Incremental sound installation development using the Smart Controller

Angelo Fraietta

PO Box 859, Hamilton NSW, 2303
email: angelo_f@bigpond.com

Abstract

The Smart Controller is a powerful integrated device that integrates control voltage, MIDI, and algorithmic generation into a single hardware device, which in turn eliminates the requirement of a laptop computer at a performance or sound installation. The device is programmed using a graphical iconic environment using a Windows or Macintosh OSX computer.

This paper will explain how the Smart Controller was implemented in Anne Norman's Garden Bells installation, where the Smart Controller played bells by actuating solenoids in response to scheduled MIDI sequences and environmental events. This paper will detail how the Smart Controller patches were incrementally and iteratively developed in order to allow the composer to effectively develop the remaining hardware portion of the installation.

1 Introduction

The author was approached by composer Anne Norman with the proposition of assisting her with the realisation of an electro-acoustic Bell Garden installation (Norman 2003a). The sound installation is based around a collection of galvanised iron caps that were once fixed to the tops of power poles. Apart from the original engineering use, the caps also function quite well as microtonal bells, and subsequently, have been used in musical performances and recordings (Norman 2003b). The composer decided to create an interactive environment that responds to both motion and percussion. Furthermore, the behaviour of the Bell Garden changes through time, whereby each performance phase of the Bell Garden was significantly different. The composer provided the following performance details:

This multifunction Bell forest can be performed:

(1) with beaters acoustically

(2) with subtle amplification of acoustic sound

(3) with electronic effect modulation of acoustic sound

(4) through movement detectors triggering an auto striking mechanism

(5) through movement detectors triggering MIDI samples of unexpected sounds and voices (Norman 2003a)

The author's responsibilities in this project would be to design a processor to schedule the interactive component of the installation, and to provide technical advice to the composer. After examining the composer's brief outline of the installation, the author suspected that the scheduled functions required for the installation were: (1) responding to audience movement, (2) generation of electrical signals to trigger a mechanical striking of the bells, and (3) sample playback. The author suggested that the Smart Controller would be able to perform the scheduling and triggering aspects of the performance, and subsequently, the composer chose to use the Smart Controller.

The primary concern in this collaboration was distance – the author was in Newcastle NSW while the composer was in Mornington Victoria. The second concern perceived was that the composer owned a Macintosh and there was no Smart Controller patch editor available for that platform. As the project developed, however, other complications became evident. The area of particular concern was the composer's lack of technical knowledge, which was quite understandable as this was her first installation.

Consequently, this led to communications deficiencies between the author and the composer, whereby each person falsely believed that the other party understood the logistics and their own area of responsibility within the project. As communications progressed, both parties realised the magnitude of the project was completely underestimated in terms of cost, complexity, and effort. Nevertheless, although the installation was not complete at the time of writing, the author has the full confidence in the success of the installation due to an iterative approach to development (MacCormack 2001).

This paper details how the author was able to map the composer's artistic concepts to technical specifications, and finally, to schematic diagrams and Smart Controller patches through iterative and incremental design strategies.

2 Requirements

During the development of any software or hardware project, a set of requirements is defined at the commencement. When using the waterfall lifecycle method of system development, the entire system is designed and documented before performing any implementation (Royce 1970). The problem with this metaphor is there is no active attempt to identify risk. Risks come in many forms, which include the use of an incorrect set of requirements, and a lack of skills or resources (Larman 2002).[1]

By contrast, the Unified Process (Jacobson, Booch, and Rumbaugh 1999) uses a practice called "iterative development" (Larman 2002), whereby the development is organised into short sub-projects called "iterations" (Larman 2002). Each iteration is a standalone executable system that can be tested by itself apart from the main project to which it belongs. One of the main advantages of this approach is that it is based upon the premise that not everything is known upfront (MacCormack 2001), this giving the designer the freedom to redirect development as a result of the preceding test. This approach appeared best in the situation as both the composer and author were unaware of the complete design at the outset. [2]

The composer provided a set of input and output requirements whereby the author constructed requirements tables 1 and 2 in order to make mapping simpler (Petty 1998).

Additionally, the composer provided a phase outline that described each phase of operation. The phases included playing the bells using MIDI sequences, bells being played as a response to movement near the bell, modification of electronic effects settings, samples being triggered in response to the bells being struck, and playback of pre-recoded music from compact disk. The information was limited because the composer did not have any sequences composed, did not know what type of effects unit to use, and did not have the MIDI sampler.

Although the complete implementation details were far from complete, there was enough information to start patch development.

Inputs			
Quantity	**Type**	**Purpose**	**Implementation**
10	Microphone	Collect audio for modification by effects unit	Plug directly into effects unit. Also used to trigger MIDI samples.
10	Infrared diodes	Monitor movement	An infrared light source would normally be on the diodes. Moving in front of the diode would cause a break in the light source, causing the diode to become high impedance. The diodes could be plugged directly into digital inputs of Smart Controller.

Table 1 Input Requirements

Outputs			
Quantity	**Type**	**Purpose**	**Implementation**
10	Solenoid	Mechanically strike bells	Use digital outputs from Smart Controller to trigger solenoids.
3	Light	Display messages	Use digital outputs to enable message display.
3 or 4	Sound effect setting	Modify audio collected through microphone	Use MIDI program change message to select appropriate sound effect setting
26	Audio samples	Playback through speakers	Use MIDI note messages on a MIDI sampler

Table 2 Output Requirements

3 Iteration one

The target of the first iteration was to provide the composer with a Smart Controller unit with patches that would load as soon as the device was powered on. The ability to install new patches installed on the device was possible using a MIDI system exclusive data stream, whereby the new patch would be sent through the MIDI port on the composer's Macintosh.

[1] The waterfall process is similar to the engineering approach used to construct bridges and buildings. The blueprints are completed before any construction takes place. The use of this approach made software development appear more structured and robust -- similar to other engineering fields.

[2] A two year study in successful software projects (MacCormack 2001) revealed that the highest common factor in successful projects was the use of iterative development rather than the waterfall lifecycle.

Additionally, the Smart Controller firmware could be upgraded through the MIDI port, thus making it unnecessary to return the Smart Controller unit for firmware upgrades.

The functionality of the patches provided in the first iteration would be twofold. Firstly, the Smart Controller would be able to respond to the movement sensors and generate a pulse on the appropriate output. This would allow the composer to create input sensors and test the operation of the solenoid strikers. Secondly, the Smart Controller would respond to MIDI note messages and generate a pulse on the appropriate bell striker output. This would enable the composer to compose the MIDI sequences required by using the Smart Controller as the bell driver. The composer could use the existing sequencing software on her Macintosh and plug the MIDI output into the Smart Controller. As the sequencer produced the MIDI, the Smart Controller would simply convert them to the appropriate outputs to drive the solenoids.

In order to test both the movement sensors and the solenoid strikers, the digital inputs of the Smart Controller were mapped to the digital outputs. Shorting a digital input would pulse the corresponding digital output for a defined period. Additionally, the pulse had to be prevented from being re-triggered while the output pulse was being generated. The composer was unsure of the actuation period required by the solenoid. A pulse duration that was too small would be insufficient to make the solenoid strike the bell, while an excessive duration would dampen the bell. Considering that neither the author nor the composer were able to ascertain what the required duration was, an upper limit of 250

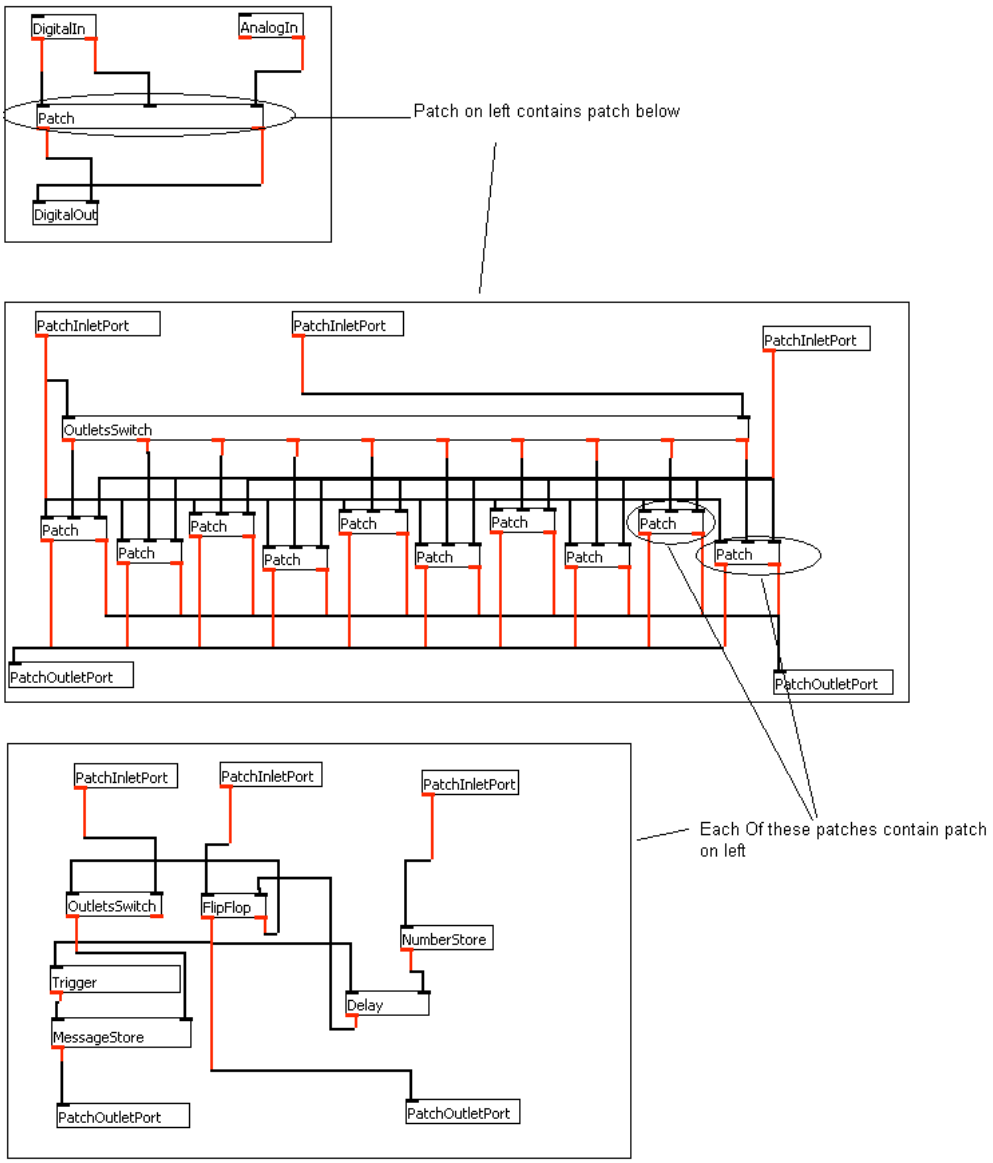


Figure 1 Digital input mapped to solenoid strikers

milliseconds was stated. The duration would be modified within the patch by sending and adjusting a control voltage, which could be effected by using a single potentiometer on an analogue input. The digital value of the analogue input, which ranges from 0 to 255, would be used to directly determine the pulse duration. In order to produce the same function for all ten inputs, the method to generate the pulse was contained in a separate patch, and subsequently included ten times in the outer patch. The resultant patch and all the sub patches are displayed in figure 1.

Next, MIDI input was mapped so as to produce a bell striker output based upon the note number input. This was effected using a calculation of modulo 12, and then adding 1 to the result. Subsequently, all "C" notes generated a pulse on output 1, "C#" on output 2, and so on up to "A." The notes were sent to the same patch as the digital inputs, and therefore produced an identical output duration that would not be retriggered before the required time had elapsed. These new objects were added to the patch in figure 1, resulting in the patch displayed in figure 2.

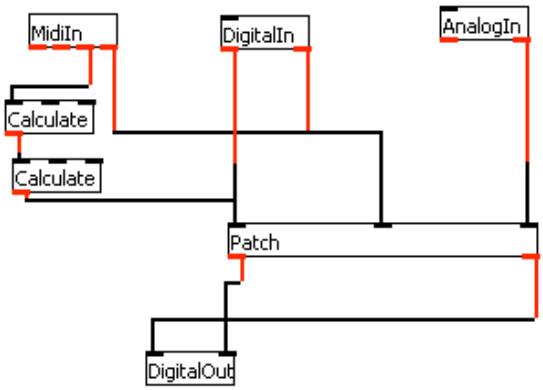


Figure 2 MIDI input added to digital input mapping

After testing the patches operated correctly within the simulator, they were loaded into the device and configured to load immediately on power up. The patches were tested by plugging light emitting diodes (hereafter LED) into the digital outputs and shorting the digital inputs one at a time, ensuring that the correct LED flashed. Additionally, a potentiometer was placed on the analogue input and manipulated to ensure that the flash rate was adjustable via the analogue input. Next, a MIDI keyboard was plugged into the input and played, ensuring that the correct LED flashed, depending on the note number. The composer would be able to test the movement sensors by plugging them directly into the digital inputs and monitoring LEDs placed on the appropriate outputs. Also, the solenoid strikers could be tested by connecting them to the digital outputs, connecting a MIDI keyboard to the input, and ensuring that a played note produced the appropriate output.

The Smart Controller was shipped to the composer, and patches contained within MIDI system exclusive messages were sent via email. The first step was to confirm that the composer would be able to download newer patches from her Macintosh running OS9.2. The composer was unable to import the data file into a MIDI file through her sequencer, and subsequently found a software package that could send the raw midi data successfully. The composer was unable to continue development with the device for one month due to other commitments. The composer indicated a very positive willingness to upgrade the software on her Macintosh to OSX, and subsequently, this break gave the author sufficient time and motivation to develop the Macintosh patch editor. The first version was complete by the time the composer resumed work on the installation.

On resumption of work, the composer commenced development of the sensors, using the Smart Controller to test them. The composer had an intermittent problem, whereby the Smart Controller appeared to fail after a short period of time.

The Smart Controller facilitates the use of a detachable plug, as displayed in figure 3, whereby a loom can be soldered to the plug and attached easily during setup. This enables all the inputs to be attached using a single plug connection during the setup of a performance. The same mechanism is also provided for the outputs. This enables the Smart Controller to be easily removed and used in different sound installations. The composer had created input and output looms by soldering to the detachable plugs. A LED was soldered to the output, and the device was tested by shorting the digital input, adjusting the potentiometer for the flash duration, and monitoring the LED.

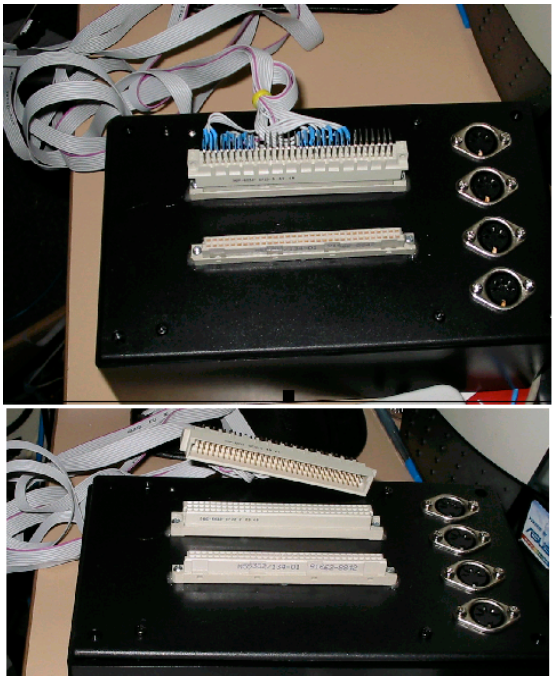


Figure 3 Smart Controller I/O plug

Input and output sensors can be connected directly into the Smart Controller inputs without the detachable plug, as displayed in figure 4, allowing users to plug three pin sensors, identical to the Infusion Systems I-Cube sensors (Infusion Systems 1998), directly into the device. The author directed the composer to remove the plugs and insert a potentiometer and LED directly into the Smart Controller inputs and outputs respectively, and to use a wire to short the digital inputs directly. The composer confirmed that device was working correctly and that the problem was probably due to unsatisfactory soldering to the detachable plugs.

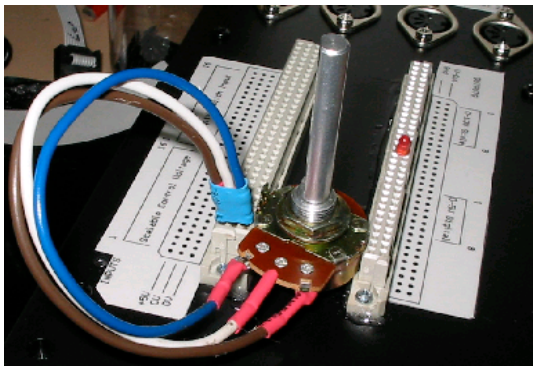


Figure 4 Three pin sensor attachment with LED on output

Next, the composer stated that the physical layout of the Smart Controller was extremely difficult to work with in this particular installation. Consequently, the author will be installing the Smart Controller into a standard rack mount unit, which will be provided by the composer.

4 Iteration two

At this point, the composer was unable to continue on the project implementation for another five weeks due to other commitments; however, the composer provided ideas and directions to the author in regard to the composition of the work, allowing the author to continue developing patches for the Smart Controller.

The composer's current area of focus is the physical layout of installation. The author was concerned with noise induced into the microphones leads due to the distance they would be required to travel alongside other signals. Additionally, plugging the microphones directly into the Smart Controller analogue inputs would cause the input to become unbalanced. This was a concern because the composer also intended to amplify the bells through these same microphones. One possibility to overcome this would be to provide a set of low cost transducers to provide audio triggers to the Smart Controller. Alternatively, an operational amplifier could be used at the microphone source to provide isolation between the audio and the triggering circuits. The composer, however, decided to remove the requirement for audio triggering samples, and subsequently, this patch will be no longer required. This phase will be replaced by a separate set of samples being triggered by the movement sensors, and subsequently a new patch will need to be written.

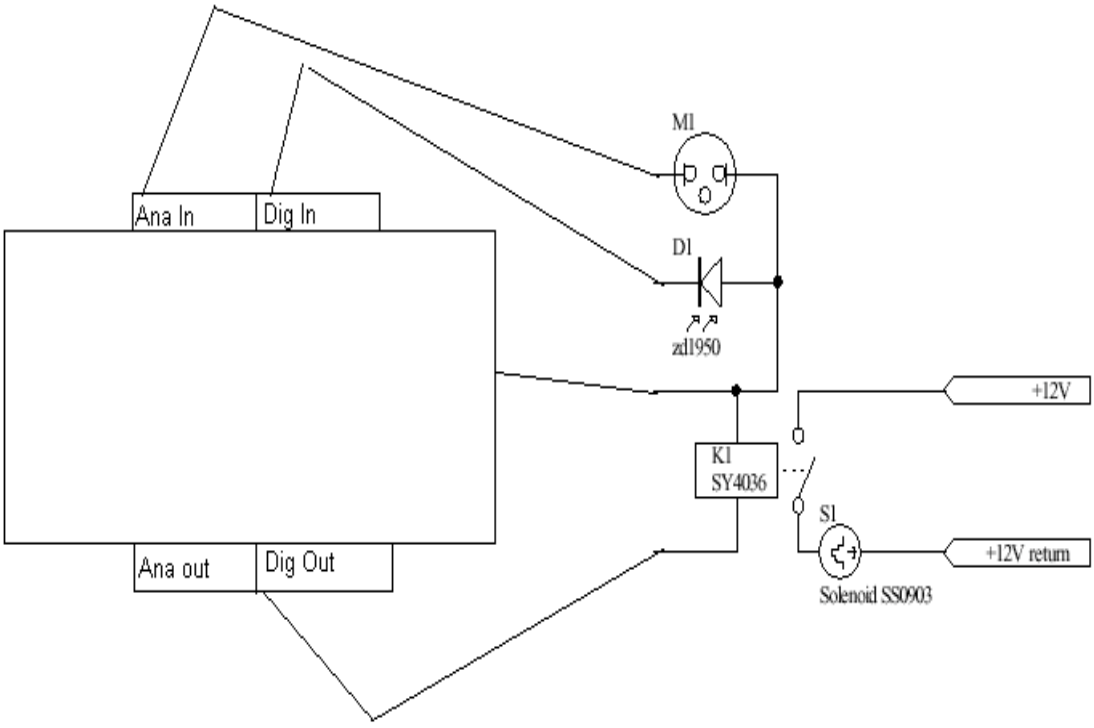


Figure 5 Smart Controller to bell stand logical connection

Another concern with regard to the physical layout was the number of cables that needed to be run and the complexity of the loom required to interface between the Smart Controller and the bell stands. Each bell stand required at least four wires that are mapped to the Smart Controller.

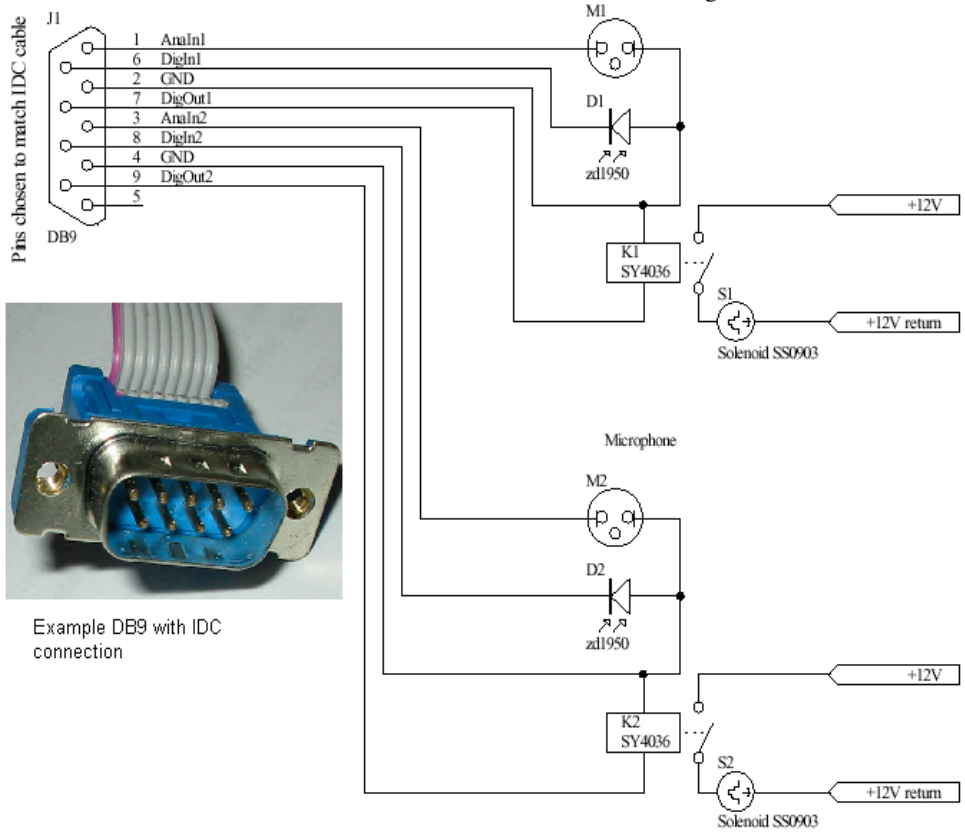
Figure 5 displays one of ten bell stand to Smart Controller connection maps. The microphone trigger (M1) is mapped to the analogue input; the infrared sensor (D1) is mapped to the digital input; the solenoid bell striker (S1) is mapped to the digital output; and the common is mapped to the Smart Controller earth. A loom to connect the two detachable Smart Controller plugs, displayed in figure 3, to all the bell stands would require at least forty solder joints. In order to reduce the number of solder connections required by the composer, it was decided that the rack mount unit would interface directly with the bell stands. The Smart Controller inputs and outputs could be mapped to the bell stands inside the rack mount unit, providing a single DB9 plug interface for every two bell stands. The composer can then use IDC plugs and ribbon cable without having to perform any soldering. The bell stands can be grouped together using physically adjacent pins instead of logical pin numbers, allowing a standard 16-way IDC ribbon cable to be split to create two cables. Then, at the bell stand location, the cable can be physically split in two again, creating two cable runs without requiring a solder connection. Figure 6 displays the arrangement as a schematic diagram.

The bell stands will still require a heavier gauge pair for the 12V solenoid supply and the speakers, and a shielded pair for microphone; however, the responsibility for this is left to the composer.

5 Iteration three

The third iteration in the bell garden will be the programming of the scheduling and the MIDI messages for the sampler and effects unit. Although the patch editor can now be run on the composer's Macintosh, the composer has never used software such as MAX, and subsequently, is unfamiliar with that programming paradigm. This effectively means that the author will perform all the programming. This will require the author to write the patch, email it to the composer, who will download it to the unit to determine its suitability, and then provide feedback as to what is unsatisfactory. This cycle will continue for each phase required for the installation.

The information required for the sampler will be the mapping of required note numbers to events (Doornbusch 2002). This could be communicated through the use of a table that maps triggered inputs to note numbers. Alternatively, a statement such as "digital inputs 1 to 10 will generate MIDI note number 60 with a velocity of 127; the MIDI channel being the same as the input channel." If digital input 5 was detected, the Smart Controller would produce MIDI note number 60, velocity 127 on MIDI channel 5. Alternatively, one could state "digital inputs 1 to 10 will generate MIDI notes with a velocity of 127 on



Example DB9 with IDC connection

Figure 6 Connection of Smart Controller to bell stands using IDC ribbon

MIDI channel 1, ranging from 60 to 69 respectively." In this example, if digital input 5 was detected, the Smart Controller would generate MIDI note number 64, velocity 127 on MIDI channel 1.

The information required for the effects could be the program change number for a particular phase. For example "switching to manual play phase 2 requires that MIDI program change 12 be sent on MIDI channel 1." Moving to this phase would cause the Smart Controller to generate the required program change message when entering that stage. The author, however, will be unable to write the patches until this information is provided by the composer.

The composer could simulate these events by creating a MIDI sequence using a standard sequencing software package, and programming the required events at this stage. Logically, it is possible for the entire phase scheduling to be implemented using a MIDI sequence. The composer could then use the sequencing software that she is familiar with to simulate the performances by plugging the output of the sequencer into the Smart Controller. Although this is similar to the method suggested for creating melodic sequences, it is different because the Smart Controller would respond to MIDI events, enabling and disabling certain mappings depending upon the message sent by the sequencer on a particular channel. For example, MIDI channel 16 program change messages could be used to change performance phases. The composer would then simply have to download the sequence into the Smart Controller.

6 Other factors

Apart from factors relating directly to the Smart Controller, issues such as the physical layout of the bell stands, the bell striking mechanism, layout of cable runs, microphones, and speakers are generally outside the domain of the Smart Controller. Consequently, these are not addressed in this paper. There is, however, a possibility that the digital output drive from the Smart Controller is insufficient due to the distance of cable, in which case, line drivers can be added within the rack mount unit. Other unforseen factors, which cannot be addressed as they are unknown, will be dealt with when they become evident or when a fix is required. This feature is part of very nature of iterative development (Larman 2002).

7 Conclusion

Although communication using telephone and email make the implementation quite achievable, the lack of substantial time blocks coupled with the composer's limited knowledge of electronics, the first presentation of the Bell Garden will not contain all the operations originally desired by the composer. Nevertheless, despite the problems associated with the collaborative process, the incremental nature of the iterative method combined with the flexibility of the Smart Controller will result in a successful conclusion. Using the iterative method, the composer was able to recognise at the midway point what targets were achievable within the required timeframe. Finally, the composer is now able to use feedback from this realisation of the Bell Garden to plan for Bell Garden 2.

References

Doornbusch, Paul. 2002. The application of mapping in composition and design. In proceedings of *Form, space, time: the Australasian Computer Music Conference*. Royal Melbourne Institute of Technology.

Infusion Systems. 1998. *I-Cube System Manual*. Last accessed 5 March 2003. http://www.infusionsystems.com/support/icubex-111-manual.pdf.

Jacobson, Ivar, Grady Booch, and James Rumbaugh. 1999. *The unified software development process*. Reading, Mass: Addison-Wesley.

Larman, Craig. 2002. *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR.

MacCormack, Alan. 2001. "Product-development practices that work: How Internet companies build software." *Mit Sloan Management Review* 42 (2):75-84.

Norman, Anne. 2003a. Email to A. Fraietta, Electro-acoustic Bell forest, 6 January 2003.

———. 2003b. *Power Pole Bells*. Anne Norman. Last accessed 12 March 2003. http://home.vicnet.net.au/~amncrow/PPBells.html .

Petty, Bob. 1998. "Requirements Management Using Tables." *Embedded Systems Programming* 11 (13):54-60.

Royce, Winston W. 1970. Managing the development of large software systems. In proceedings of *IEEE WESCON*.