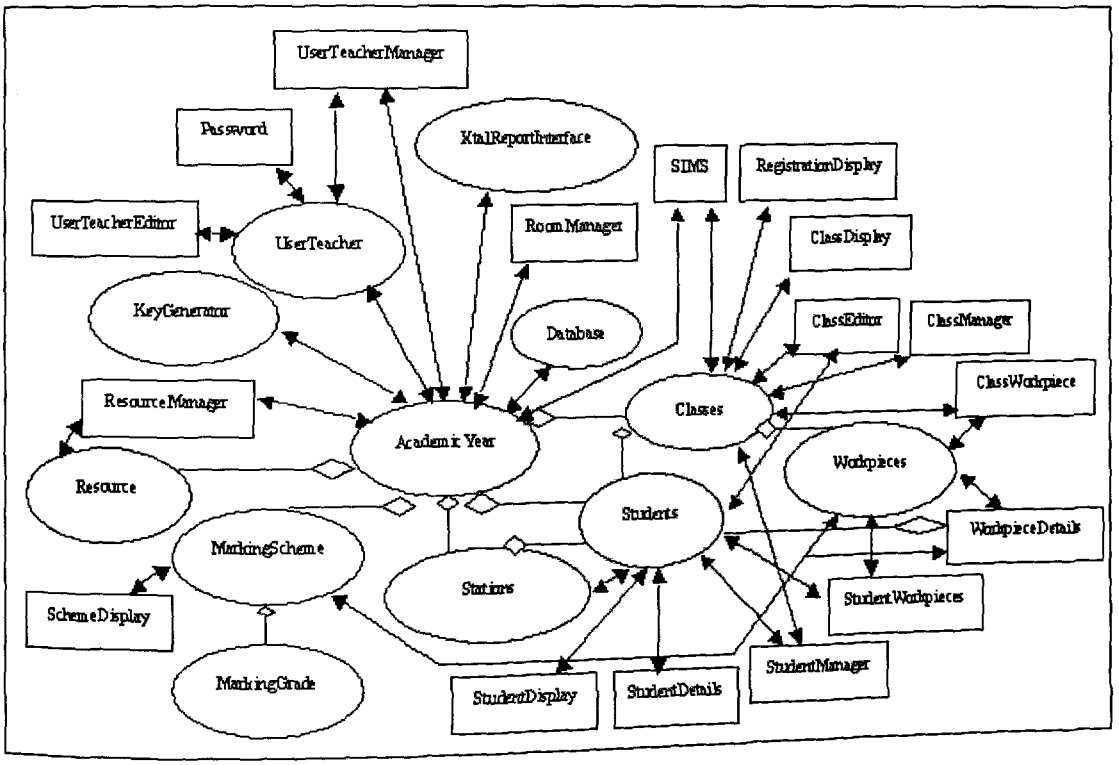
Description: Contains the objects to interface with KAAN - ESAAMS database (Information System) and store system status. Tablet PC enabled.

Contained Entities:

• ClassDisplay	Public User control	User Control
• ClassEditor	Public User control	User Control
• ClassManager	Public User control	User Control
• ClassWorkpieces	Public User control	User Control
• Password	Public User control	User Control
• RegistrationDisplay	Public User control	User Control
• ResourceManager	Public User control	User Control
• RoomManager	Public User control	User Control
• SchemeDisplay	Public User control	User Control
• SIMS	Public User control	User Control
• StudentDetails	Public User control	User Control
• StudentDisplay	Public User control	User Control
• StudentManager	Public User control	User Control
• StudentWorkpieces	Public User control	User Control
• UserTeacherEditor	Public User control	User Control
• UserTeacherManager	Public User control	User Control

• WorkpieceDetails	Public User control	User Control
• AcademicYear	Public	Class
• Classes	Public	Class
• Database	Public	Class
• KeyGenerator	Public	Class
• MarkingGrade	Public	Class
• MarkingScheme	Public	Class
• Resource	Public	Class
• Stations	Public	Class
• Students	Public	Class
• UserTeacher	Public	Class
• Workpieces	Public	Class
• XtalReportInterface	Public	Class
• PublicEsaams		Module

Concept Diagram:



OBJECTS RELATIONSHIPS FOR MIXER OCX

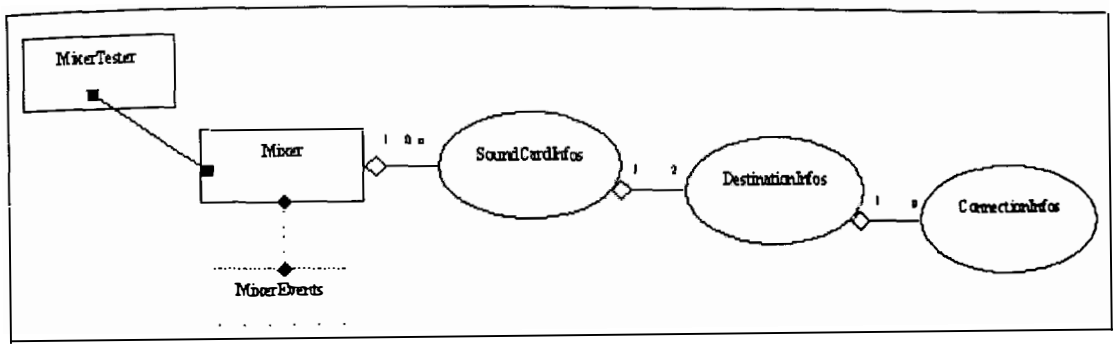
Description: Interface to windows mixer and sound cards (Read/Write + events). Multi Sound cards, built hierarchically:

- Mixer (1)
  - Sound card (n)
    - Destination (usually 2: Out – Wave Out / Record – Wave IN)
      - Connection (depends on sound cards, stuff to be set and usually are Boolean or integer ➔ check box interface or slider)

Contained Entities:

• Mixer	OCX display / Public (No display = container)	User Control
• MixerTester	OCX Public	User Control
• SoundCardInfos	Public	Class
• DestinationInfos	Public	Class
• ConnectionInfos	Public	Class
• MixerPublic		Module

Concept Diagram:



8.4. Information gathering

This section consisted of:

A) Survey 1: Results
B) Survey 2: Results
C) Interviews: Results

A) SURVEY 1: RESULTS

CURRENT KAAN FEATURES

0: What is it!, 1: Never, 2: Never but can use it in Future, 3: sometimes, 4: Often: 5: Always

Features	Bideford	Charlton	Davisons	East Brighton	Holly Lodge	Holy trinity	Perins	Poole	tot	number replies	average
1. Forgot your password		1	1			1	1		4	4	1
2. Change password		4	1			1	1		7	4	2
3. Lock interface		3	5			0	0		8	4	2
4. Use system with no control (offline)		1	0			3	3		7	4	2
5. Restore communication in offline mode		1	0			0	3		4	4	1
6. Clock		5	5			1	5		16	4	4
7. Alarm		3	1			1	0		5	4	1
8. Select all / all left / all right (Release 2 Users ONLY)		4	5				4		13	3	4
9. Select with shift and ctrl keys		4	1			4	1		10	4	3
10. Speak device setting and level balancing		4	4			3	4		15	4	4
11. Speak to all		5	4			4	3		16	4	4
12. Automute		3	5			4	3		15	4	4
13. Muting when teacher speaks		5	5			4	3		17	4	4
14. Mute all		5	5			4	3		17	4	4
15. Mute 1 station		4	4			1	3		12	4	3
16. Keyboard modes (solo/split)		4	5			3	4		16	4	4
17. Duet mode		4	4			3	4		15	4	4
18. Instrument voices setting		4	4			3	4		15	4	4
19. Instrument sections		4	4			3	4		15	4	4
20. Instrument locking		4	1			3	4		12	4	3
21. Instrument favourites management		1	1			3	1		6	4	2
22. Transpose		3	3			2	4		12	4	3
23. Lock local settings		4	4			3	4		15	4	4
24. Lock Teacher keys		3	4			0	5		12	4	3
25. Use Students, local function keys (split/duet mode only)		1	4			3	3		11	4	3
26. Use of rhythm (solo mode only)		4	4			4	3		15	4	4
27. Grouping/ungrouping		4	4			4	4		16	4	4
28. Auto grouping		4	5			1	1		11	4	3
29. Play an audio file		4	5			3	3		15	4	4
30. Play a cd track		4	5			3	3		15	4	4
31. Record work as wave		3	4			3	3		13	4	3
32. Record work as mp3		4	4			4	4		16	4	4

33. Record work as midi		3	3			1	3		10	4	3
34. Record work onto tape		1	1			2	1		5	4	1
35. Perform		4	4			4	4		16	4	4
36. Perform with player and/or recorder		4	5			3	1		13	4	3
37. MIDI dump management (Release 2 Users ONLY)		3	1				4		8	3	3
38. Reset to current settings		3	4			3	4		14	4	4
39. Reset to startup settings		3	5			1	3		12	4	3
40. Reset to last class settings (Release 2 Users ONLY)		3	5				3		11	3	4
41. Manage default class settings (Release 2 Users ONLY)		3					3		6	2	3
42. Create a new student		4	4			3	3		14	4	4
43. Create a new class		4	3			3	3		13	4	3
44. Edit Classes		4	3			3	3		13	4	3
45. Set students' name viewing (Release 2 Users ONLY)		4	3				4		11	3	4
46. Generate students' nickname (Release 2 Users ONLY)		4	1				4		9	3	3
47. Use Registration facility (Release 2 Users ONLY)		5	5				3		13	3	4
48. Autoregistration (Release 2 Users ONLY)		5	5				0		10	3	3
49. Registration views (Release 2 Users ONLY)		5					1		6	2	3
50. Change class name and/or year		4	3			2	1		10	4	3
51. Edit students details		5	4			3	3		15	4	4
52. Set student's picture (picture capture with webcam) (Release 2 Users ONLY)		3	1				1		5	3	2
53. Import student's workpieces		3	4			2	3		12	4	3
54. Export student's workpieces		3	4			2	3		12	4	3
55. Send a workpiece to the player or recorder (Release 2 Users ONLY)		3	5				3		11	3	4
56. Use SIMS import facility		3	3			1	1		8	4	2
57. Edit marking schemes		4	3			3	3		13	4	3
58. Assess students' workpieces		5	5			4	3		17	4	4
59. Delete classes		3	3			2	1		9	4	2
60. Delete students		3	3			3	3		12	4	3
61. Find students to edit		3	3			3	1		10	4	3
62. Backup facility (Release 2 Users ONLY)		3					5		8	2	4
63. Right popup settings window disappearing options		1	0			1	1		3	4	1
64. Startup settings management		3				1	1		5	3	2

NEW KAAN FEATURES YOU REQUIRE

	Feature Name	Details
Charlton		
	Record in mp3	waiting for mp3s to be saved (after students performance, takes a long time during which student cannot hear you speak)
Davisons		
	Record in mp3	takes time to save / convert. During this time it would be good to be able to type comments in the profile.
	Database / report	database does not directly correspond with excel. If it did, I could do my annual reports on this.
	GUI	when editing a class / importing a student, you need to see the class in front of you to check the list as you go
Holy trinity		
	Notebook while recording	write workpiece comments while recording to be imported when needed
	CD playback to station without teacher having to listen	when no one selected: option?
	Delete work	with securities
	Append recordings - track edtion	-wave editor befor conversion to mp3 -append recordings when first part is OK just the secon part can be re-recorded
	Set instruments	non GM instruments (drums)
	Registration	use registration and/or database features in different rooms with kaan or other programs
Perins		
	Registration	When teaching the class in a different classroom room it would be useful to have the registration part of the programme on my lap top and be able transfer info between laptop and main Kaan computer

PROPOSED NEW KAAN FEATURES

Blank=0, C=1, B=2, A=3

Features	Tot	Bideford	Charlton	Davisons	East Brighton	Holly Lodge	Holy trinity	Perins	Poole
Lesson creation (resources + notes + settings + marking schemes + tests)	8		3	3			2		
Handle any kind of Workpiece / resources and implement appropriate editing tools (as well as visual resources)	6			3			3		
Advanced SIMS import / export	5						2	3	
Audio tools implementation (Audio explorer, encoder/ripper, drum editor, wave/mp3/midi editor, CD writer, video/still capture,...)	5			2			3		
Dual Network System (a PC on each student's desk)	5		2	3					
Email facility (in + out)	5						2	3	
On line help/tips/updates/manual	5		3				2		
Remote control with PDA or webpad	5		2	3					
Intelligent Student modelling tool for reporting and producing customised tasks	4		2				2		
links between Workpieces	4			1			3		
Resource management with Internet browser	4			1				3	
Resources/lessons/... server	4		2				2		
Reporting (Crystal report type)	3							3	
Users / teachers (logging + custom usage levels)	3			3					
Links with external software and resources within software (Auralia,... + Composer notes +...)	2			1			1		
Remote control for diagnostics	2		2						
Security policies	2			2					
Multiple classrooms	1		1						
Planning / scheduling	1		1						
Mobile EMIs management	0								

B) SURVEY 2: RESULTS

Please state how KAAN system benefits your teaching:

- Students no longer had to share a keyboard and could work independently
- Students could play with another person or as part of a larger group without having to move around the room
- Music files could be shared
- Teachers could monitor students and record their performances
- The new systems appeared to be popular with both teachers and students
- The new systems helped to improve students' compositions
- Instruments were ready-to-use, less liable to be damaged, and used more effectively
- Teachers could use other CAI or audio software with the whole class
- Interaction with students was enhanced and more targeted to individuals or groups, rather than being general to the whole class

Please state the drawbacks and remaining issues of KAAN system:

- Teachers still needed to organise themselves to deliver their lesson efficiently
- The interaction with students using the system was good. However, time was sometimes wasted using the interface. Students then lost focus on the lesson and teachers had to regain control to continue their lesson (although this was an improvement compared with teaching without the system)
- Teachers used and interacted with the system in many different ways, some of which were not anticipated
- Keyboards were used as primary instruments with KAAN system. A multitude of keyboard playing skills was taught, especially with new students (Year 7)

C) INTERVIEWS: REVIEWS

Music Teacher using a Keyboard system to teach music.

- **Why is a Keyboard useful to teach music?**
  - Cheap
  - Lots of extras
  - Backing
  - Chords
  - Loads of voices
  - 2 students / instrument (only instrument allowing that)
  - No formal technique to START playing

- **How effective a tool is a keyboard to teach music?**
  - Effective: allows students to do:
  - Single note melodic melody
  - Chords
  - Harmony
  - Multitrack sequencing
  - Collaboration with somebody else ➔ ensemble technique

- **What aspect of music teaching does the keyboard do well? (not in priority order)**
  - Harmony
  - Selection of voices (auditioning sounds)
  - Visual side: key represents a note ➔ can see when note higher than another

- **What benefits do you see from using a keyboard rather than conventional acoustic instruments?**
  - Headphones (no noise)
  - Sound quality not depending on skill/level

- **What teaching activities are performed on a keyboard?**
  - Melody learning
  - Composing
  - Improvising
  - Some keyboard technique (fingering)
  - Chord / harmony
  - Playing to backings

- **When you have new students learning music with you, what is their knowledge on using keyboards?**
  Variable:
  - Best ones → give a simple melody to start with then give them more difficult ones
  - Others → Differentiation: 3 types:
    i. By outcome: same material to everybody. More able are noticed as they produce more in quantity or in quality
    ii. By Task: (see music matters): same task to everybody. Level assessed against set marks.
    iii. By Resources: different instruments / ability and more or less teacher help.

- **In the scope of your teaching, do you have to teach them how to play an instrument (for instance keyboard)? – In your teaching, do you teach students keyboard techniques right from the start? – How important is teaching keyboard technique? – How much keyboard technique do you teach?**
  Variable:
  - Schools with more able students or private schools will more likely give more emphasis on keyboard technique

  Note: the formal keyboard technique based teaching compared to a music piece based teaching is the same argument as notation compared to reading music

- **For your other teaching activities performed with keyboards, what basic knowledge on playing the keyboard do you require from the students to have?**
  - In theory key stages 1 and 2 should bring music skills and key stage 3 should bring material and repertoire. But as music is considered as optional in KS1/2, the skills have to be (re)learned in KS3.
  - There is no particular need to read/play on keyboard, but some schools teach keyboard methods. National curriculum is flexible.

- **How do you teach them this basic knowledge?**
  Variable:
  - Schools will teach keyboard technique (rhythm and pitch in any order)
  - Knowledge brought with pieces to play (Note: James Bond theme is favourite). For example, teach music structure (binary / ternary) and have a minuet to listen/play → the teacher will show how to play piece / bring side knowledge on how to play it on a keyboard.

- **Do you improve this basic knowledge later to accomplish tougher activities? If yes, what do you teach them?**
  - Depends on students' background and culture. For some, teachers cant teach notation but would emphasise more on pop/dance listening

8.5. ITS System

8.5.1. Tutoring Database System Analysis

This section consisted of:

A) Requirements
B) Top / Down Analysis
C) Database Design – Tables Relationships

A) REQUIREMENTS

CAI SYSTEM:

- Management of Concepts:
  - Concept details
  - Hierarchy of Subconcepts
  - Order of Subconcepts
- Management of Subconcepts:
  - Subconcept details
  - Lessons
  - Tasks
- Management of Lessons and Tasks:
  - Lesson details
  - Task details
  - Parameters
  - Feedbacks

ICAI SYSTEM:

- Management of Students:
  - Student details
  - StudentKnowledge
  - TeachingAction
  - StudentSessions
- Management of StudentKnowledge:
  - Subconcepts / Student
  - Grade history → learning curve / date or / StudentSession
- Management of TeachingActions:
  - TeachingAction details
  - Lesson / Student (task?)
  - Given tasks details
  - Given tasks' answers (used to debug errors?)

B) TOP / DOWN ANALYSIS

ENTITIES:

CAI SYSTEM:

- Concept
- SubConcept
- Lesson
- Task
- Parameter
- Feedback

ICAI SYSTEM:

- Student
- StudentKnowledge
- GradeHistory
- StudentSession
- TeachingActions
- GivenTask
- Answers

ENTITIES RELATIONSHIPS:

CAI SYSTEM:

1	Concept	can contain	MANY	SubConcepts
MANY	**SubConcepts**	**can follow**	**MANY**	**SubConcepts**
1	SubConcept	can contain	MANY	Lessons
1	Task	can be set with	MANY	Lessons
1	Lesson	can have tasks with	MANY	Parameters
1	Task	can generate	MANY	Feedbacks

ICAI SYSTEM:

1	Student	can have	MANY	StudentKnowledge
1	StudentKnowledge	can generate	MANY	GradeHistories
1	Student	can learn in	MANY	StudentSessions
1	StudentSession	can generate	MANY	GradeHistories
1	StudentSession	can be linked with	MANY	TeachingActions
1	Student	can be given	MANY	TeachingActions
1	TeachingAction	can generate	MANY	GivenTasks
1	GivenTask	can get	MANY	Answers
1	Subconcept	can create	MANY	StudentKnowledges
1	Lesson	can generate	MANY	TeachingActions

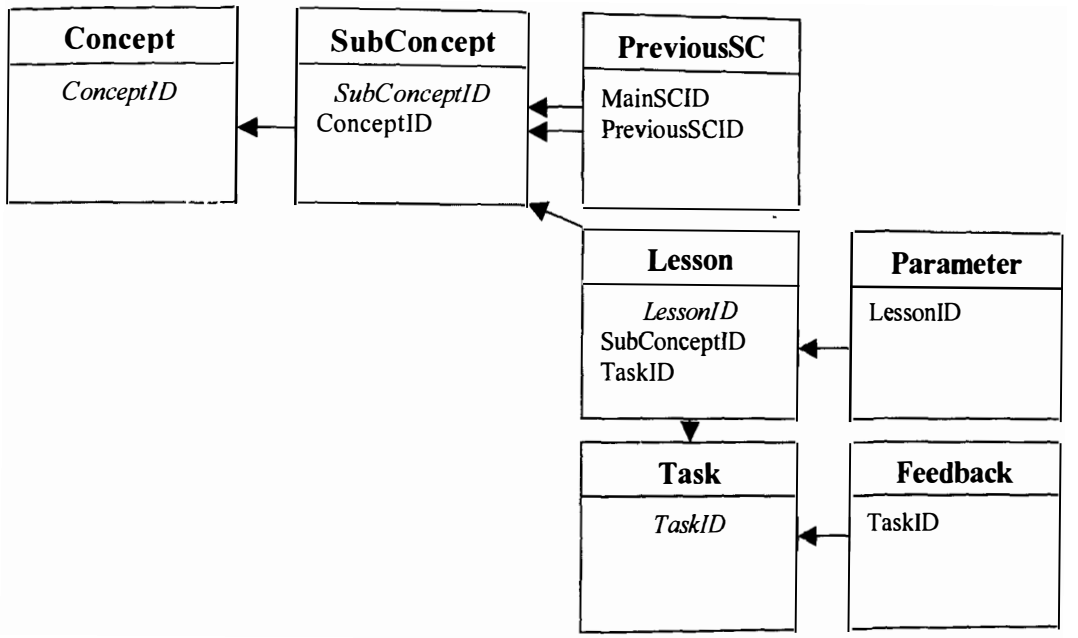
The **MANY to MANY** relationship (in bold) will generate 1 extra table:

- PreviousSC

TABLE RELATIONSHIPS:

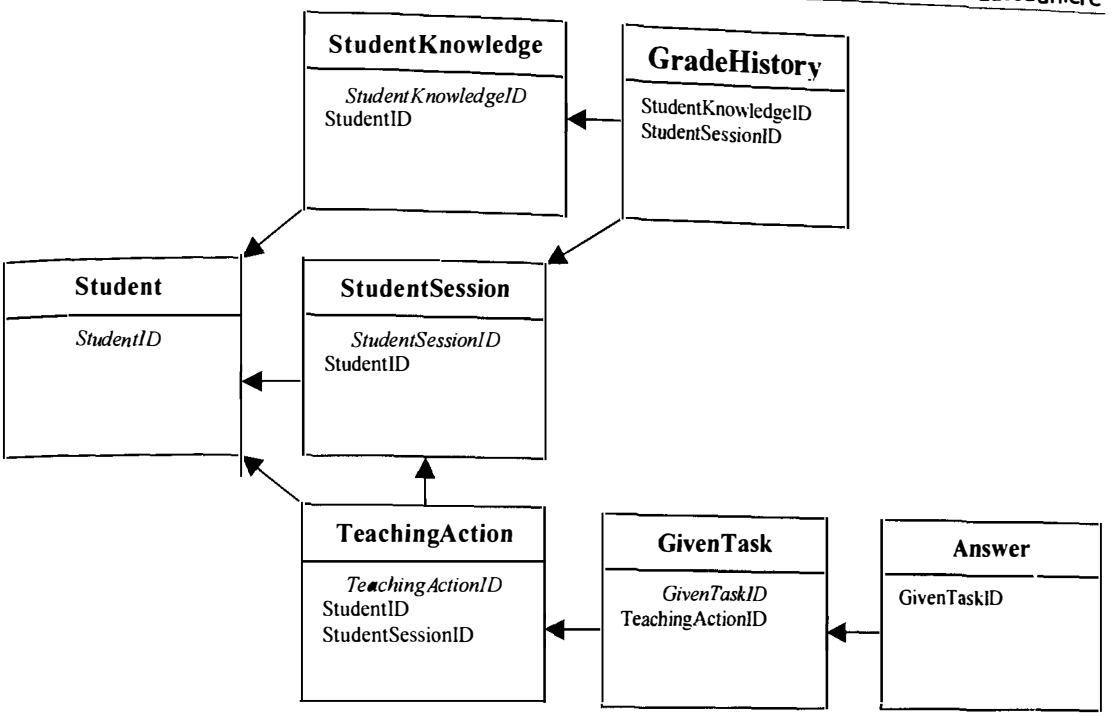
CAI SYSTEM:

• 1	Concept	can contain	MANY	SubConcepts
• **MANY**	**SubConcepts**	**can follow**	**MANY**	**SubConcepts**
• 1	SubConcept	can contain	MANY	Lessons
• 1	Task	can be set with	MANY	Lessons
• 1	Lesson	can have tasks with	MANY	Parameters
• 1	Task	can generate	MANY	Feedbacks

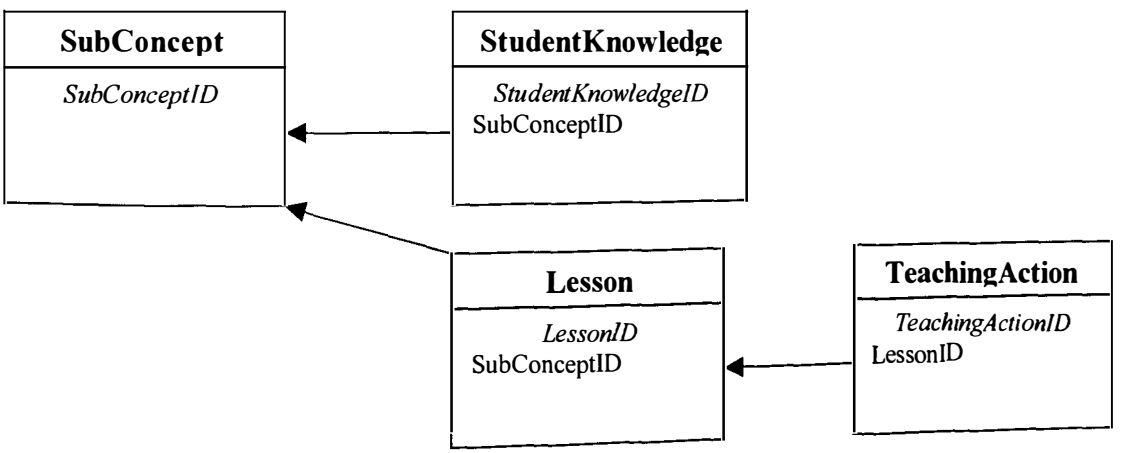


ICAI SYSTEM:

• 1	Student	can have	MANY	StudentKnowledge
• 1	StudentKnowledge	can generate	MANY	GradeHistories
• 1	Student	can learn in	MANY	StudentSessions
• 1	StudentSession	can generate	MANY	GradeHistories
• 1	StudentSession	can be linked with	MANY	TeachingActions
• 1	Student	can be given	MANY	TeachingActions
• 1	TeachingAction	can generate	MANY	GivenTasks
• 1	GivenTask	can get	MANY	Answers



- 1 Subconcept can create MANY StudentKnowledges
- 1 Lesson can generate MANY TeachingActions



ENTITIES ATTRIBUTES:

*: Primary keys.

- Student
  - S_ID* AutoNumber
  - S_Name Text
  - S_Comments Memo
  - S_Station Number
  - S_SpeechSpeed Number
  - SspeechEngine Number
  - S_AutomationTimerTime Number
- StudentKnowledge
  - SK_SC_ID* Number
  - K_S_ID* Number
  - SK_ID AutoNumber

- GradeHistory
  - GH_SK_ID* — Number
  - GH_Grade — Text
  - GH_DateTime* — Date/Time
  - GH_SE_ID — Number
- StudentSession
  - SE_S_ID — Number
  - SE_ID* — AutoNumber
  - SE_Start — Date/Time
  - SE_Stop — Date/Time
- TeachingActions
  - TA_S_ID* — Number
  - TA_L_ID* — Number
  - TA_DateTime* — Date/Time
  - TA_ID — AutoNumber
  - TA_LessonGiven — Boolean
  - TA_SE_ID — Number
- GivenTask
  - GT_ParameterScript — Memo
  - GT_ParameterText — Memo
  - GT_TA_ID — Number
  - GT_ID — AutoNumber
- Answers
  - A_Try* — Number
  - A_Answertext — Text
  - A_AnswerScript — Text
  - A_FeedbackLevel — Text
  - A_FeedbackText — Text
  - A_GT_ID* — Number
- PreviousSC
  - PS_PreviousSCID* — Number
  - PS_MainSCID* — Number

C) DATABASE DESIGN – TABLES RELATIONSHIPS

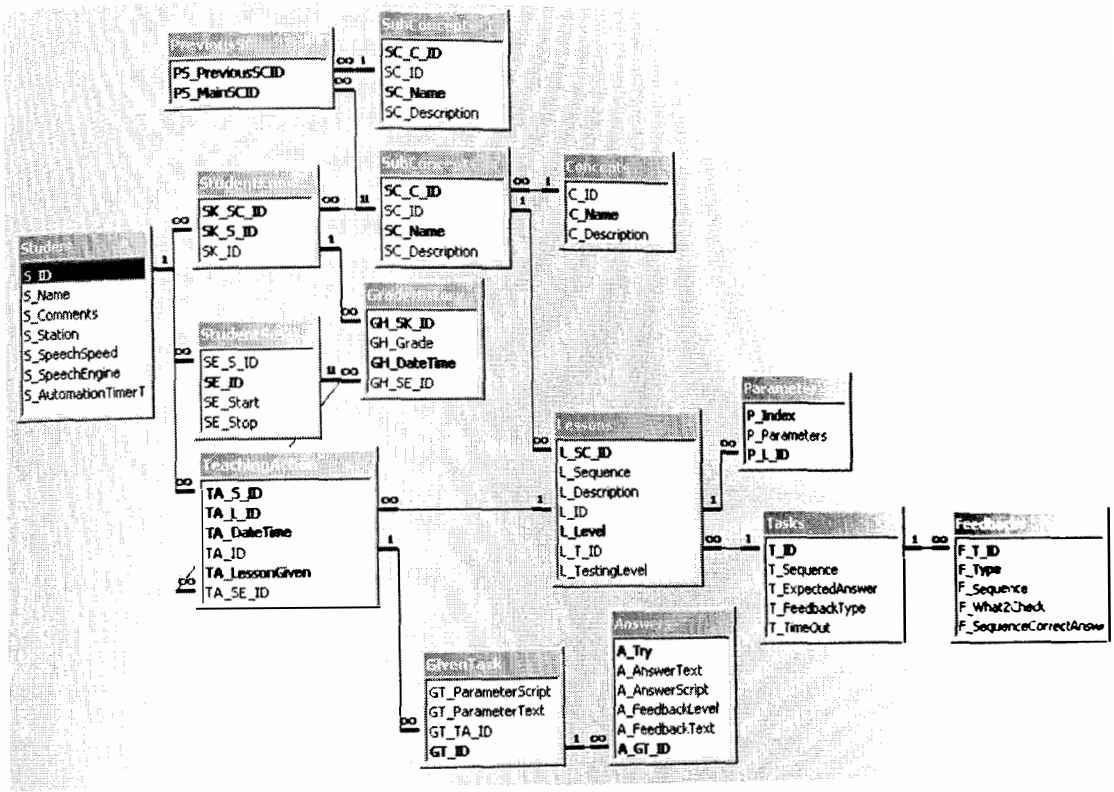


Figure 8-16: Tutoring Database: Tables relationships

8.5.2. Knowledge Elicitation on Music teaching (Results)

description/Required achievement	order	prev C	task group	task group order	tasks	task order	Prev task	param	lesson
student can play any of the given note (natural or altered)	1	-	**Basic notes / white keys**	1	recognise C	1.1.1	-	ANY C	A 'C' note corresponds to the white key just BEFORE a group of 2 black keys
					recognise A and G	1.1.2	-	ANY A or G	Note names go from A to G. A 'A' note corresponds of the white key BETWEEN the second and the third black key of a group of 3 black keys. A 'G' note corresponds of the white key BETWEEN the first and the second black key of a group of 3 black keys.
					recognise any note (except C)	1.1.3	1.1.1 1.1.2	**ANY note from A to G (Except C)**	You know how to play a 'A', a 'C' and a 'G' note. A 'B' note would correspond to the white key BETWEEN a 'A' and a 'C' note, a 'D' note to the white key just after a 'C' key and so on up to a 'G' key. After a 'G' key is a 'A' key.
			ALT.: sharps	2	recognise C#	1.2.1	1.1.3	ANY C#	A sharp is represented with a hash. A sharp note is a note a semitone ABOVE its named note. A semitone is any difference between 2 CONSECUTIVE keys. If a white key is followed by a black key, the difference between the 2 notes is 1 semitone. It is the same with a black key followed by a white key. The difference between 2 white keys separated by a black key is the called a tone. There are 2 exceptions: the difference between a 'E' and a 'F' note (represented with white keys) is a semitone as no black key is between the 2 white keys. It is the same between a 'B' note and a 'C' note. C sharp is therefore the black key ABOVE a 'C' key.
					recognise F#, G#, D#, A#	1.2.2	1.1.3 1.2.1	ANY F#, G#, D#, A#	A sharp note is a note a semitone ABOVE its named note. Therefore a F sharp is the black key just after a 'F' key, a G sharp is the black key just after a 'G' key and so on.
					recognise B# and E#	1.2.3	1.2.1 1.2.2	ANY B# and E#	B sharp and E sharp are exceptions are there is no black key just after a 'B' or a 'E' key. The difference between a 'E' key and a 'F' key or between a 'B' key and a 'C' key is 1 semitone. As a sharp note is a note a semitone ABOVE its named note, B sharp is in fact a 'C' which is represented by a white key. It is the same with E sharp, which corresponds to 'F'.
					recognise any sharp note	1.2.4	1.2.1 1.2.2 1.2.3	ANY flat note	
			ALT.: flats	3	recognise Bb	1.3.1	1.1.3	ANY Bb	A flat is represented with a 'b'. A flat note is a note a semitone BELOW its named note. B flat is therefore the black key BELOW a 'B' key.
					recognise Ab, Db, Eb, Gb	1.3.2	1.1.3 1.3.1	ANY Ab, Db, Eb, Gb	A flat note is a note a semitone BELOW its named note. Therefore a A flat is the black key just after a 'A' key, a D flat is the black key just after a 'D' key and so on.
					recognise Cb and Fb	1.3.3	1.3.1 1.3.2	ANY Cb or Fb	C flat and F flat are exceptions are there is no black key just before a 'C' or a 'F' key. The difference between a 'E' key and a 'F' key or between a 'B' key and a 'C' key is 1 semitone. As a flat note is a note a semitone BELOW its named note, C flat is in fact a 'B' which is represented by a white key. It is the same with F flat, which corresponds to 'E'.
					recognise and flat note	1.3.4	1.3.1 1.3.2 1.3.3	ANY flat note	

description/Required achievement	order	prev C	task group	task group order	tasks	task order	Prev task	param	lesson
			any note	4	recognise any note (without exceptions)	1.4.1	1.1.3 1.2.3 1.3.3	ANY note (without B#, E# and Cb, Fb)	
					recognise any note (with exceptions)	1.4.2	1.1.3 1.2.4 1.3.4	ANY note	
Student can play any chord	2	1	Intervals (3rd + 5th)	1	recognise 3rd note above Note	2.1.1	1.4.2	ANY 3rd above C	The third of a given note is 2 tones above that note. As a reminder a semitone is the difference between 2 consecutive keys and a tone is 2 semitones. Therefore the third of a 'C' note is a 'E' note.
						2.1.2	2.1.1	ANY 3rd above F, G	The third above a 'F' note is a 'A'. The third above a 'G' note is a 'B'
						2.1.3	2.1.2	ANY other 3rd	
					recognise 5th note above Note	2.1.4	2.1.3	ANY 5th above C	The fifth of a given note is 7 semitones above that note, or 3 semitones above the third of the given note. Therefore the fifth of a 'C' note is a 'G' note.
						2.1.5	2.1.4	ANY 5th above F, G	The fifth above a 'F' note is a 'C'. The fifth above a 'G' note is a 'D'
						2.1.6	2.1.5	ANY other 5th	
					recognise any 3rd or 5th	2.1.7	2.1.6	ANY 3rd or 5th	
			Major chords	2	recognise C chord	2.2.1	2.1.7	ANY C chord	A chord is composed of 3 notes played simultaneously. These 3 notes are: the root note, the third and the fifth. The root note is the key of the chord name. As a reminder, the third is 2 tones above the root note and the fifth is 3 semitones above the third. Therefore a 'C' chord is composed of the root note 'C', the third 'E' and the fifth 'G', which gives 'C', 'E' and 'G'.
					recognise G and F chords	2.2.2	2.2.1	ANY G or F chord	The root note of a 'G' chord is 'G', its third is 'B' and fifth is 'D', which gives 'G', 'B' and 'D' for a 'G' chord. The root note of a 'F' chord is 'F', its third is 'A' and fifth is 'C', which gives 'F', 'A' and 'C' for a 'F' chord.
					recognise A, B, D, E chords	2.2.3	2.2.2	ANY A, B, D,E chord	The root note is the chord name, the third is 2 tones above the root note. The fifth is 3 semitones above the third. The third and fifth can be sharp notes.
					recognise any UNALTERED Major chord	2.2.4	2.2.3	ANY A, B, C, D, E, F, G chord	
					recognise F# and Bb chords	2.2.5	2.2.4	ANY F# or Bb chord	For a F sharp chord, either recover each note one by one (the root note is 'F sharp', its third is 'A sharp' and fifth is 'C sharp'), or recover a 'F' chord and INCREASE every note of this 'F' chord of 1 semitone. For a B flat chord, either recover each note one by one (the root note is 'B flat', its third is 'D' and fifth is 'F'), or recover a 'B' chord and DECREASE every note of this 'B' chord of 1 semitone.
					recognise any ALTERED Major chord	2.2.6	2.2.5	ANY A#, B#, C#, D#, E#, F#, G# and Ab, Bb, Cb, Db, Eb, Fb, Gb chord	Either recover each note one by one, or recover the unaltered chord and INCREASE every note of the recovered chord for a sharp root note or DECREASE every note for a flat root note.

description/Required achievement	order	prev C	task group	task group order	tasks	task order	Prev task	param	lesson
					recognise any Major chord	2 2 7	2 2 6	ANY major chord	
			Minor Intervals (3rd + 5th)	3	recognise minor 3rd note above Note	2 3 1	2 1 7 2 2 6	ANY minor 3rd above C	The minor third of a given note is a normal (or major) third decrease by 1 semitone. which is 3 semitones above the given note. Therefore the minor third of a 'C' note is a 'E' note decreased of 1 semitone. which gives 'E flat'
						2 3 2		ANY minor 3rd above F. G	The minor third above a 'F' note is a 'A flat'. The third above a 'G' note is a 'B flat'
						2.3.3		ANY other minor 3rd	
					recognise 5th note above Note	2.3.4	2.1.3	ANYminor 5th above C	The minor fifth of a given note is 1 semitone below the major 5th of that note, which is therfore 6 semitones above the given note. Therefore the minor fifth of a 'C' note is a 'G flat'.
						2.3.5	2.1.4	ANY minor 5th above F, G	The minor fifth above a 'F' note is a 'C flat' or 'B'. The minor fifth above a 'G' note is a 'D flat'
						2.3.6	2.1.5	ANY other minor 5th	
					recognise any 3rd or 5th	2.3.7	2.1.6	ANY minor 3rd or minor 5th	
			Minor chords	4	recognise Am chord	2.4.1	2.3.7	Any Am chord	A minor chord is a major chord, which third is a minor third. It is notated like a major chord with a little 'm' or '-' for minor added to the chord name. To recover a minor chord, either find the root note, then the minored third (now 3 semitones above the root note) and the major fifth. Or recover the major chord, find the third and decrease it by 1 semitone. For a 'A minor', the root note is 'A', the minored third is now 'C' instead of 'C sharp' and the fifth is still 'E', which gives 'A', 'C' and 'E' for chord 'A minor'.
					recognise Em and Dm chords	2.4.2	2.4.1	Any Em and Dm chord	For 'Em', the root note is 'E', the minored third is 'G' (instead of 'G sharp') and the fifith is till 'B', which gives 'E', 'G' and 'B' for chord 'Em'. For 'Dm', the root note is 'D', the minored third is 'F' (instead of 'F sharp') and the fifith is still 'A', which gives 'D', 'F' and 'A' for chord 'Dm'.
					recognise Bm, Cm, Fm, Gm chords	2.4.3	2.4.2	Any Bm, Cm, Fm, Gm chord	To recover a minor chord, either find the root note, then the minored third (now 3 semitones above the root note) and the fifth (now 4 semitones above the minored third). Or recover the major chord, find the third and decrease it of 1 semitone.The third and/or fifth could be sharp notes.
					recognise any UNALTERED Minor chord	2.4.4	2.4.3	ANY Am, Bm, Cm, Dm, Em, Fm, Gm chord	
					recognise F#m and Bbm chords	2.4.5	2.4.4	ANY F#m or Bbm chord	For a F sharp minor chord, either recover each note one by one (the root note is 'F sharp', its minored third is 'A' and fifth is 'C sharp'), or recover a 'F minor' chord and INCREASE every note of this 'F minor' chord of 1 semitone. For a B flat minor chord, either recover each note one by one (the root note is 'B flat', its minored third is 'D flat or C sharp' and fifth is 'F'), or recover a 'B minor' chord and DECREASE every note of this 'B' chord of 1 semitone.

description/Required achievement	order	prev C	task group	task group order	tasks	task order	Prev task	param	lesson
					recognise any ALTERED Minor chord	2.4.6	2.4.5	ANY A#m, B#m, C#m, D#m, E#m, F#m, G#m and Abm, Bbm, Cbm, Dbm, Ebm, Fbm, Gbm chord	Either recover each note one by one, or recover the unaltered minor chord and INCREASE every note of the recovered chord for a sharp root note or DECREASE every note for a flat root note.
					recognise any Minor chord	2.4.7	2.4.6	Any MINOR chord	
			any major/minor chord	5	recognise any minor or major chord	2.5.1	2.4.7	Any maj/min chord	
			Inversions	6	recognise 1st inversion and root position	2.6.1	2.5.1	ANY 1st inversion	The chords you know comprise 3 notes and therefore can be played in 3 different ways called inversions. An inversion determine which of the notes of the chord would be the lowest. The chords you learnt to play were learnt in their root position, that means that the root note was the lowest of the 3. A first inversion consists in playing the third first, then the fifth followed by the root note. For example a 'C' chord in its root position is 'C', 'E' and 'G'. Its first inversion is 'E', 'G' and then 'C'.
					recognise 2st second inversion	2.6.2	2.6.1	ANY 2nd inversion	A first inversion consists in playing the third first. A second inversion consists in playing the fifth first, then the root note followed by the third. For example a 'C' chord in its root position is 'C', 'E' and 'G'. Its first inversion is 'E', 'G' and then 'C' and its second inversion is 'G', 'C' and 'E'.
						2.6.3	2.6.2	ANY inversion	
			7th	7	recognise C7 chord	2.7.1	2.5.1 2.6.3	ANY C7 chord	A chord with 7th is a chord with 4 notes. The chord is notated as usual with a '7' added at the end. The chord is composed of the usual 3 notes of the major or minor chord plus the 7th. The root note is in fact the 8th of the chord, which means that the 7th is 1 tone below the root note. For a 'C7' chord, the chord is composed of the 3 notes of the 'C' chord, which are 'C', 'E' and 'G', plus the 7th, which is 1 tone below the root note 'C' and is equal to 'B flat'. This gives for chord 'C7': 'C', 'E', 'G' and 'B flat'.
					recognise F#7 and Bb7 chords	2.7.2	2.7.1	ANY F#7 or Bb7 chord	For a F sharp 7 chord, either recover each note one by one (the root note is 'F sharp', its third is 'A sharp', fifth is 'C sharp' and 7th is 'E'), or recover a 'F7' chord and INCREASE every note of this 'F7' chord of 1 semitone. For a B flat 7 chord, either recover each note one by one (the root note is 'B flat', its third is 'D', fifth is 'F' and 7th is 'A flat or 'G sharp'), or recover a 'B7' chord and DECREASE every note of this 'B7' chord of 1 semitone.
					recognise any Major chord with 7th	2.7.3	2.7.2	ANY Major chord with 7th	
					recognise Am7 chord	2.7.4	2.7.3	ANY Am7 chord	A 'A minor 7' chord, the chord is composed of the 3 notes of the 'A minor' chord, which are 'A', 'C' and 'E', plus the 7th, which is 1 tone below the root note 'A' and is equal to 'G'. This gives for chord 'A minor 7': 'A', 'C', 'E' and 'G'.

description/Required achievement	order	prev C	task group	task group order	tasks	task order	Prev task	param	lesson
					recognise F#m7 and Bbm7 chords	2.7.5	2.7.4	Any F#m7 and Bbm7 chord	For a F sharp minor 7 chord, either recover each note one by one (the root note is 'F sharp', its minored third is 'A', fifth is 'C sharp' and 7th is 'E'), or recover a 'F minor 7' chord and INCREASE every note of this 'F minor 7' chord of 1 semitone. For a B flat minor 7 chord, either recover each note one by one (the root note is 'B flat', its minored third is 'D flat or C sharp', fifth is 'F' and 7th is 'A flat or 'G sharp'), or recover a 'B minor 7' chord and DECREASE every note of this 'B minor 7' chord of 1 semitone
					recognise any Minor chord with 7th	2.7.6	2.7.5	Any minor chord with 7th	
					recognise any chord with 7th	2.7.7	2.7.6	any chord with 7th	
			any chord	8	recognise any taught chord	2.8.1	2.7.6	any taught chord	

Table 8-6: Music Lessons and Concepts from Knowledge Elicitation

8.5.3. Results

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
1	1	24/01/03	A1	10'					-Lessons to be revised -Use other speech engine or use phonetic enhancement -Ability to answer before the end of task speech
2	2	28/02/03	B1	5'					-Bugs and Crashes to be solved
Changes: ITS5 -Debugging (function keys sometimes taken into account as responses) -speech speed available and saved againt a student -phonetic: wait for Duncan to give me a final version of the database									
3	3	28/02/03	C1	53'	-Names of notes(1) -Semitones/tones -sharp -flat keys (half through)	-Could not understand Voice -too fast -did not tell me about notes at the start --> went straight to Semitones(1)	-Should tell you what is coming next -Should tell me when I could go to the next stage -Needs an explanantion early! -Get the explanations in sequence with the light (light first then talk then light off) -tell me that the note will light up (and what that means)	-Lights were good -Retry or repeat button (test)	-Needs an introduction to how it will all work -Intro to each section (NB: this may may be really required at lesson 1) (1): Note names were written on the left side of the keyboard. That is why he passed through white notes easily in the knwoledge assessment. YC learnt to find notes because he knew where 'c' was + then YC looked ref to 'c'. -Briefing is little long -Shows control keys and where to avoid -Needs to tell you what you need to do + what is going on
4	3	28/02/03	D1	8'	Nothing(1)	-Goes though it once - needs repetition + explanations -Voice is difficult to understand	-Repeat explanations(2)	-keys lighting up	(1): Alex thinks he learnt the white keys and then it became too complicated with the semitone lesson. He failed once or twice --> gave up (2): When the system thanks it is time to move on it needs to say 'well done. You know ... 'Would you like to go further or to revise about what you have done.' -Needs confidence building -Needs to be given choice to revise (lesson for UC tests only played for revision)

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
5	3	28/02/03	A2	30'					-start is too complicated -Needs a question 'do you want to continue?' and to show a light -People tend to press any key once things delay too long -Needs an intro to say 'we will check what you understand, etc.' -Can't understand English from computer -For testing, we need to be able to hear what is going on -Needs to be shown a scale right at the beginning and tell 'we are going to teach you...' in first lesson -Need to look at phoetic spelling of words to force the computer to say things correctly
	Changes: ITS5-->ITS6 -New Intro and transitions (Scale at beginning) -speech engine fixes: waiting for Duncan -prepare function key labels and hide note names on kbd								
6	4	03/03/03	A3	3'					-Kaan Bug still there -Rename New test to Continue Training -After test feedback offer next actions
	Changes: ITS6 -renamed new test to Continue (training) -after test transition								
7	#	05/03/03	E1	#	-first lesson should start with C -chords section needs more explanations -7ths should be referred to as intervals notes (1 note below an octave) -slower speech -Tasks need to be more musical -Some chords lessons are too difficult to introduce at this stage (leave them out) ALSO major intervals and chords not progressive enough -Could you introduce playing melodies as a test? Primary chords / key OR Teacher records in MIDI a simple 'perfect' melody that is played to student ==> autotest their performance (accuracy and notes) -remove tasks' 'note' -lesson/task transition: play it only once or twice a session -change script for text/note seq -2 lessons for semitone (just change the range for the tests) -C flat and Fflat (E sharp and B sharp) can be left for afterwards (keys)				
	changes: ITS6 -database scripts corrected (transitions and intros to debug) -lessons order changed (+ some ditched and some added) -after test transition changed								

S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
5 8	07/03/03	C2	70'	-fifth -chord -F# chord	-Time to answer to short (add 5 to 10 sec) -Transitions to be smaller (boring) -lesson tasks (not necessary lesson type2) -pronounciation fingering needed for chords -way to answer chords missing	-Way to answer chords + fingering (1 off) -pronounciation -retry count OK only first time OK for the test	-Transitions and directives -function key lighting -Instead of Transition, have system do it for you automatically (if no answer for n sec:give next question)	-piece of paperused as notes -did forget:semitones/notes -KA: not very good:feedback what level student is the same for lesson tasks and SC/C tasks: feedback new level Notes: -remove Cb, E# from tasks -bug for any notes lesson (had to stop and restart) -replay lesson has no transition, retry did not work -have channel 4 no volume work better ->=2 param tests: long to get approved (removed followinggoodtries) -use of retry to count correct tries to be rethought -change keylighting interaction with text for intervals. do |N-C|T-C|W-.5|T-and its third E|N-E|
5 9	07/03/03	F1	60'	-third / fifth -concepts in English	-boring and lost tim with transitions: lost concentration (have only keys lighting upto do transition + auto goto next test) -"play twice if not captured": play only a few times -time was too much of a constraint	-pronounciation -fingering -button not to repeat -repeat lesson option before testing	-learnt new things and in different ways -refresh memory	-Concept transition: key left lighting -Chord response problems: removed key sense Notes: -been given system intro -pb with first function key transition (about test keys) -C transition much too long (could be in individual differences) -if give no answer --> auto retry -use habit capturing -too slow for chris too attain correct level -test giving/parameters: collection of parameters decreased after new test not to give same param, refilled if empty and still not OK) -students have to be more concentrated on teaching rather than on function keys -some any.. SC/C to be ditched? -simultaneous chords played in seq: one at a time -pb with chris: learnt music in French and other way: changed his habits -Chord major 6 lesson is strange -Chord answering problems

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
Changes: ITS6-->ITS7 -removed E#, Cb etc from param and tasks -KaanKbd/Seq/buildseq: option to add or not allnotesoff at the end -retry/new test triesOK count -param giving -replay transition -transition cganges -database changes and added individual speech fields -bug fixed with full K downgrade -real time scripting with 'a' replacing by 'an' -stop bug fixed -K update restored for lessons (seen) -tranistion between lessons and tests -time to respond changed -timer to use function keys with auto stuff -Added new voices									
9	6	13/03/03	C3	5'		-Explanations too fast -Still not clear phonetically (with new voice) -Still no taught fingering -too many bugs		-not boring any more	
Changes: ITS7 -Added script header S- to change speech speed relatively or absolutely -Added fingering lesson as part of 1st lesson on chords -Corrected Learning curve bug									

S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
7	14/03/03	F2	67'	-Minor chords -Inversions -7th	-Do not know when it is a lesson, test or whatever -Timer to respond (function keys) need to be set against user (and have it enabled or not) -Difficult to Remember chord names for tasks --> do cycle as normal, if no answer 5sec before the end repeat task	-Add a pause button for a break -New voice has no rhythm, needs to be more dynamic -I feel you understand THIS lesson, if SC test --> tell let's do an overall test on the concept	-Huge improvement with function keys --> not boring -To show and practice is good as well as repetition	Notes -Chord term new for Chris -Given Gb chord failed in KA -First transition test --> new lesson not answered: automatically jumped to next lesson -'Crashed' when clicked on next training and finger slipped to nex lesson -Use of Chord Major (4) and (7)? Transition to next lesson--->Offered to replay lesson but there was none (just for revising). Also if there is only one option: do it -Interval Minor: BIG CRASH NEW SESSION: answered OK minor chord (not seen) but failed with inversions. Note: no SC test for minor interval -Bug: retry test allowed after replaying lesson just after played first time. -Interval Lesson: add root position = inversion 0 -Chris had problems with inversions (to understand the concept) -Don't offer autoretry if OK -Allow plenty of time to respond from that lesson -Inversion tasks are difficult to remember (1: understand chord name, 2: remember inversion) -Inversion 3: add any inversion (only offered 2nd inversion). IN GENERAL, review tasks and lessons to have generalisation coming quicker -Difference between Inversion 3 and 4? -Add in intro to keep key pressed until system responds (take lot of time to recognise chords, uses lot of CPU) -Remove parenthesis from lesson scripts -Use of 7th (3)? -Start of 7th (4)? -Introduce Stop / Pause button at intro (have a play button as well (same button or another) -Tasks given with sometimes no space between param such as minor7 -Not enough time to respond at the end -use of three last 7th? -add a transition after the last lesson
10								

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
11	7	15/03/03	G1	27'	-White notes -Sharps and flats -Chords	-Phonetic problems between A and E -nothing to repeat task	-spoken 'Hand' in fingerings lesson -Voice could be better -Gives the same exercise sometimes (looks like a bug)	-Lesson progression -Can do more training if necessary -Lessosn are short so there is no problem to replay them and they are easier to assimilate	Notes: -KA failed at semitones -First time asked for training: did not answer --> did it automatically: need to set timer longer??? -Peggy waited for some speech for the first transition without it -Bug in lesson script: F, F+F#, F# -SC Lessons go directly to test -CRASHED AT FLATS: function key pressed just after having it automated NEW SESSION: KA: semitones, sharps and flat OK. Failed at chords -problems if test was not understood: needed to be repeated -Went to chord without having been told about third and fifth--> need SC test there -Replay lesson with no first test: offers retry test (none given) -Did not speed down for Major chord (2) -Major (4): no lesson but offered to replay lesson

changes: ITS7
-Pause/Play function key (changed label on kbds, + intro modified for it + added a t ansition when paused)
-Increased time 2 respond for chords and inversions
-Removed parentheses from lesson scripts
-Last lesson transition
-fixed crashed when selected function key during a function key automation
-Added Automation Timer settings
-Replay lesson does not allow retry
-Replay lesson changed to just Replay to allow also replaying tasks orders
-Transition changed for SC/C lessons with auto goto test and some speech
-Lesson report

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
12	8	16/03/03	G2	25'					Notes: -KA started with Chord Major and then direct to Chord Minor. Failed and went back to Minor third: failed -Replay used in KA TBC -Parenthesis to remove in Interval(1) -Peggy did not have a lesson about 3rd and 5th --> problems to understand about 3rd minor -Relate intervals to chords and semitones/tones to intervals -Crash with pause on Chord minor NEW SESSION: -KA started with tones and semitones!!!! -Replay does not always give same test -Problems with voice for A/E -Problems for P. to understand or remember lessons + a bit lost with first transition with no speech -Crash @ sharp (3) paused and played back (debug bug appeared) -Doesn't put back play lesson into right mode
Changes: ITS7 -new pause setting: now resets correctly parameters -bug about intervals (not being done): changed tones and semitones to SC and minor interval to C -pb with chords CRASHING: player accepted col with >= nb of expected notes but not test object: changed -pb with chords NOT accepting easily new chords: happened because for some reasons the collection given by transserver had not correct number of notes: now have it filtered by kaankbd and read its current noteon col -kaankbds now displays ok notes on and off									
13	9	21/03/03	H1	15'	Notes C, A and G	-Crash -expected longer lessons -voice was distracting	-add a 'no answer' button	-did what it was expected to do: teach	Notes: -did not know answer in KA and waited for feedback ==> needed a 'no answer' response -crashed with lesson and keylighting with stop key -surprised to have no speech at the first transition with no speech
14	9	21/03/03	I1	31'	-notes -sharps and flats -third and fifth	-voice		-keys lighting up -go over again ==>makes it easy	-KA: problems with semitones/tones -lesson: a few mistakes ==> replayed lesson -Bug (kreeze) when pressing to function keys at a time -interval major (1) too fast -interval major (3,6) :no lesson (revisions?) asked for transition with no speech (+offers to play lesson)
15	9	21/03/03	J1	27'	-sharp chords	-words she did not know or understand -chord playing	-add gaps between concepts in lesson so that students have time to assimilate	-replay / retry -shows corrections -key lighting	-speeds up at interval majar and minor (1) -check script for chord minor (change - with minus and add a space on (3) after last but 1)

S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
9	21/03/03	A4	28'	-fingering for a chord -chord with root as b or # -chord with 3 notes	-voice needs gap between sentences + difficulty to understand 'E' 'four given', 'chord' -revise / replay	-voice -key lighting feedback at the beginning add 'please look at the keyboard'	-function keys title easy and stays lit on -more verbal feedback + 'well done' + 'I feel you understand the lesson'	-what happens if play a 2 note chords -interesting to put F# and Gb chords -KA failed at chord major(7) played 2 notes instead of 3 -problems to understand difference between new test and retry -problems to answer chords -played notes that were not function keys when waiting for function key switches them off -after a revision, just been offerred new test/replay -replay / revise needed on label -new session: minor third: thought has just been asked to naswer a chord
10	09/04/03	B2	45'	-interval: semitones, third and fifth -chords	-does not capture all the time response -voice asked twice to press some keys	-fingering checking?	-did not crash -very easy, clear -checking concept is OK: good not sure yourself but feel confident the systems knows for you -KA is good to check and revise -lights -well done and encouragements	-Answered by chance note name and fast then failed at semitones (but ok at lesson) -voice prolems with A and E -bug: check post notes / waiting for function key awaited -did not have instruction at beginning (told him about replay test (used it a lot) -for A flat/sharp use 'A' or Ah -answered very quickly (had to re-answer many times) -good to have following tests like F# and Gb -replayed three times first lesson for third and fifth -allow SC lesson replay if has been replayed at least once earlier -speed decrease too noticeable -change text of Major(6)
10	09/04/03	D2	25'	-white keys	-voice -lighting keys with sun	-ability to train more to have it instinctive (instead of counting)	-functions keys to go forward / backward -easier interface with system: have control of learning	-had to remove buggy check for awaited fct key as student kept pressing on bad keys, which froze system -voice changed -did not understand use of new / retry test and next lesson -pressed function keys before being asked for -trained a lot on white (3), few problems between G and F because from A upwards
11	23/04/03	K1	20'	-lots	-repetition of transitions		-Level of lesson -Actual steps -testing	-too fast for system -pause lesson:system froze -asked for chord and answered 2 notes: problem

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
20	11	23/04/03	L1	34'	-keys -intervals -chords	-not checking mistakes / bugs -voice	-3rd / 5th show in lesson counting up with key lighting	-Taught quickly -Small bits of knowledge to practice on	KA failed at semitones, but knew about notes (ref from C) -fast learner -replayed Major(1) -problems with major(2): always same error: counted 2 semitones instead of 2 tones -Movement of hands showed that he tries to find 3rd and 5th backward -spent a long time on Maj(6)
21	11	23/04/03	M1	14'	-about herself	-scared from technmology and system -thought she was not good enough to learn with it			-No knowledge on Music -Changed voice on-line, crashed and did not have first session intro -was very stressed and listened more than watched -KA: notes OK as was asked to play a C and answered a B (System fedback it was a B) and then asked to play a B -replayed semitones(1)
22	11	23/04/03	N1	19'	-white notes	-could not understand semitone lesson -too fast, too general does not explain in details -voice	-function keys	-needed longer to learn more	-Do not understand principles of system at beginning (Changed voice on-line, crashed and did not have first session intro, but understoof while playing) -guessed answers from semitones then replayed lesson
23	11	23/04/03	O1	34'	-knew all before but reminded -semitones/tones	-could not check fingering -chord capture	-Should show what was played and bad then what is correct -lesson on fifth, if played a third do not feedback wrong but (you played a third) -retry not necessary with tests with only 1 param	-correct answer feedback -lighting up -time to look at what is wrong	KA: failed at major interval -MI(1): remove 'notes' -MI(3,6)+CM(4): bug: no lesson to play: offers test and lesson with no lesson -Played his chords with left hand -problems to play Bb chord
24	11	23/04/03	A5	67'	-altered chords -minor chords -7th chords -minor 7th chords + altered	-SC/C tests with no lesson: no transition and do nearly all the same==>use? -chord capturing	-have offline training (use pause button) during a practice session OFFERED by system so that students can train freely. Have a timer to get studenst back on to lesson every 5-10 minutes	-feedback(well done is very positive especially when difficult: boosts confidence) -pressured and challenged: hard but still wanted to carry on -function keys are easy when understood -play and say instead of say and play	-KA: failed first at Int Min, but ok with Chord maj -IM(2) lesson: say Minor third of G -Cmi(3) lesson: check .The -Cmi(3,4) offers lesson even with none -Cmi(6): to get an altered minor chord... (also with major chord) -Cmi(7) is different from Cmi(6) because it offers major chords as well -Chord inversion(3): test have root pos, first inv and sec inv -in inversion tests: hard to catch everything -fingering with 7th

	S	Date	Ref	Duration	What have you learnt	What was a problem	How could it be improved	What was good or worked well	Anything else
25	12	25/04/03	P1	10'	-semitones but forgot (needed to replay lesson)	-need to give you an advice on next action to do with function keys		-speech and keylighting -good explanation at beginning	KA: failed on semitones (but has some Knowledge)
26	12	25/04/03	P2	10'	-passed through semitones				KA: failed on semitones but ok with whites ==> started at semitones -problems with voice A and E -very fast to answer sharps: knew it? -used to the way it is teaching
27	12	25/04/03	Q1	23'	-notes	-felt the system hated him! -semitones lessons too long	-voice (got stressed as needed to guess and answered badly)	-did teach me where notes were	-No knowledge on music -pressed on base note and wanted to press notes until found good one -problems with semitones(2) and no way to get lesson...lost motivation -NOTES: if can sit there without getting stressed, then it will teach you something. Doing it by myself would have mad it less stressful (perhaps with headphones). Did not understand semitone lesson
28	12	25/04/03	R1	38'	-chords	-voice is annoying -fingering hard to try==> used own fingering		-repeat as you learn -appraisal -brings you to next lesson when needed -enjoyed it	KA: failed at semitones -use of first lesson/test? -check it feedbacks in b when task in b -watching hand on fifth showed counted number of white keys instead of semitones -always have lesson replay -waiting for action after fingering -feedback correct chord need to stay longer -played chords with fingers 2,3,4 -for F# and Bb chords used method: F then +1 semitone and ...
29	12	25/04/03	S1	31'	-intervals	-too much talking -too long		-teach and do -clear	-warning with setting as he was much smaller -KA failed at int minor -had problems understanding function keys -I think he knows about chords but not as taught here. Then some lessons will be new others boring.

REFERENCES

Ackroyd S., Hugues J.A. (1981). *Aspect of Modern Sociology: data collection in context.* Longman inc (ISBN: 0-582-48015-9)

AECT, (2001). *Intelligent Tutoring Systems Defined.* Retrieved 22/06/2003 from AECT, Web site: www.aect.org/intranet/publications/edtech/19/19-03.html

Altman Klein H., Vincent E.J., Isaacson J.J. (1998). *From managing the car to managing the road: the development of driving skills.* In Proceedings of the human factors and ergonomics society (volume 2, pp 1271-1275, ISSN: 0163-5182)

Andre B.A. (1997). *Individualizing Computerized Instruction Based on Learning Styles and Strategies.* Thesis retrieved 22/06/2003 from University of Texas Austin, Web site: www.edb.utexas.edu/mmresearch/students97/Andre/

Aoyagi T., Hirata K. (1992). *Music server system distributed music system on local area network.* In Journal of information processing (volume 15 number 1, pp 1-9)

Atolagbe T.A., Hlupic V. (1997). *SimTutor: A multimedia intelligent tutoring system for simulation modelling.* In Proceedings of the 1997 Winter Simulation conference (pp 504-509, ISBN: 0-7803-4278-X)

Baecker R.M., Buxton W.A.S. (1987). *Readings in Human-computer interaction: a multidisciplinary approach.* Morgan Kaufmann publishers inc, (ISBN 0-934613-24-9)

Bierman D.J. (1988). *'Intelligent' Authoring Systems*: towards *better Courseware.* Retrieved 22/06/2003 from web site: a1162.fmg.uva.nl/~djb/publications/ 1988/itssoviet88.pdf

Besanet. (2002). *ICT 2002: Ch5: Financing ICT.* Retrieved 22/06/2003 from Besanet, Web site: http://www.besanet.org.uk/ict2002/summary/s-ch5.pdf

British Department for Education and Employment. (n.d.). *The National Curriculum for England.* Retrieved 22/06/2003, from National Curriculum, Web sites: http://www.nc.uk.net/download/nc_download.html (to download the document) or http://www.nc.uk.net/Mu-3--POS.html (for details)

Brown. S. (1999). *Institutional Strategies for Assessment.* In Assessment Maters in Higher Education: Choosing and Using Diverse Approaches [S. Brown. and A. Glasner. (eds), SRHE & Open University Press, pp 3-13]

Buick P., Lennard V. (1995). *Music Technology, Reference Book.* PC Publishing (pp 1-3, ISBN: 1-870775-34-1)

Butz B.P. (June 1999). *Freedom of Choice in an Intelligent Tutoring System.* In ASEE Annual Conference & Exposition. Retrieved 22/06/2003 from ASEE Web site: http://www.asee.org/conferences/search/99conf229.PDF

Butz B.P. (Oct 2000). *The learning mechanism of the interactive multimedia intelligent tutoring system (IMITS).* In 30th ASEE/IEEE frontiers in Education conference, IEEE (pp T3D-11 to T3D-16, ISBN: 0-7803-6424-4)

Buxton H., Howell J., Sage K. (n.d.). *EU ActIPret project at COGS.* Retrieved 22/06/2003 from COGS (Sussex), Web site: http://www.cogs.susx.ac.uk/lab/vision/actipret/

Capita Education Services. (n.d.). *SIMS (Data Exchange).* Retrieved 22/06/2003 from Capita Education Services, Web site: www.capitaes.co.uk/capitaesdotco/ (section 'Data Exchange')

Casio Corporation. (n.d.). *CTK series portable keyboards*, Retrieved 22/06/2003 from Casio Corporation, Web site: http://www.casio.co.uk/music/keyboards/ (section CTK series)

Cavallo D., Papert S., Basu A., Blikstein P. and Sipitakiat A. (n.d.1). *1-Teacher Schools*, Retrieved 22/06/2003 from MIT Media Lab, Web site: http://www.media.mit.edu/research/ResearchPubWeb.pl?ID=22

Cavallo D., Mikhak B., Papert S. and Resnick M. (n.d.2). *Constructionism*, Retrieved 22/06/2003 from MIT Media Lab, Web site: http://www.media.mit.edu/research/ResearchPubWeb.pl?ID=22

Chan M., Hariton C., Rigeard P., Campo E. (1995). *Smart house automation system for the elderly and the disabled.* In Proceedings of the IEEE International Conference on systems, man and cybernetics (part 2, pp 1586-1589, ISSN: 0884-3627)

Compaq-Hewlett Packard. (n.d.). *Handheld Devices*, Retrieved 22/06/2003 from Hewlett Packard, Web site: http://welcome.hp.com/country/us/eng/prodserv.html (Handheld Devices section)

Corker K.M., Pisanich G. (1998). *Cognitive performance for multiple operators in complex dynamic airspace systems: computational representation and empirical analyses.* In Proceedings of the human factors and ergonomics society (volume 1, pp 341-345, ISSN: 0163-5182)

Counterpoint MTC Ltd. (n.d.1). *MTS-1 Keyboard System.* Retrieved 22/06/2003 from Counterpoint MTC Ltd, Web site: http://www.cmtc.co.uk/musicstore/keyboard_systems.htm (section MTS-1)

Counterpoint MTC Ltd. (n.d.2). *Matrix 32 System.* Retrieved 22/06/2003 from Counterpoint MTC Ltd, Web site: http://www.cmtc.co.uk/musicstore/keyboard_systems.htm (section Matrix 32)

Counterpoint MTC Ltd. (n.d.3). *Product Index.* Retrieved 22/06/2003 from Counterpoint MTC Ltd, Web site: http://www.cmtc.co.uk/musicstore/productindex_new.htm

Crystal Decisions (n.d.). *Crystal Reports 9*, Retrieved 26/06/2003 from Web site: www.crystaldecisions.com/products/crystalreports/default.asp

Debar H., Becker M., Siboni D. (1992) *A neural network component for an intrusion detection system.* In Proceedings of the symposium on security and privacy (pp240-250, ISBN: 0-8186-2825)

Diaper D. (1989). *Knowledge elicitation: principles, techniques and applications* (ISBN: 0-7458-0451-9)

Dick D. (2002). *The P.C. Support Handbook.* Dumbreck publishing. (ISBN: 0-9541711-0-1)

Dickman A. (1995). *Two-Tier versus Three-Tier Apps.*, Retrieved 22/06/2003 from InformationWeek, Web site: http://www.informationweek.com/553/53olapp.htm

Dix A., Finlay J., Abowd G., Beale R. (1998). *Human-computer interaction, 3rd edition.* Prentice hall Europe (ISBN: 0-13-239864-8)

Do**ughty K., Williams G., King P.J., Woods R. (1998).** *DIANA: a telecare system for supporting dementia sufferers in the community.* In Annual International Conference of the IEEE Engineering in Medicine and Biology (volume 4, pp 1980-1983, ISSN: 0589-1019)

Driver P. (1994). *Music technology and the learning process, Aspects of Educational and Training Technology* (volume 27, pp 240-246)

Du Boulay B. and Luckin R. (2001). *Modelling human teaching tactics and strategies for tutoring systems.* In International Journal of Artificial Intelligence in Education (volume 12, number 3, pp235-256)

Dynadirect. (n.d.). *PSR-290 portable Keyboard.* Retrieved 22/06/2003 from Dynadirect, Web site: http://www.dynadirect.com/ek-psr290.html

Ebihara Y. (1992). *Knowledge based spelling correction in Unix command names.* In Journal of information processing (volume 15, number 3, pp394-399, ISSN: 0387-6101)

Elwood. J., Klenowski. V. (2002). Creati*ng Communities of Shared Practice: the challenges of assessment use in learning and teaching.* In Assessment & Evaluation in Higher Education (volume 27, number 3, pp 243 – 256)

E. Winer (1981). *Audio Filters--Theory and Practice.* Retrieved 22/06/2003 from web site: http://www.ethanwiner.com/filters.html

Focus-Texas Instrument. (n.d.). *Analogue audio.* Retrieved 10/03/2003 from Texas Instrument, Web site: http://focus.ti.com/analog/docs/articles.tsp

Gerlič I. (1998). *Fields of Computer Application in Education.* Retrieved 22/06/2003 from University of Maribor (Slovenia), Web site: http://www.pfmb.uni-mb.si/ivan/model98/eng/s1.htm (Site map)

Gingrich D.M. (1995). *PHYS395 Course .* Retrieved 10/03/2003 from University of Alberta, Web site: http://jever.phys.ualberta.ca/~gingrich/phys395/notes/node105.html

Gorayska B., Marsh J., Mey J.L. (1997). *Putting the horse before the cart: Formulating and exploring methods for studying cognitive technology.* In Humanizing the Information age Proceedings of the International Conference on Cognitive technology (pp 2-9)

Gralla P. (1996). *How intranets work.* Ziff Davis. (ISBN: 1-5627644-1-1)

Gupta U. (2000). *Information Systems, Success in the 21st century.* US Imports & PHIPEs. (ISBN: 0-13-010857-X)

Hart A. (1986). *Knowledge acquisition for expert system.* Kogan (ISBN: 1-85091-091-X)

Hearing Siemens. (n.d.). *Analogue signal processing.* Retrieved 22/06/2003 from Siemens Audiology Group, Web site: http://hearing-siemens.com/00_uk/10_produkte/12_analog/121_music1.jsp

Heinemann Educational. (n.d.). *New Music Matters* , Retrieved 22/06/2003 from Heinemann Educactional, Web site: http://www.heinemann.co.uk/secondary/ (section Music)

Heinze A., Stephan V., Surmeli D., Gross H-M. (1999). *A cortical architecture for parallel anticipation of sensorimotor sequences.* In Artificial Neural Networks (volume 1, number 470, pp 407-412, ISBN: 0-8529-6721-7, ISSN: 0537-9989)

Hinton Instruments. (2001). *MIDI protocol Guide.* Retrieved 22/06/2003 from Hinton Instruments, Web site: http://www.hinton.demon.co.uk/midi/midicode.html

Hodson P. (1997). *Local Area Networks.* Continuum International Publishing Group. (ISBN: 1-85805-230-0)

Jarke M., Pohl K. (1993). *Vision driven system engineering.* In IFIP transaction A: computer science and technology (nA-30, pp3-20, ISSN: 0926-5473, ISBN: 0-444-81594-5)

John Stone. (n.d.). *Standard MIDI File 1.0.* Retrieved 22/06/2003 from University of Illinois at Urbana-Champaign, Web site: http://jedi.ks.uiuc.edu/~johns/links/music/midifile.html

Jones R.P. (1992). *Using Artificial Intelligence in Legal Computer Assisted Instruction,* Retrieved 22/06/2003 from Law Technology Journal, University of Warwick, Web site: www.law.warwick.ac.uk/ltj/2-1b.html

Kaicheng Y, Kekang H. (1997). *An Intelligent Hypermedia Teaching system for Chinese Language Learning.* Retrieved 22/06/2003 from Digital Media Research Centre, Web site: www.media.uwe.ac.uk/masoud/cal-97/papers/kekang.htm

Kim S.K., Kim J.W., Kim H.J. (1996). *On-line recognition of cursive Korean characters using neural networks.* In Neurocomputing (volume 10, pp 291-305, , ISSN: 0925-2321/96)

Korg Corporation. (n.d.). *Group Education Controller (GEC3 system).* Retrieved 22/06/2003 from Korg Corporation, Web site: http://www.korg.com/gear/info.asp?A_PROD_NO=GEC3

Kremer R., Norrie D., Flores R., Shen W. (2000). *An infrastructure for flexible, modular multi-user intelligent interfaces* (pp 739-744, ISBN: 0-7803-6583)

Krogh J. (Jan / Feb 1998). *Computers in Music Education.* In Music & Computers (pp 18-34 and p 61)

Lai W., Berthouex P.M. (1992). *Testing an expert system for the activated sludge process.* In Knowledge acquisition in civil engineering, (pp 124-146, ISBN: 0-8726-2864-7)

Lee, S.W., Lim K.C. (Dec 1994). *Address block location on handwritten Korean envelopes by the merging and splitting method.* In Pattern recognition (volume 27, number 12, pp 1641-1651, ISSN: 0031-3203)

Leemis L. (1997). *Seven habits of highly successful input modelers.* In Proceedings of the 1997 winter simulation conference (pp 39-46, ISSN: 0275-0708)

Lehrman P.D., Tully T. (1993). *MIDI for Professional.* Amsco (pp 9-10 and pp 189-208, ISBN: 0-71192-327-2)

Lelouche R. (September 1998). *The successive contributions of computers to education: a survey.* In European Journal of Engineering Education (Volume 23, Number 3, pp 297-308)

Linscott R. (n.d.). *Opamp Theory and Design.* Retrieved 22/06/2003 from home.maine.rr.com (no details, personal web site designed by Randy Linscott), Web site: http://home.maine.rr.com/randylinscott/learn.htm#OPAMP

Luger G.F., Stubblefield W.A. (2002). *Artificial Intelligence: Structures and strategies for complex problem solving.* Addison Wesley (ISBN: 0-201-64866-0)

MacLellan. E. (2001). *Assessment for Learning: the differing perceptions of tutors and students.* In Assessment & Evaluation in Higher Education (volume 26, number 4, pp.307-318)

Maddix F. (1990). *Human-computer interaction, theory and practice.* Ellis Horwood series [ISBN 0-13-446238-6 (library edition), 0-13-446220-3 (student edition)]

McTaggart J. (2001). *Intelligent Tutoring Systems and Education for the future.* Retrieved 22/06/2003 from Drake University, Web site: www.drake.edu/mathcs/mctaggart/CI512X/LitReview.pdf

Mergel B. (1998). *Instructional Design And Learning Theory.* Retrieved 22/06/2003 from University of Saskatchewan, Web site: http://www.usask.ca/education/coursework/802papers/mergel/brenda.htm#The%20Basics%20of%20Cognitivism

Metaxas P.T. (1996). *On user interfaces for education multimedia applications.* IEEE (pp 199-208, ISBN: 0-7803-3173-7)

MiBAC. (n.d.). *MUSIC lessons.* Retrieved 22/06/2003 from MiBAC, Web site: http://www.mibac.com (Section MUSIC Lessons)

Microsoft. (n.d.1). *Windows Media encoder 9.* Retrieved 26/06/2003 from Microsoft, Web site: www.microsoft.com/windows/windowsmedia/players.aspx

Microsoft. (n.d.2). *Windows Media player 9.* Retrieved 26/06/2003 from Microsoft, Web site: www.microsoft.com/windows/windowsmedia/9series/encodder/default.aspx

Microsoft. (n.d.3). *Windows XP Tablet PC Edition.* Retrieved 22/06/2003 from Microsoft, Web site: www.microsoft.com/windowsxp/tabletpc/

Mills J., Murray A. (2000). *Music technology inspected: good teaching in Key Stage 3.* In British Journal of Music Education (volume 17, number 2, pp 129-156)

Mkpe – Karagosian M. (1999). *Software Components Bring Flexibility to Control Networks.* Retrieved 22/06/2003 from MKPE consulting, Web site: http://www.mkpe.com/articles/2000/Components/components.htm

Molnar A. (June 1997). *Computers in Education: A brief History.* Retrieved 22/06/2003 from T.H.E Journal ONLINE, Web site: www.thejournal.com/magazine/vault/a1681.cfm, (Volume 24, Number 11)

Moray N. (Nov 1998). *Identifying mental models of complex human-machine systems.* In International journal of industrial ergonomics (volume 22, number 4-5, pp 293-297, ISSN: 0169-8141)

Music Central Inc. (n.d.). *Music In Education.* Retrieved 22/06/2003 from Music Central (Hopkinsville, USA), Web site: http://www.mus-central.com/MIE.htm

Najjar L.J. (June 1998). *Principles of Educational Multimedia User Interface Design.* In Human Factors (volume 40, number 2, pp 311-323, ISSN: 0018-7208)

Nicholas Haines. (n.d.). *Haines Music Laboratory.* Retrieved 22/06/2003 from Web site: http://www.nicholashaines.com/Music-Laboratory.html

Niessen C., Eyferth K., Bierwagens T. (1999). *Modelling cognitive processes of experienced air traffic controllers.* In Ergonomics (volume 42, number 11, pp 1507-1520, ISSN: 0014-0139)

Norman D.A., Drapers S.W. (1986). *User centred system design, new perspectives on human-computer interaction.* Lawrence Erlbaum associates [ISBN: 0-89859-781-1 (Hardback), 0-89859-872-9 (paperback)]

Odam G. (1997). *Teaching composing in secondary schools.* In British Journal of Music Education (volume 14, number 2, pp 111-126)

Odam G., Walters D. (n.d.). *Dreaming or AWAKE: composing in the classroom.* Retrieved 22/06/2003 from Coda Music Trust (Christchurch, UK), Web site: www.mri.ac.uk/mri/cd/cd1.html

Orwant J. (1996). *For want of a bit the user was lost: Cheap user modelling.* In IBM systems journal (volume 35, number 3-4, IBM systems journal, pp 398-416, ISSN: 0018-8670)

Ou Y.C., Horng S.H., Hu J. (June 2000). *An object oriented interface for control and automation software in open architecture systems.* In Proceedings of the American control conference, AACC (pp 3555-3559, , ISBN: 0-7803-5519-9)

Penfold R.A. (1995). *Practical MIDI Handbook, 3rd edition.* PC Publishing, (pp 1-3 and pp 16-19, ISBN: 1-87077-536-8)

Preece J., Rogers Y., Sharp H., Benyon D., Holland S., Carey T. (1994). *Human-computer interaction.* Addison Wesley (ISBN: 0-201-62769-8)

Quality Assurance Agency (n.d.). *Code of Practice: Assessment of Students General Principles – Quality and Standards in H.E.* Retrieved 22/06/2003 from Quality Assurance Agency, Web site: www.qaa.ac.uk/public/cop/copaosfinal/genprin.htm

RAD - Zalcberg M. and Matityaho B. (1994). *The development of Communication Networks.* Retrieved 22/06/2003 from RAD, Web site: http://www.rad.com/networks/1994/networks/preface.htm

Rane. (n.d.). *Cobranet.* Retrieved 22/06/2003 from Rane (Mukilteo, USA), Web site: www.rane.com/pdf/cobrapps.pdf

Rapture. (n.d.). *Digital Audio Theory.* Retrieved 22/06/2003 from Web site: http://www-personal.engin.umich.edu/~jglettle/section/rapture/digiaudio.html

Raskutti B., Beitz A., Ward B. (1997). *Feature based approach to recommending selections based on past preferences.* In User Modelling and user Adapted Interaction (volume 7, number 3, pp 179-218, ISSN: 0924-1868)

Rational. (n.d.). *UML resource centre.* Retrieved 22/06/2003 from rational group (IBM), Web site: http://www.rational.com/uml/index.jsp

Ravden S.J., Johnson G.I. (1989). *Evaluating usability of human-computer interfaces: A practical Method.* Ellis Horwood ltd [ISBN 0-7568-0614-7 (Ellis Horwood Ltd), ISBN: 0-470-21496-1 (Halsted press)]

Rice – Welsh E. (n.d.). *Background on the Psychoacoustic Model.* Retrieved 22/06/2003 from Rice Web site: http://is.rice.edu/~welsh/elec431/psychoAcoustic.html

Rising Software. (n.d.1). *Auralia.* Retrieved 22/06/2003 from Rising Software (Fairfield, Australia), Web site: http://www.RisingSoftware.com/auralia21/

Rising Software. (n.d.2). *Musition.* Retrieved 22/06/2003 from Rising Software (Fairfield, Australia), Web site: http://www.RisingSoftware.com/musition2/

Rogers K. (1997). *Resourcing music technology in secondary schools.* In British Journal of Music Education (volume 14, number 2, pp 129-136)

RS485. – RES (n.d.). *Quick Reference for RS485, RS422, RS232 and RS423.* Retrieved 22/06/2003 from R.E. Smith (Hamilton, USA), Web site: http://www.rs485.com/rs485spec.html

Russel S., Norvig P. (1995). *Artificial Intelligence: A modern Approach.* Prentice Hall International Editions (ISBN: 0-13-360124-2)

Rust. C. (2002). *The Impact of Assessment on Student Learning.* In Active Learning in Higher Education (volume 3, number 2, pp145-157)

Ruvini J.D., Fagot C. (1998). *IBHYS: A new approach to learn user habits.* In Proceedings of the International conference on Tools with Artificial Intelligence (pp 200-207, ISSN: 1063-6730)

Salaman W. (1997). *Keyboards in school.* In British Journal of Music Education (volume 14, number 2, pp 143-148)

Schneiderman B. (1998). *Designing the user interface strategies for effective human-computer interaction, 3rd edition.* Addisson-Wesley (ISBN: 0-201-69497-2)

Seetrax. (n.d.). *Ranger XL CAD Software.* Retrieved 22/06/2003 from Web site: http://www.seetrax.com/software.htm

Segal R.B., Kephart J.O. (1999). *Mailcat: an intelligent assistant for organizing e-mail.* In Proceedings of the National Conference on Artificial Intelligence (pp 925-926, ISBN: 0-262-51106-1)

Self J. (1988). *Artificial Intelligence and Human Learning (Intelligent Computer-Aided Instruction).* Chapman and Hall (ISBN: 0-412-16610-0)

Shapira B., Hanani U., Raveh, A., Shoval P. (Mar-Apr 1997). *Information filtering: a new two-phase model using stereotypic profiling.* In Journal of intelligent information systems (volume 8, number 2, pp 155-165, ISSN: 0925-9902)

Smoliar S.W., Waterworth J.A., Kellock P.R. (1995). *PianoFORTE: A system for piano education beyond notation literacy.* In Proceedings of the ACM International Multimedia Conference & Exhibition 1995 (pp 457-465)

So A.T.P., Tse K.Y., Chan W.L. (1994). *Music education in hypermedia environment.* In IEEE international Conference on Multi-Media Engineering Education Proceedings 1994 (pp 392-397)

Squires D., Preece J. (1999). *Predicting quality in educational software: evaluating for learning.* In Usability and the synergy between them, Interacting with computers (volume 11, Elsevier, pp 467-483, ISSN: 0953-5438)

Stankov S. (March 1996). *Computers in Education: technological transformation, development and perspective.* In grkg/Humankybernetik (Band 37 – heft 1, pp 16-26)

Taras. M. (2002). *Using Assessment for Learning and Learning from Assessment.* In Assessment & Evaluation in Higher Education (volume 27,number 6, pp 501 –510)

Terveen L., Murray La T. (1996). *Helping users program their personal agents.* In Conference on human factors in computing systems (pp 355-361, ISBN: 0-89791-777-4/96)

Tewkesbury G.E. (1994). *Design Using Distributed Intelligence Within Advanced Production Machinery.* PhD Thesis (University of Portsmouth, UK)

Thimbledy H. (1990). *User Interface design.* ACM press frontier series (ISBN: 0-201-4168-2)

Tomeski E.A., Lazarus H. (1975). *People oriented computer systems, the computer in crisis.* Van Nostrand Reinhold company (ISBN: 0-442-28556-6)

UK Governement. (1998). *Data Protection Act 1998.* HMSO. (ISBN: 0-10-542998-8). Retrieved 22/06/2003 from Web Site: http://www.hmso.gov.uk/acts/acts1998/19980029.htm

Veloso M., Stone P., Bowling M. (1999). *Anticipation as a key for collaboration in a team of agents: a case study in robotic soccer.* In SPIE conference on sensor fusion and decentralized control in robotic systems II (volume 3839, pp 134-141, ISSN: 0277-786X)

Vivancos Calvet J., Serres Moliner E. (1999). *MULTIMOLD: multi-agent system for intelligent calculation and design moulds.* IEEE (pp 635-643, ISBN: 0-7803-5670-5)

Volberda H.W., Rutges A. (1999). *Farsys: a knowledge-based system for managing strategic change.* In Decision support systems (volume 26, number 2, Elsevier, pp 99-123, ISSN: 0167-9236)

Vraneš S., Stanojević M., Stevanović V., Lučin M. (May 1996). *INVEX: Investment advisory expert system.* In Expert systems (volume 13, number 2, pp 105-119, ISSN: 4542-3735)

Wathieu L. (Nov 97). *Habits and the anomalies in intertemporal choice.* In management science (volume 43, number 11, pp 1552-1563, ISSN: 0025-1909)

Webopedia. (n.d.). *Online dictionary for computer and Internet terms.* Retrieved 22/06/2003 from Jupitermedia Corporation (copyright 2003), Web site: http://webopedia.internet.com

Whatis-Techtarget. (n.d.). *Online dictionary for computer and Internet terms.* Retrieved 22/06/2003 from TechTarget Network (www.techtarget.com), Web site: http://whatis.techtarget.com

Wilson B.G. and Cole P. (1996). *Cognitive Teaching Models.* Retrieved 22/06/2003 from University of Colorado at Denver, Web site: http://carbon.cudenver.edu/~bwilson/hndbkch.html

Wing A.M., Flanagan J.R. (1998). *Anticipating dynamic loads in handling objects.* In Proceedings of the ASME dynamic systems and control division (volume 64, ASME, pp 139-143, 0-7918-1586-2)

Yamaha Corporation. (n.d.1). *Mlan.* Retrieved 22/06/2003 from Yamaha (Copyright 2002, www.yamaha.com), Web site: http://www.yamahasynth.com/pro/mlan/

Yamaha Corporation. (n.d.2). *PSR series (Portatone).* Retrieved 22/06/2003 from Yamaha (Copyright 2001, www.yamaha.com), Web site: http://www.yamahapkclub.com/english/top_e.htm (section Portatone)

Yeates D., Shields M., Hely D. (1994). *Systems analysis and design.* Pitman publishing (ISBN: 0-273-60066-4)

Yu L., O'Brien A. (1999). *Practical typology of adult fiction borrowers based on their reading habits.* In Journal of Information Science (volume 25, number 1, pp 35-49, ISSN: 0165-5515)

Zalila Z., Fatene M., Kadhi R. (1998). *Fuzzy supervision in statistical process control.* In Proceedings of the IEEE international conference on systems, man and cybernetics (volume 1, pp 638-643, ISBN: 0-7803-4778-1, ISSN: 1062-922X)

PUBLICATIONS

Lassauniere A., Tewkesbury G.E., Sanders D.A., Marchant J., Close. A, (1999a). *A New Music Teaching Technology System to Teach Music.* In IEEE Proceedings of Euromicro (Volume 2, pp 30-34, IEEE, ISSN 1089-6503)

Lassauniere A, Marchant J., Close. A. (1999b). *A Prototype Remote System To Teach Music to Disabled Students that can be Powered from a Wheelchair Battery.* In Intelligent mobility, Proceedings of the 2nd symposium on powered vehicles for disabled persons (pp 32-36, PCR, ISBN 0-9527524-1-7)

Lassauniere A., Tewkesbury G.E., Sanders D.A., Marchant J., Close. A. (2000). *A New Computer Controlled Music Teaching System to Assist Music teachers in Education.* In Journal of Computing in Systems and Engineering (pp 5-8, ISSN 1472-9083)

Lassauniere A., Tewkesbury G.E., Sanders D.A., Bergasa J. (July 2003). *Computer controlled systems for music teaching.* Accepted by International Conference Information Communication Technologies In Education (ICICTE), INEAG and UCFV.

Bergasa Suso J., Lassauniere A., Marchant J., Close A. (2003) *Assisting Teachers in the Use of the Internet with a 'Caught In the Act' (CITA) Filter.* Accepted by Journal of computing in systems and engineering (ISSN: 1472-9083)

Lassauniere A., Tewkesbury G.E., Sanders D.A. (n.d.). *A New Music Technology System Controlled by Computer or Keyboard.* Submitted to 'Software practice and experience, Wiley