3. Design: eRadio

Frida **Kahlo**.
Self Portrait on the **Borderline** Between **Mexico** and the **United States**
Henry Ford Hospital, Detroit, USA, 1932.

An expression of Frida's situation:
stuck in a foreign country and disconnected from her homeland.

> "One of the most important and most overlooked development assets is organization—the ability to mobilize and organize for collective action." [Narayan and Shah, 2000. p.7]

This chapter describes how I first envisioned the eRadio project, and it is written accordingly. The description includes the design, the methodological approach employed, and the electronic tools used in the pilot implementation. My objective was to design a methodology that could trigger the generation of ideas in a target community as well as to specify the features needed in electronic tools that could be custom-made or bought to best fit that methodology. However, fostering participatory community involvement was the key design challenge.

Section 3.1 describes the logistics of the workshop, Section 3.2 the proposed methodology, and Section 3.3 the technology implemented.

3.1 Logistics

Logistics deal with each part of a process, resources needed, timing, and integration of all the parts. In the case of the eRadio workshop, logistics meant making decisions regarding the community, participants, workshop, program, and timely deployment and installation of tools.

3.1.1 Community

Although the eRadio project could be applied in different settings, it was meant to be for communities with high levels of migration. My plan was to work with an underserved community in Mexico that had many of its migrants localized in one city within the United States that needed to increase their transnational interaction. eRadio could help boost their existing network. Since there exists a migrant pattern of moving in blocks, a relationship network is established and "it has a tendency to self-sustain, in such a way that the migrants from the same place or region of origin practically monopolize access to specific work market segments" [Duran, 2000. p. 257 (my translation)].

The notion of community is not restricted by space or time. "The concept of community concerns a particularly constituted set of social relationships based on something which the participants have in common—usually a common sense of identity." [Marshall, 1998] Another definition of community is "a body of persons […] having a common history or common social, economic, and political interests" [Merriam-Webster, 2000]. Individual members or groups of a community may be scattered geographically and may have different times or schedules, all of which can be more of a motivator and a *raison d'être* for eRadio than an insurmountable deterrent. Therefore, reuniting diaspora communities was an attainable ideal. Expected benefits

were initiating or strengthening the sensation of being with the community, belonging to the community, and sharing with the community.

I planned to carry out the workshop in Spanish since it was the mother tongue of the Mexican target community. This way, one benefit of the eRadio project was that oral language could be used for the transfer of knowledge in the training workshops and in the resulting audio pieces. Since dissemination is through verbal communication via web radio or radio waves, it could also be used to reach people who speak indigenous languages and to preserve those languages that were in danger of extinction. This is specially important in Mexico because according to "The most conservative estimates […] ten percent of the country's population is indigenous—twelve million indigenous people belonging to nearly sixty ethnic groups with diverse languages and cultures." [Ramos and Diez, 2003. p. 174] Moreover, since the methodology and the tools would be oriented to people within a wide range of ages, no familiarity with computers, and little or no reading and writing skills, almost anybody would be able to participate.

3.1.2 Participants

A participative and involved community is a community that has participative and involved individuals who have different responsibilities, who perform different tasks, who do their individual share with regard to a given project or in different walks of life in their community.

The notion of community participation and involvement should not be seen quantitatively. Even though quantity matters, quality is what makes the difference, as will be seen by the results obtained in the town of Tulcingo and in New York City.

In this project, volunteers[1] who go through the radio production process will be called communicators. They can be diverse and have different backgrounds and characteristics as long as they have the willingness and time to work on the eRadio project.

"Letting communities of people of different ages, backgrounds and literacy skills create and critique" [Ananny and Strohecker, 2002. p. 2] will avoid the not-inclusion of the poorest of a community when giving to a community access to technology [Gumucio, 2003].

3.1.3 Workshop

I planned workshops of six to ten communicators per location, aiming to get at least the same number of audio pieces. A brief description of the main components of the workshop follows as well as the expected requirements for each component.

[1] Volunteers signed an Informed Consent form approved by the MIT Committee On the Use of Humans as Experimental Subjects.

Location

The location of the Operational Center, the name given to the location where the electronic tools were going to be installed, had to be accessible to everybody in town so that communicators could use the electronic tools, get advice on the processes, work cooperatively, and have a place to work on their projects.

Scheduling

The scheduling of the workshops was planned to introduce the communicators to the methodology and tools in a period of nine days. Two weekends would have fixed schedules for whole group activities, and five weekdays would be for individual or team tasks at schedules of their convenience. The last Sunday of the workshop would be for transmitting the final audio pieces produced by the communicators at one site to those at the other.

Sessions

The sessions, as mentioned above, would be of two kinds: the group sessions and the individual sessions. In the group sessions, cooperation, apprenticeship, and guided participation would be fostered. The individual sessions would be more a matter of participatory appropriation.

Group sessions

The group sessions were meant to be eight-hours long with a one-hour break for lunch. The first weekend was allocated to introduce the project and to do team, radio production. The introduction would be given by explaining the concepts, giving a demonstration on how to use the tools, and by playing some audio piece examples. The hands-on team, radio production sessions were for the participants to create a simple audio piece by handling and experimenting with the equipment (discovery method).

In the second weekend, the group would listen to everybody's pieces and make comments on them, in case some improvement could be done before the transmission. Participants had to also make decisions about the structure of their program (sequence of the pieces, introduction and closing), and its transmission (coordinate with the remote communicators for setting the timing and dynamics of the transmission).

Individual sessions

The objective of these sessions was to get the participants to work on their own, producing the content and doing the recordings and the editing. However, guidance would be at hand, face to face, via Internet and, possibly, by telephone.

3.1.4 Radio programming

Radio programming deals with time, length, and frequency of transmission; topic, content, number, and sequence of the audio pieces; as well as the media used. The program can be a once-only or periodic transmission with fixed time and day. The pilot implementation was planned as a two-way, once-only transmission; that is, produced and transmitted in two directions, once each, first from Tulcingo to New York, and then from New York to Tulcingo. The two eRadio pilot-implementation programs would have an opening, a body comprised by the audio pieces and a closing, and they would have no fixed topical area.

Media

Content in the project is of great importance if we take into account Marshall McLuhan's insight: "The message is the media." [McLuhan, 1994, p.7] The Tulcingo community would surely be impacted by listening to a loved one, two-thousand kilometers away or by learning of the success of a former neighbor. As a result, eRadio could pave the way for community-to-community communication, surpassing person-to-person telephone conversations. The combination of telephone, radio, and Internet would connect two distant sites and FM radio would transmit within a site.

Radio/Internet is more than its communication potentials [Girard, 2003]. In this project, it would also be an audio piece repository of the community affairs and history.

3.2 Methodology

The eRadio project makes flexible use of principles, strategies and techniques taken from various fields of knowledge in order to achieve its aims. Educationally, the project adheres mainly to the 'constructionist' approach [Papert, 1993a] and to 'inquiry' and 'discovery' approaches [Dewey, 1916; Postman & Weingartner, 1969]. Strategy-wise, it adheres mainly to 'ethnomethodological' strategies and techniques [Garfinkel, Lynch, and Livingstone, 1983] and to 'technomethodological' precepts [Dourish and Button, 1998].

eRadio methodology

The aim was to develop a community-specific, on going, long-term cyclic process of self-discovery and empowerment, by bringing together electronic tools, applications, procedures, and persons and making use of inquiry, discovery, and ethno-methodological strategies and techniques to get the community to generate content—a voice—and to web-cast it so as to multiply participation, generate more feedback, and further the iterative process.

The methodology to be followed was that of a collaborative project that encouraged members of the community to participate in the production of radio programs that would be recorded, edited, and transmitted by them. A selected group of voluntary participants would learn to handle the electronic tools and to carry out production, creation, and elocution tasks and to elicit from members of the community relevant content, for example, anecdotes, interviews, music, news, and fun talk. Both groups would listen to each other's programs, initiating an interactive process that could lead to integration and empowerment in spite of geographic and cultural separation.

Therefore, as mentioned earlier, the project's methodology should lead to an effective correlation of the main elements it would deal with: people, audio pieces, processes, radio program, the workshop, and the tools.

In eRadio, the software, process, and workshop would be structured the same way; that is, going through four stages, gather, edit, publish, and listen. However, even when the process established the standard stages, it would be flexible enough to let communicators learn by themselves and to let them contribute to improving the process, considering that "the critical content of any learning experience is the method or process through which the learning occurs." [Postman and Weingartner, 1969. p. 19]

Progress was also expected in terms of handling of the tools and of communication skills, and consequently of passing on this know-how to others in the future. One community can directly or indirectly motivate other communities to initiate their own process of self-discovery and empowerment as well as to participate in an inter-community process of learning from and of supporting each other. Communicators were expected to learn from observing and listening "with the intent concentration and initiative, and their collaborative participation is expected when they are ready" [Rogoff, *et al.,* 2003. p. 176] to propagate the knowledge and skills they learned to others through apprenticeship or following the same method as was used with them.

Generating self-discovery

The purpose of this project was to get individuals and the community to discover who they are, what they have, and what they can do; that is, to generate content, (something to say, messages) to develop a voice, and to broadcast or web-cast the content within the community and to other communities, multiplying participation, generating feedback, and furthering the iterative process.

To accomplish this self-discovery, I considered the inquiry approach would be the most efficient means to get their intuitive knowledge to surface.

3.2.1 Radio production process

My goal was for communicators to assimilate the radio production process by observing, listening and experiencing, with minimal verbal explanations. [Papert, 1993a; Rogoff, *et al.*, 2003].

This section is concerned with the procedural aspects that take place during the creation of an audio piece. The main process consists of the four stages that are explained below: gather, produce, publish and listen. However, some intermediate steps are the links between one step and the other, such as idea generation, where communicators imagine their final product, they write down the script or the interview questions, and design audio ambience or effects.

The Gather stage

Gathering is the stage at which communicators obtain content in the form of an audio recording; that is, interviews, voices, events, and live effects. Of all the possible kinds of recordings, I will describe two: one is interviewing and the other is live sound effects.

Interviewing

Interviewing starts with asking the right question. A question has to consider the way of thinking of the potential interviewee [Freire and Faundez, 1989. p.34]. The question has to help the communicators develop and internalize the concepts to take them through a "testing and verifying, reordering and reclassifying, modifying and extending process […to transform the subject into] an active producer of knowledge." [Postman and Weingartner, 1969. p.59]

Before interviewing, and other gathering techniques, permission to record should always be requested. The United States' and Mexico's laws require prior consent of at least one party of the conversation. [RTNDA site; CDDHCU site, 2004]

Live sound

Live sound effects give context and orientation. The communicators have to try to make the audience experience what is being transmitted, for the same reason an audio piece has to have only the right amount of live sound effects, recording sounds of what is being talked about; e.g. when talking about typewriting, we may be hearing a typewriter in the background. [Hilliard, c2004]

Tools technique

It is convenient to know about and develop recording technique before setting out to do an interview or to record an event that may not take place again.

Microphone. Richmond recommends that the communicator should have full control of the microphone at all times and that it has to be close to the mouth of the person that is speaking (approximately 6 to 13 inches away, as shown in Figure 3-1) [Richman, 2000].

Figure 3-1. Microphone position [adapted from Glass, 1999]

Headphones. Wearing the headphones while recording —and listening attentively and critically— enables the communicator to react early to different situations and avoid or quickly correct undesirable sound levels or irreparable noise. Richman recommends wearing headphones during the whole recording session [Richman, 2000].

Examples and counterexamples. Communicators should listen to examples of good recording practice, but also to several instances of disastrous results due to lack of constant attention, such as fingers rubbing the microphone, or the hum of a refrigerator.

Maintaining awareness. Communicators should acquire the habit of maintaining awareness, while recording, of volume levels, of audio quality, of sources of noise and, most important, of whether the recorder is recording. It is also wise to test by recording and playing back before initiating the interview but also immediately after finishing, just in case another take is necessary—and while everybody is still there to redo the recording.

The Produce stage

The *Produce* stage is when steps are taken to transform a field audio-recording into a worthy audio piece. Communicators usually spend most of their time on this phase. *Produce* mainly involves logging, composing and editing.

Logging

Logging consists of taking notes on a piece of paper of the location of the relevant parts of the recording. While listening to a recording composers 'mark' the 'in' point (beginning) and the 'out' point (end) of each selected part.

Composing

Composing refers to making decisions regarding their piece in terms of structure, sequence, tone, style, and background audio. The reordering and reshaping of the elements are mapped on a piece of paper by using the names of the parts and the edit points and written indications.

Editing

Editing is making with the recording the audio piece that was envisioned. It usually involves performing the following operations with audio chunks: move, cut, paste, copy, insert, and delete until we get a final audio piece. This part takes time, patience, and creativity.

The Publish stage

The 'Publish' stage is sharing the produced audio piece with others. Sharing can be done locally or remotely. Locally, it can be heard on the VoxPopBox and via FM/AM radio broadcast. Remotely, it can be via Internet, or by playing it on the remote VoxPopBox.

The Listen stage

'Listen' is here defined as paying attention to what was transmitted and providing the communicators who authored it with useful critique or feedback. [Rogoff, *et al.,* 2003. pp. 177-179] Ideally, listeners will close this proposed radio production cycle by becoming producers. The cycle starts again when the creators of the audio pieces and the listeners switch roles.

3.2.2 Audio piece

> "[Words] existed in a context, through a striking tale that easily accrued associations and depth." [Kohl, 1967. p. 25]

Audio pieces are the final product of the whole audio production process. There are many elements that the communicators of an audio piece have to deal with beforehand, consciously or unconsciously, directly or indirectly, in order to visualize what is to be done. The most important of these elements are briefly explained in Appendix A: Composition Elements.

3.3 Technology

The design and implementation of the technological tools considered the characteristics of the target users and the methodology previously described; this approach is called 'technomethodology' [Dourish and Button, 1998], which is particularly useful in this project due to the nature of the community.

The assumed characteristics of the target users were people coming from developing communities with little or no literacy nor computer skills. [Disessa, 2001]. For that reason, the system had to be reliable, inexpensive, plug-and-play, illiterate user-friendly, and culture adaptable with modifiable icons and translatable to their language, in case the user had to learn to recognize a few words in the menus. Given the characteristics of the eRadio technomethodology,

a set of existing devices and applications were selected, adapted, installed and integrated to each other to obtain the expected functionality and interface.

Following is a detailed description of the design and implementation of the system. The description begins with the hardware components. It then goes on to the system of the main hardware components, described from the perspective of user interaction. Then, it narrows down to the System and user interaction and finishes with the Base software components and the Software integration.

3.3.1 Hardware

Hardware requirements

The eRadio Project requires basically two electronic components. One is an inexpensive dedicated computer capable of receiving, storing, editing, organizing, and transmitting audio data. It should contain special user-friendly software, capable of being handled by users with no literacy and no computer skills. The other component is a portable digital audio recorder that can also be easily handled by the users described above. It should be able to handle large amounts of audio as well as computer connectivity features for downloading and uploading.

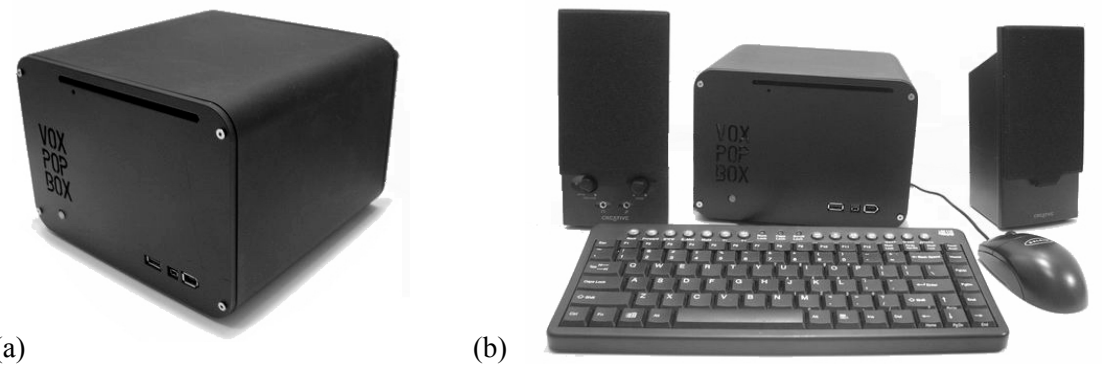
In the sections that follow, after specifying which equipment was utilized, we discuss desirable requirements as well as decisions made.

Hardware utilized

The eRadio hardware is a VoxPopBox minicomputer, an iRiver H120 digital encoder, a Sony ECM-MS907 stereo microphone, a pair of Sony MDR-V300 Studio Monitor Series Headphones, and a Veronica 1 watt FM radio transmitter coupled to a 5/8 Wave Colinear Vertical Antenna (CFM-95SL). The transmitter and antenna are optional, for example, for a rural community with no local radio station or no access to it.

VoxPopBox

The ‘VoxPop’ in VoxPopBox stands for the Latin etymology *vox populi* which literally means voice of the people [Merriam-Webster, 2000], and ‘Box’ is because it is a box, measuring 147mm (H) by 210mm (W) by 210mm (D), shown in Figure 3-2.



(a) (b)

Figure 3-2. (a) VoxPopBox; (b) VoxPopBox and peripherals

The VoxPopBox[2] is the core hardware of the Tulcingo-NewYork eRadio implementation. When outfitted with the eRadio software, it becomes a user-friendly, task oriented, broadcast-ready, data-and-audio electronic processor, and relatively small and economical. It consists of a mini-ITX board, housed in a portable simple case, which has only the essential external ports. [http://mini-itx.com/] These specifications are particularly well suited for deployment and use in rural communities, but most Intel-compatible PCs with similar specifications could be used. The box is fitted with a minimum of off-the-shelf peripherals. Its main purpose is to simplify the making of small audio pieces, easy downloading, uploading, recording, editing, encoding, archiving, and transmitting (via Internet or radio) small audio pieces and audio projects or radio programs. The peripherals of the VoxPopBox include keyboard, mouse, microphone, headphones, and a pair of amplified speakers. It can display via a computer monitor or a TV set (PAL or NTSC) for a minimal deployment.

The portable digital recorder

In order to gather audio from the community, users need to become un-tethered from the VoxPopBox. We chose a digital audio recorder, which could store a large number of high-quality recordings in the field but would allow the user to return to the box and easily connect the recorder for seamless download to the box. The iRiver H120 is one of the few portable digital audio recorders which has large storage capacity, records to professional-grade audio formats, has built-in Automatic Gain Control for simplified recording, and provides a standard USB Mass

[2] **The technical specifications of the VoxPopBox are the following:** ***Processor***: VIA 1GHz Nehemiah C3 Mini-ITX MII; ***Memory***: Crucial 512MB DDR400 Memory; ***Hard Drive***: 80GB Ultra ATA/133 7200 RPM Hard Drive; ***Video Card***: Integrated VIA Unichrome AGP graphics with MPEG-2 decoder; ***Sound Card***: VIA Vinyl Six-TRAC 5.1 Surround Sound; ***CD/DVD ROM drive***: DVD/CD-RW combo drive (QuickSlot tray loading system.); ***External Connectors***: 3 x USB 2.0, 1 x Firewire (6-pin), 1 x Mini-Firewire (4-pin), 10/100 Ethernet, 2 x PS2, Serial, Parallel, VGA (video), Audio, S-Video out, SPDIF (digital audio)/TV-out; Consumer Infrared for PC control. [http://hoojum.com/html/product_cubit3.htm]

Storage Device interface. It can store up to 20 Gigabytes, which translates into hours of recordings. It encodes directly to MP3, which is commonly supported and allows additional storage capacity and faster transfer speeds through compression. The MP3 of human voice at 44.1 Khz sample rate, 128 Kbps, 16-bit sample size is nearly indistinguishable from uncompressed audio and is approximately one twelfth the size. This format is commonly used in professional radio production. The automatic gain control eliminates the often-confusing step of adjusting levels while recording. The recorder's USB Mass Storage Device interface allowed us to easily implement the downloading logic on the VoxPopBox.

3.3.2 System and user interaction

This section describes some characteristics of the system in relation to the user, that is, system and user interaction. The tool guides the user through a process, which consists of intermediate tasks which can be started, stopped and resumed at almost any time. The goal was to guide the user through what is largely a linear process while recognizing that the creative process, mental models, and outside factors do not necessarily proceed directly from beginning to end in a single limited amount of time. Thus, the interface was divided into four Task areas represented by a bar of visual symbols which flow linearly, but can be selected at will. Within each area, the user is presented with simplified tools for completing the task at hand. Users can become comfortable with one area before moving to another, ideally mastering all areas and eventually switching between them with ease. The interface elements are designed for the user with little or no reading skills and with no experience in using a computer or in audio production.

In this first stage, we had limited iterations in which to refine the tool, but a long-term goal should be to integrate lessons learned from user experience to increase overall usability.

The one step away feel

The tool is designed in such a way that novice users feel that they are one step away from their desired end, and they never get lost or forget what to do next. That makes the system easy to learn [Joshi and Rasai, 2002]. At all times the user can see their current status directly because it is visually connected to the end goal in the Task menu-bar.

User management

As with the radio production process, within the system users have different profiles. These profiles are set when first assigning users into the system, and from then on they can access their personal profile with a user name and a password. There are three different profiles. First, everyone can be a communicator if they carry out gathering, producing, or publishing tasks. The

published audio piece is only accessible to the communicator who authored it and to the editor. Second, the *editor* is the user in charge of administrating the system locally and of publishing the reviewed and accepted audio pieces on the editor's system. In fact, there can be a 'remote' editor or even a bilateral editorial committee composed of 'locals' and 'remotes.' In this case, publishing means that the audio piece is not just on the local VoxPopBox, but also on the 'remote' editor's system. The editor is the publisher and can also be the broadcaster. Third, the *broadcaster* has access to the pieces that were already reviewed and OK'd for broadcasting and is in charge of broadcasting them on the Internet. All user files—clips, projects, etc.—are stored separately on the file system for each user. This way, each user has his or her own unique environment, which can be left off and resumed again and again, allowing multiple users to operate (non-concurrently) without interfering with each others' tasks.

User experience and Task areas

This section describes the interaction of the users with the electronic tool while they go through the radio production process. The system, mirroring the radio production process, has four Task areas that represent the stages of the process, that is, *Gather*, *Produce*, *Publish*, and *Listen*. Each profile is a fixed Task area accessed by a Task menu-bar. The Task menu-bar also has four buttons, as shown in Figure 3-3. When the users switch between Task areas, a label indicates what stage of the process they are in.

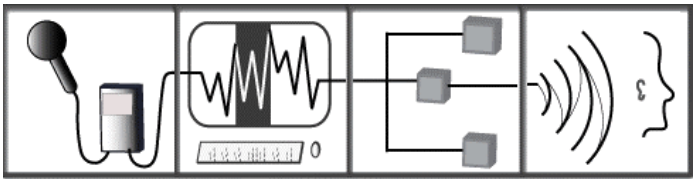


Figure 3-3. Task menu-bar

The modified Digital recorder

The primary drawback of the iRiver H120 was its complicated user interface. We solved this problem by altering the controls to remove any unnecessary functionality. We removed the jog handle (for multi-function control) and a mode button. After the modification, users could only record, pause, stop, and play back the last recorded clip. This effectively locked the audio format settings and prevented users from entering the confusing menu system. In Figure 3-4 we see the recorder before modification.

Figure 3-4. The iRiver H120

The modified recorder has three buttons: Play to turn it on and playback the last recording; Record to start, pause, and resume recording, and to switch from play to record mode by pressing the button for a second; and the Stop button is to stop play and record modes and to turn off the recorder by pressing the button for a second. The more advanced functions of the recorder still can be carried out through the remote control.

To record, the user plugs the microphone into the recorder, turns both of them on and starts recording. Once the recordings are done, the recorder is connected to the VoxPopBox with a USB cable. In the following section we see what happens next.

The VoxPopBox

The 'Gather' task-area. The moment the recorder is connected to the USB slot of the VoxPopBox, the system detects the recorder and automatically reads the audio files and places them in the 'Gather' task-area, and removes them from the recorder, leaving it empty for the next user. The 'Gather' task-area, as shown in Figure 3-5, is divided into two panes: Clips and Projects. In the Clips section the users see the clips (recordings), which can be listened to with a single click. The user can drag any number of clips onto any project in the Projects pane, 'adding' the clips to the project and preparing the user to edit that set of clips in the 'Produce' task-area. More details regarding the 'Gather' task-area are given in section 2.3.4.

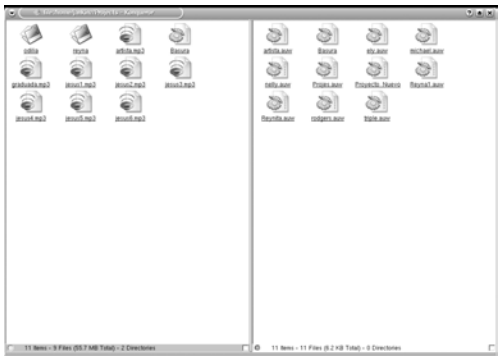


Figure 3-5. *Gather* task-area

Users can click on any project to be switched to the *Produce* task-area with that project open for editing.

The 'Produce' task-area. The 'Produce' task-area is displayed as audio waveforms, where logging and editing have to be performed.

The waveform

With practice, users develop the ability to distinguish broad categories of audio by looking at the visual waveforms. Thus, the editing process can be done more rapidly because visual searching makes finding the wanted parts easier and faster, without having to re-listen to the whole piece.

As shown in Figure 3-6, a flat line is silence, small waves may be noise or someone breathing, irregular waves of greater amplitude probably are people speaking, and long periodical and condensed waves are likely to be certain kinds of music.

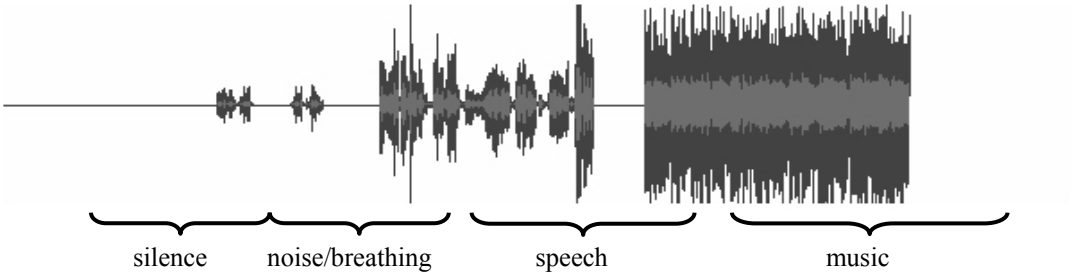


Figure 3-6. Audacity waveforms

Producing

Producing is to edit a recording by modifying it, reordering it, and ornamenting it in order to achieve a desired audio piece. This is done by 'selecting' with the mouse the chosen part of the waveform. Then by using the iconic editing tools, users, as they listen, can do the following one-step basic editing tasks known as play, move, copy, cut, paste, delete, and insert. Complex tasks of more than one step are also possible, such as increasing or reducing volume levels and doing fade-ins and fade-outs by using Audacity's 'Effect Menu.' In future implementations tasks of this kind will be activated via an iconic menu. Some or all of the operations can be done on different tracks and then put together onto one or two tracks for mono or stereo. At any stage copies can be made of parts or the whole track to experiment making variations of the same track. Upon completing an editing step users can listen repeatedly to verify results. There exist convenient "Undo"

and "Redo" features. At regular intervals and when the user exits the Task area, the project is saved invisibly.

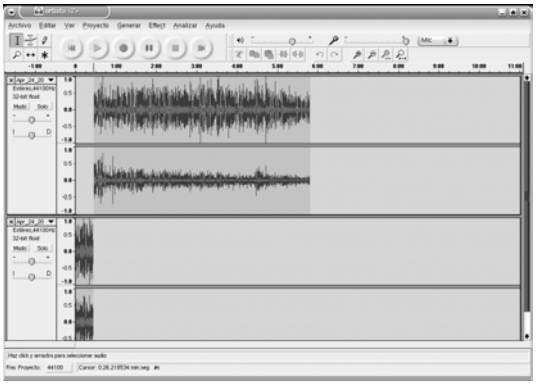


Figure 3-7. 'Produce' task-area (Audacity)

The 'Publish' task-area. The 'Publish' task-area lists each of the user's projects and allows the user to choose whether or not to publish each one of them. It allows users to enter with the keyboard an optional description of a project and press "publish." At that time, the system creates a single MP3 file from the multi-track project. The original project file is not deleted so that the user may return at any time to perform more editing and then re-publishing. The MP3 file along with its description is called a 'piece' and is treated like a "final cut." Pieces are accessible, through the 'Listen' task-area, to their authors, their fellow users, and 'Editors' and potentially to the world.

Depending on how the system is configured, publication may trigger distribution to other VoxPopBoxes, which are networked to the local box. This distribution may occur during off-peak hours to minimize network load and cost. As explained later in this section under "Networking-publishing," in the pilot implementation, audio pieces were manually synchronized so that people at both sites could listen to them.

The 'Listen' task-area. The 'Listen' task-area was implemented in limited fashion for the pilot, but is envisioned as structured in logical trees in a file manager. The local and networked sites are represented as VoxPopBoxes at the top level, with users underneath them. When clicking on a site, it allows reading piece descriptions and listening to published pieces. One can envision a network of interlinked VoxPopBoxes, each of which can listen to the published content of the others. In the broadcaster role, the VoxPopBox audio output is connected to the transmitter or broadcasting console.

3.3.3 Base software components

Our goal was to start with 'off the shelf,' free, open-source components and modify or configure them to collectively fulfill the requirements of each Task area. This allowed us to start with established software packages and then remove or abstract away unnecessary or distracting elements, a strategy similar to our recorder hardware modifications. Because most components were under a GNU Public License or similar license, the final eRadio software package mostly consists of patch files, configuration files, and instructions. In most cases these are necessarily open source. These files are put together at: http://eradio.media.mit.edu/download.html.

Some components performed poorly in the Tulcingo workshop, so we replaced, modified or debugged them for the New York workshop, as noted in corresponding sections.

Operating system and Distribution

We required an open operating system that had components we could easily modify, powerful audio support and tools, and compatibility with Mini-ITX motherboards and on-board components. We ultimately selected Linux because of the many compatible open-source software packages that supported our needs, like graphical file browsing, waveform audio editing, and virtual desktops. We then chose the Knoppix GNU/Linux distribution because it auto-detects the Mini-ITX hardware well, provides a simple 1-CD install, is based on the well-maintained Debian GNU/Linux distribution, and inherit's Debian's 'apt-get' tool for easy addition of software packages. Knoppix comes with KDE 3, and using 'apt-get' we were able to easily add Audacity, apache, hotplug and several other packages we required.

The Window system and Window manager

Our goals with the Graphical User Interface (GUI) were to make it simple and intuitive, presenting the user with only the functions required to complete the task at hand. In order to represent the four Task areas, we required four different full screens, each with their own stateful interfaces that could be switched between at any time (Task areas with their Task menu-bar are shown in Figure 3-8).

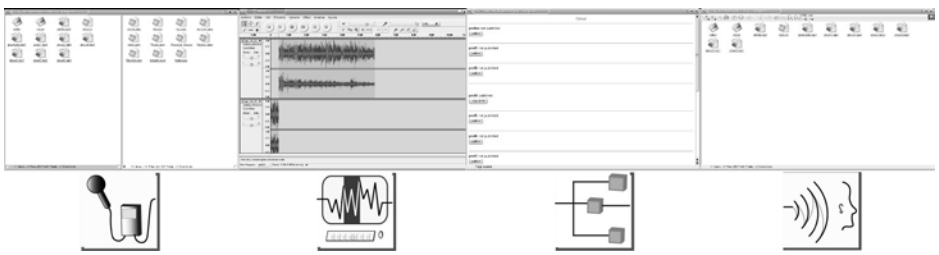


Figure 3-8. Windowing System

We chose the X Window System ("X11") [http://www.x.org/], and used the XFree86 implementation [http://www.xfree86.org/]. When combined with a window manager, X11 provides "virtual desktop" functionality. This allowed the system, upon user login, to pre-create four independent screens which could then be populated with various software components for each Task area.

We initially chose Windowmaker [http://www.windowmaker.org/] for the window manager. Windowmaker is configurable through files, is relatively free of visual "clutter," out of the box, and can be made to perform virtual desktop switching from other applications. Windowmaker does not have a rich Application Programmer's Interface (API) or event system for starting application windows on particular virtual desktops. This required some non-standard techniques for pre-populating and switching between desktops, which proved unreliable in the Tulcingo workshop.

For the New York workshop, we re-evaluated window managers and chose the KDE desktop environment [http://www.kde.org/] which provides a window manager and several other integrated applications. KDE provides an easy and reliable interface for opening application windows on different desktops and for triggering other events.

Graphical file browser

For the 'Gather' task-area, we needed basic file browsing functionality, including click-to-open, drag-and-drop, and rename. KDE includes a versatile application called 'Konqueror', which can function in file browser mode and can be configured to show multiple "panes" and to hide advanced functionality [http://konqueror.kde.org/]. We configured Konqueror to hide all toolbars and show two panes: one containing "Clips" (MP3 files) and one for "Projects" (Audacity "aup" files). The Konqueror window was also "maximized" to fill the whole Task area (except for the Task menu-bar), as shown in Figure 3-5. When users single-clicked on Clips they would play using the program 'x11amp', but any MP3 player could be used. We created unix shell scripts for performing simple tasks like "create new project" or "delete clip" and added these to the panes. The scripts were triggered through click or drag-and-drop and are discussed in section 3.2.4.

Waveform audio editor

'Produce,' the most complex and time-consuming Task area for users, required a visual representation of all clips in the project along with simple tools for combining them into a completed piece. Multi-track waveform-based editing remains the dominant paradigm for software audio editing but most software packages are complex and professionally oriented. We

chose ‘Audacity’, a mature open-source digital audio editor. [http://audacity.sourceforge.net/]. Audacity is relatively simple compared to most digital audio editors. However, tasks such as “saving,” “opening,” and “importing” require menu-selection, hierarchical browsing and file system comprehension which users could not be assumed to have or learn. Thus we made several modifications to the Audacity source code (described in section 3.2.4) to allow for “signaling” from other components to trigger these actions in Audacity when appropriate.

Audacity’s limited workload capacity

During the Tulcingo workshop, Audacity slowed to an almost unusable state while editing some pieces. We realized that this only occurred when working with very large files, in this case files of multiple Gigabytes. In the two weeks between the Tulcingo and New York workshops, we obtained some explanations. The three main causes for system slowdown are fragmented file storage, audio data memory and temporary file overloading, and re-sampling. Schmandt points out that editing produces “many small snippets of sound that result in fragmented files” [Schmandt, 1994. p. 319] and that fragmentation by itself slows down the system. Richard Ash, a developer of Audacity, says that every time a track is played back, Audacity loads the audio data from the disk and that it re-samples it [Ash, 2004]. Sipitakiat adds that higher resolutions and stereo mode also increase the size of the temporary files generated by Audacity [Sipitakiat, 2004].

Avoiding Overload

The first lesson learned is to work with as few tracks or clips as possible and to remove unused portions during editing. This reduces audio data load on memory and on disk. Second, when recording, we must remember to press “Stop” at natural stopping points in order to create multiple files which are smaller and less processor and memory intensive than the single large file created if we only press “Pause” at those points. Third, recording in mono will halve file size and is often indiscernible from stereo when recording the human voice. Finally, the recorder should be set to record at the same sample rate as the Audacity project, avoiding a re-sampling step. One could also reduce sampling rate or sampling size, but this would risk significant audio quality degradation.

Task menu-bar

The Task menu-bar is a tool for navigating between Task areas, which also provides a reference point for the user’s current location in the piece creation process. It is positioned in a fixed location and persists across all Task areas.

When using Windowmaker, we chose the open-source application ‘AcidLaunch’, which allowed us to define iconic buttons via an XML configuration interface. These buttons switched

between Task areas. The AcidLaunch-based Task menu-bar is shown in Figure 3-3. However, the AcidLaunch and Windowmaker combination did not enable us to ensure that the bar would be placed in the same location on all monitor configurations, causing significant malfunctions and usability problems, as was the case in Tulcingo.

Since our target users were perfectly literate, for the New York workshop, we used the built-in KDE text-based Task menu-bar, which did not incorporate the iconic buttons but was significantly more reliable. A reliable icon-based Task menu-bar would have fully met the initial project goals, and should be considered for future work. The KDE-based Task menu-bar is shown in Figure 3-9.

Figure 3-9. Text-based Task menu-bar

Publishing interface

The publishing interface was minimal for the workshops. It provided a list of current projects with "publish" or "unpublish" options. We used a script written in 'PHP: Hypertext Preprocessor' language running on a local Apache web server to provide this interface and to operate on the user's files (http://eradio.media.mit.edu/download.html). If several boxes are networked, when one user checks the "publish" option for a given project, PHP tells Audacity to export a final MP3 of that project and place it on the file system, making the file accessible to other users of the same box via their 'Listen' task-area and to the users of the other boxes (details in Audacity Modifications in section 3.3.4). In the workshops, files were manually made available via the Internet on PRX, but we envision these operations taking place automatically in the future. And future work should also include a more rich set of metadata for completed pieces.

Listening interface

We again used Konqueror for the Listening interface, which consisted of a simplified file browsing window. When several boxes are networked, users can single-click on any other box within their network, then single-click on any user associated with one of those boxes, and then single-click on any of that user's published pieces. This causes the audio file to play with 'x11amp,' a popular open-source MP3 player. Several improvements can be made to the Listening interface to ensure that network-box-user-piece browsing is more intuitive, but the fundamentally hierarchical nature of graphical file browsing served well for the purposes of this study.

User management and Login window

Users need to be able to log into the system in order to work on their content uniquely. When no users are logged in, the system presents a Login Window. KDE and Window Maker each provide a simple login interface with fields for username and password along with a visual identifier. All user administration was done using the built-in Linux user management system.

Networking–publishing

Each individual box requires a network connection to the other boxes for publishing beyond the local site. Although this functionality was not entirely functional for the workshops, it was implemented in limited form. The Tulcingo box was networked via a sporadic dialup connection and the New York box was networked via DSL.

In order to allow people at each site to listen to pieces from the other, directories containing published pieces on the two boxes were occasionally 'pushed' from one box to a central Internet host. These files were then 'pulled' by a user at the other site, at which point the files are said to be 'synchronized.' We chose the software package 'rsync' [http://samba.anu.edu.au/rsync/] to provide this functionality. Rsync performs integrity checksums to guarantee that files are transferred successfully and can operate recursively on directory trees. This allowed us to synchronize the files manually overnight to make them available to the other site during the day.

The Public Radio Exchange (PRX) is a non-profit web service, which allows members to post pieces for listening, review, license, and download. PRX servers provided the central Internet host for file synchronization and allowed us to manually make the pieces available to the broader public and broaden exposure of the published pieces [http://prx.org/].

Networking – for administration

When the boxes are at least sporadically accessible via the public Internet, administrators are able to access the boxes to perform tasks such as user adding, system administration, and file synchronization. This is performed via the encrypted Secure Shell protocol (SSH). We chose the OpenSSH suite [http://www.openssh.com/] because it was freely available and easy to install via apt-get.

3.3.4 Software integration

Several areas of functionality required more in-depth software integration. These efforts are described below and specific implementation files are provided via the web site (http://eradio.media.mit.edu/download.html).

Audio file downloading from recorder

The project required a mechanism to trigger a seamless download of audio files from the digital recorder without any user intervention other than plugging the recorder into the box. In order to achieve this, we used the linux-hotplug package [http://linux-hotplug.sourceforge.net/] which allowed us to write custom scripts to perform actions every time the recorder was plugged in. Upon being plugged in, the recorder is recognized as a USB Mass Storage Device. This triggers a custom script to mount the recorder on the file system, move all audio files to the 'clips' subdirectory in that user's home directory, and unmount the device. At this point, the Konqueror-based 'Clips' pane of the user's 'Gather' task-area shows the new files along with any files that were already there. The user is then free to begin editing, using all available clips or return to the field to gather more.

File structure

Each user's files are stored in a separate 'home' directory on the Linux file system. Their files are further subdivided according to type. Within each home directory is a 'Clips' directory, which contains any raw MP3 audio files gathered by that user. When the user creates a new project, a complete small metadata file is placed in a similar 'Files' directory and Audacity creates '.aup' files (AUPs) to hold project status and corresponding '_data' directories with all audio data. An '.auw' file (AUW) is also placed in a 'Projects' directory and is the user's single method for interacting with these various project files, as described in "Scripting" below. When a project is published, a single MP3 file is created and placed in the user's 'Files' directory.

Scripting

The following scripts written in Perl and Bash were used for the following tasks and are available from the web site (http://eradio.media.mit.edu/download.html):

→ In each 'projects' directory:

- ***New_Project*** prompts the user for a project name and then creates a new AUP, data directory, AUW, and metadata file.
- ***Trash*** allows users to drag an AUW onto this icon to delete the AUW.
- ***AUW files*** can be clicked on in order to open the corresponding AUP file in the Produce task-area. Also, by dragging a clip (MP3) onto the AUW, the user can import the clip into the corresponding AUP, making the clip available for editing within the project. AUW files accomplish this by sending to any open Audacity process signals to save and quit

before opening others. AUW files also call Audacity with our custom options in order to allow clip importing when opening a project.

→ In each 'clips' directory:

- ***Trash*** allows users to drag a clip onto this icon to delete the clip.

→ In /etc/hotplug/usb/

- ***usb-storage*** contains all the logic for mounting the recorder as a USB Mass Storage device, moving the audio files to the 'clips' directory, and un-mounting the device.

Audacity Modifications

→ "no check" option: We modified Audacity to accept the command-line option "-nc" that would prevent it from checking whether another Audacity process was running. This allowed us to call our customized Audacity binary in order to perform operations such as "import" without generating errors if another project was simultaneously open.

→ "import" option: We modified Audacity to accept the command-line option "-i" followed by an MP3 filename and an AUP filename. This would cause Audacity to open the AUP and import the MP3, making the MP3 available for editing within that project.

→ "export" option: We modified Audacity to accept the command-line option "-e" with filename after it. This would cause Audacity to open the project named, export it to an MP3, and quit.

→ "save" signal: We modified Audacity to listen for the signal SIGUSR1 and, upon receiving the signal, to save any open projects.

→ "quit" signal: We modified Audacity to listen for the signal SIGUSR2 and, upon receiving the signal, to quit any open projects. In combination with the above options and signals, this allowed us to open, import, export, save, and quit Audacity projects without requiring the user to perform these actions explicitly.

User Environment Configuration

Many user environment settings for KDE and Audacity were placed in files in the /etc/skel directory so that all users created on the box would share these settings.

Interface Customization

We used the following built-in KDE functions for controlling the interface:

→ ***kstart.*** This is a program that allows you to call any other program with KDE-specific settings. For example, we used '--maximize' to force the windows to fill the screen and '—desktop <num>' to specify on which desktop window it should appear.

→ ***Konqueror*** profiles. Konqueror can be called with a '—profile' parameter which loads a user-customizable profile to enable or disable toolbars, enter web or file browser modes, split into panes, and open specific directories in panes. For example, we used the command 'kstart --desktop 1 --skiptaskbar --maximize konqueror --profile gather' to initialize the 'Gather' task-area.

→ ***Autostart***: This file inside the .kde directory is executed every time the user logs in. We placed environment initialization commands here, including calls to konqueror and audacity via kstart.