2009

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

GNU RADIO AND THE USRP AS A SOLUTION FOR REMOTE EMERGENCY MONITORING

Douglas Casey Tucker

A Capstone Project Submitted to the
University of North Carolina at Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management
University of North Carolina at Wilmington

2009

Approved by

Advisory Committee

Dr. Thomas Janicki

Dr. Bryan Reinicke

Richard Linhart

Dr. Gene Tagliarini, Chair

Accepted By

Dean, Graduate School

Abstract

GNU Radio and the USRP as a solution for remote emergency monitoring. Douglas Casey Tucker, 2009. Capstone Paper, University of North Carolina at Wilmington.

Software-Defined Radio (SDR) is a field that has emerged to solve the problem of interoperability between differing radio frequency (RF) communication standards. Instead of requiring different hardware for each waveform as in traditional analog radio, SDR attempts to provide a more hardware-independent solution by moving the software closer to the antenna. In this document the author presents the case of an information providing company whose goal is to expand into remote locations, the evolution of its operations from low-tech to high-tech automated information systems, and its interest in a particular application of SDR in conjunction with streaming audio and Voice over IP (VoIP) services.

Table of Contents

Acknowledgements

First and foremost I would like to acknowledge the support of my family who have encouraged and supported me throughout my life as a student and an aspiring professional. My father has always supported me in the development of my skills from a very early age by making sure I have access to current technology, and my mother has always made sure that I'm always learning. Thanks, Mom and Dad.

I would also like to thank Jim Stevens, without whom I would not have gained the opportunity to develop my skills in the radio industry, and Dale Miller for allowing me to take on more responsibility at the radio station. I'd also like to thank Jay Stevens who has always been an honest manager, has supported my research, has always had the utmost confidence in my abilities, and has only had good things to say about me to those with whom he works.

Finally I want to thank the patient and knowledgeable staff of UNCW, Dr. Sridhar Narayan for his support and encouragement, Dr. Gene Tagliarini for having faith in my abilities and sending me to IEEE SouthEastCon 2009, Dr. Brian Reinicke for his professional insights, and Dr. Devon Simmonds for getting me into the habit of reading academic research papers. I also need to extend thanks to Dr. Tom Hudson for expecting and inspiring excellence in my undergraduate network programming course.

Thanks, all!

1. Introduction

Wilmington Traffic on Demand has been in business since the first quarter of 2006. Their operations revolve around obtaining information from various sources about local road conditions and traffic incidents, and delivering this information to local broadcasting affiliates in audio form. This paper will discuss the development of this company from its early stages, its attempts to expand into other markets, and the development of the technological infrastructure of the company. The primary research presented is the development of a system that enables listening to police and emergency radio communications at remote locations via Internet using software-defined radio (SRD) techniques.

Prior techniques using analog radios have resulted in less maintainable systems when deployed at remote locations. As demonstrated in the following sections, SDR allows for more maintainable, more robust, and more standardized deployments of systems that are capable of listening to terrestrial radio signals present in remote locations.

What follows will provide context of the company's general operations, describe the reengineering processes that enabled its expansion efforts, discuss the research involving a specific software-defined radio application and evaluate alternative technologies, provide quantitative analysis of the improvements introduced via the reengineering process, and conclude with a timeline showing the life-cycle of the development process.

2. Operations

Wilmington Traffic on Demand was initially formed with a simple idea: to provide a third-party source to local broadcasters for obtaining traffic information reports. In order to monetize this idea, ten-to-fifteen-second sponsorships are added to the end of each report. During a normal weekday, an announcer will continuously gather information, produce a 30-second audio recording of the traffic conditions and sponsorship, and send this report to the designated radio station. There are several shifts that are considered "drive-times" during which these reports are recorded and transmitted to broadcasters: 6:00am to 10:00am, lunch-hour at noon, and 3:00pm to 6:00pm. Before each working day, a log must be generated so that the announcers know the order in which to record and deliver reports to various stations.

During the initial operating phases, the demand was small enough to be managed by the two founders of the company. A frequency of three reports per hour was common for all broadcasters during drive times. With five initial affiliate broadcasters, this totaled fifteen 30-second reports per hour, split between two announcers, leaving plenty of time to manage the business, make sales calls, and gather information between reports.

As it is the nature of business to expand and grow, the company eventually acquired more affiliate broadcasters enabling greater sales, generating additional revenue. In order to manage this demand, the company hired a sales staff member to manage revenue-generating activities, allowing the president to focus less on sales and more on reporting, and allowing the operations manager to focus more on maintaining relationships with affiliates and observing the operations of the company.

The author's role in the company has been infrastructure support since day one. The first task was to build a website via which they could market their idea, and to provide a central pickup-and-drop-off point for the audio traffic reports they produce. The initial operational technologies were not very robust, and consisted of common commercial software such as Microsoft Windows XP and Office 2003, de-facto radio production software such as Cool Edit Pro 2.0, and open-source FTP Client software FileZilla. These tools were used somewhat inefficiently to enable the reporter to keep track of sponsorship data, produce an audio recording, and transmit the recording to the central web-server. As the company expanded its network, there arose a need to refine the operations by progressively introducing customized software. The process of developing and introducing this software is detailed in the Reengineering section.

Its current operations are more finely tuned such that announcers are able to focus on originating reports via a database-driven operations interface known as Traffic Dealer. Because announcers are able to send out more reports, the company is able to acquire more affiliate broadcasters while hiring fewer announcers. Analysis of the efficiency improvement is presented in Appendix B.

3. Reengineering

The reengineering process consisted of several phases. This section will discuss the changes that were made in a chronological fashion, beginning with the inefficient the pen-and-paper method and the migration to a Microsoft Access database solution, and concluding with the eventual move from the decentralized Access database to the current MySQL-based client-server database architecture.

3.1 COMPUTERIZATION OF MANUAL PROCESSES

Inefficiencies arose due to the disjointed nature of the data; separate document entities with no unifying database. This was the first point of entry for customized software. Since the workstations were already equipped with Microsoft Access 2003, it became the platform for the operations management software. Access was also a good choice due to the fact that it offers Rapid Application Development (RAD) to take place, by its WYSIWIG development interface for forms, reports, and database entity relations. Figure 1 shows the domain-classes or tables for the initial database design.

The stations to which the reports are to be sent are stored in the `Station` table. Each station has a number of time-slots available for advertising, stored as `Avails`. Every `Sponsor` is recorded with their contact information, and every sponsor can have a number of `Orders` that it places with Traffic on Demand to gain advertising space. Every sponsor also has several `Copy` scripts, which are read by the announcer. Since there are various types of stations (e.g. Commercial, Non-profit, TV), each `Copy` is assigned a `StationType`, and when generating a daily `Log`, stations of a given type will only be assigned copy scripts of matching type. There will be one or more announcers working during the day, so in order to separate who is broadcasting to

whom, each `Log` entry refers to an `Announcer`. To aid in the process of assigning `Log` entries to `Announcers` and splitting responsibilities, each `Station` has a default announcer assigned to it that will be used during `Log` generation.

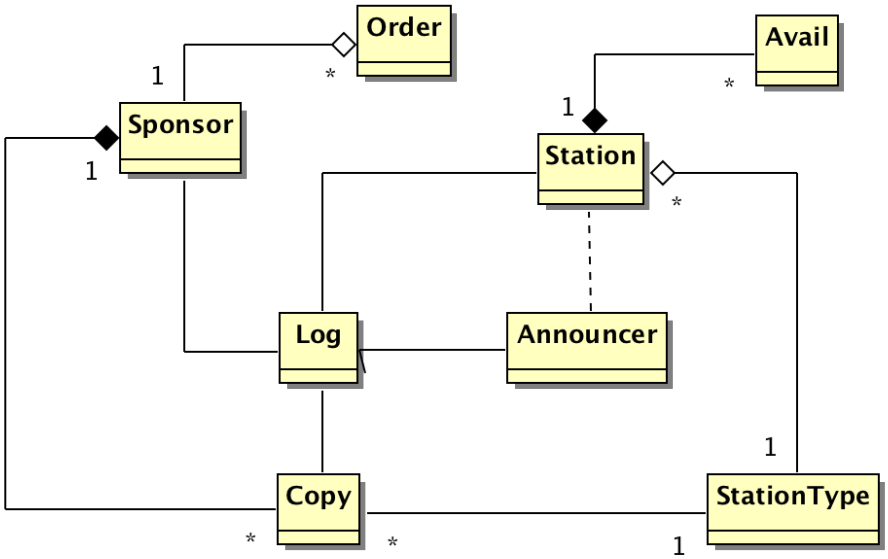

Figure 1: Domain class diagram of the initial database

The author was hired as a reporter for a period of two weeks, which provided insight regarding possible operational improvements, and offered an opportunity for him to undertake on-site development and testing. Without this experience, it is possible that more time would have been needed after deployment to make the product usable.

Rather than having a paper log of times when broadcasters need to receive reports, the database system provides a heads-up style interface to direct the announcer. This interface includes color-scheming to alert the announcer of upcoming deadlines during the day, using green to indicate at least five minutes of time before the next report, yellow to indicate that a report is due in five minutes or less, and red to indicate that the report is due to have been transmitted already. Another important feature of the interface is its ability to launch the audio production software such that it will overwrite the correct file for transmission. This dramatically reduced the amount of

time spent on routine loading and saving activities. By correctly overwriting the appropriate file, the interface's transmit button, when pressed, makes use of scripting tools to automate the function of sending the appropriate report to the broadcaster's drop-off point, further reducing the time needed from start-to-finish to originate a report. After starting the program, the reporter first selects his/her name from the Announcer drop-down to access the appropriate set of tasks. The reporter can then go about daily recording and transmitting duties by scrolling through the timeslots (see Figure 2)

Another major factor of improvement was in the generation of the daily logs that the reporters use to keep track of when reports are due, as it takes several human hours to generate, but only a matter of seconds for a computer. A more quantitative analysis of the timesavings is presented in Appendix B.

It later became important for the managers to generate invoices to send out to their expanding list of sponsors. Since Access provides reporting features, the author was able to develop an invoice feature using these features combined with Visual Basic for Applications (VBA) incorporating the company logo, and reporting to the customer when sponsorships occurred in a given month. This extra functionality's database entities are shown in the Figure 3 - perhaps counter-intuitively, the author chose the name `InvoiceData` to represent the header of a given invoice, whereas the name `Invoice` was chosen to represent multiple lines of description content to be printed on a given invoice.

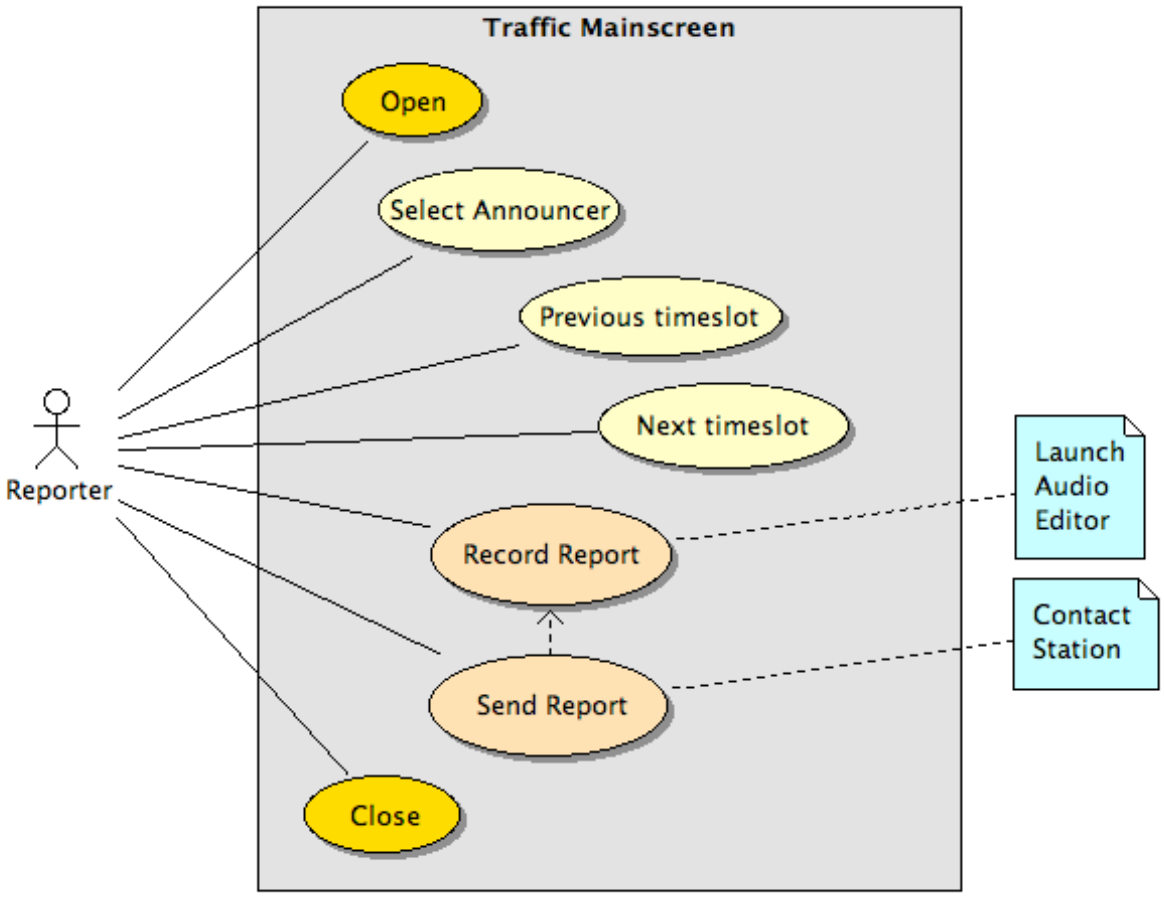

Figure 2: Use cases for reporter operations

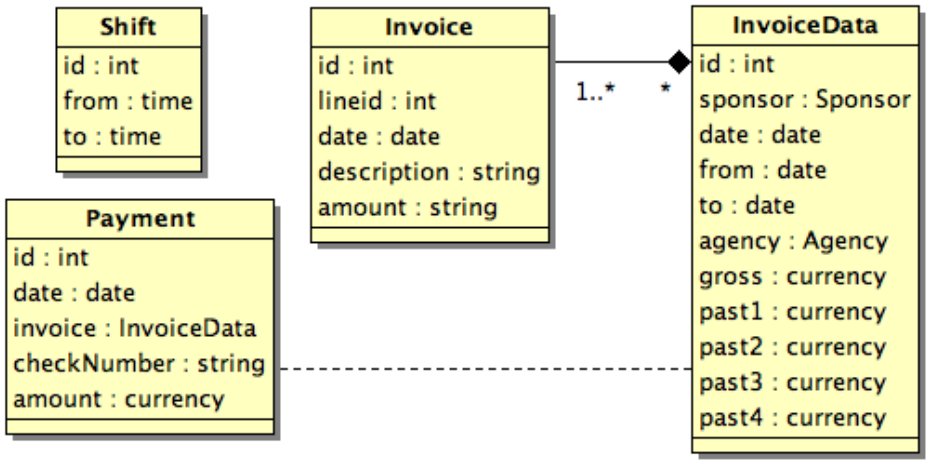

Figure 3: Invoice, payment, and shifts functionality domain classes

3.2 Migration from Access to MySQL

There arose a need to migrate from Microsoft Access due to limitations of the platform. The main issue was that the database needed to be accessed on multiple workstations simultaneously, and although Access can connect to a database file (MDB) located on a Windows file share of another computer, it can only withstand four simultaneous connections before the system grinds to a halt. Having hired an additional salesperson, this issue began to show itself even with only three workstations connected, and anticipating hiring additional reporters, the issue had to be resolved for operations to continue at a steady pace.

The author investigated Open Source Software (OSS) solutions and decided to use MySQL as the new database platform. Although the data migration effort could have been simplified by choosing a Microsoft product such as SQL Server, licensing fees would have driven up the overall cost of the new system. The author was able to find free utilities to export data from the MDB file into plaintext CSV tables. Using this format made it reasonably simple to import the old data into the new MySQL database.

The development process for the new system began immediately after purchase of new server equipment. A RAID-1 configuration was chosen for storage using three identical Western Digital YS-series hard disks. The base operating system is Arch Linux [1], which carries the benefit of good Wiki documentation, and the philosophy of a rolling-release distribution method ensuring availability of up-to-date software and easy upgrade procedures. A web interface was the logical choice for accessing the database, and its development involved PHP 5 and an OSS library called SAJA to provide

dynamic AJAX functionality for updates and insertions to the database. To validate user-input, regular expressions were put in place within the SAJA callback functions.

To provide invoice-generation capability, having lost the intrinsic reporting capabilities of Access, a new method was constructed using the fPDF library. This actually had more benefits than Access could have provided, as PDF generation is not built into access, and would have required third-party software. In addition, a greater degree of flexibility was provided by directly coding the reports. With this new system it became easier to send invoices to clients, as the PDFs can be sent via email, or alternatively printed and sent via post.

To deal with financial information, spreadsheets had previously been used, but this proved to be error-prone. Several new tables were introduced into the database, one of which would keep track of account receivable. The Payment table was created for this purpose, with each entry referring to a previously generated invoice, and an additional field to record the check number.

Another feature of the new system is that it allows the operations manager to define `Shifts` during which various announcers are present, simplifying the Log generation and Announcer assignment routines.

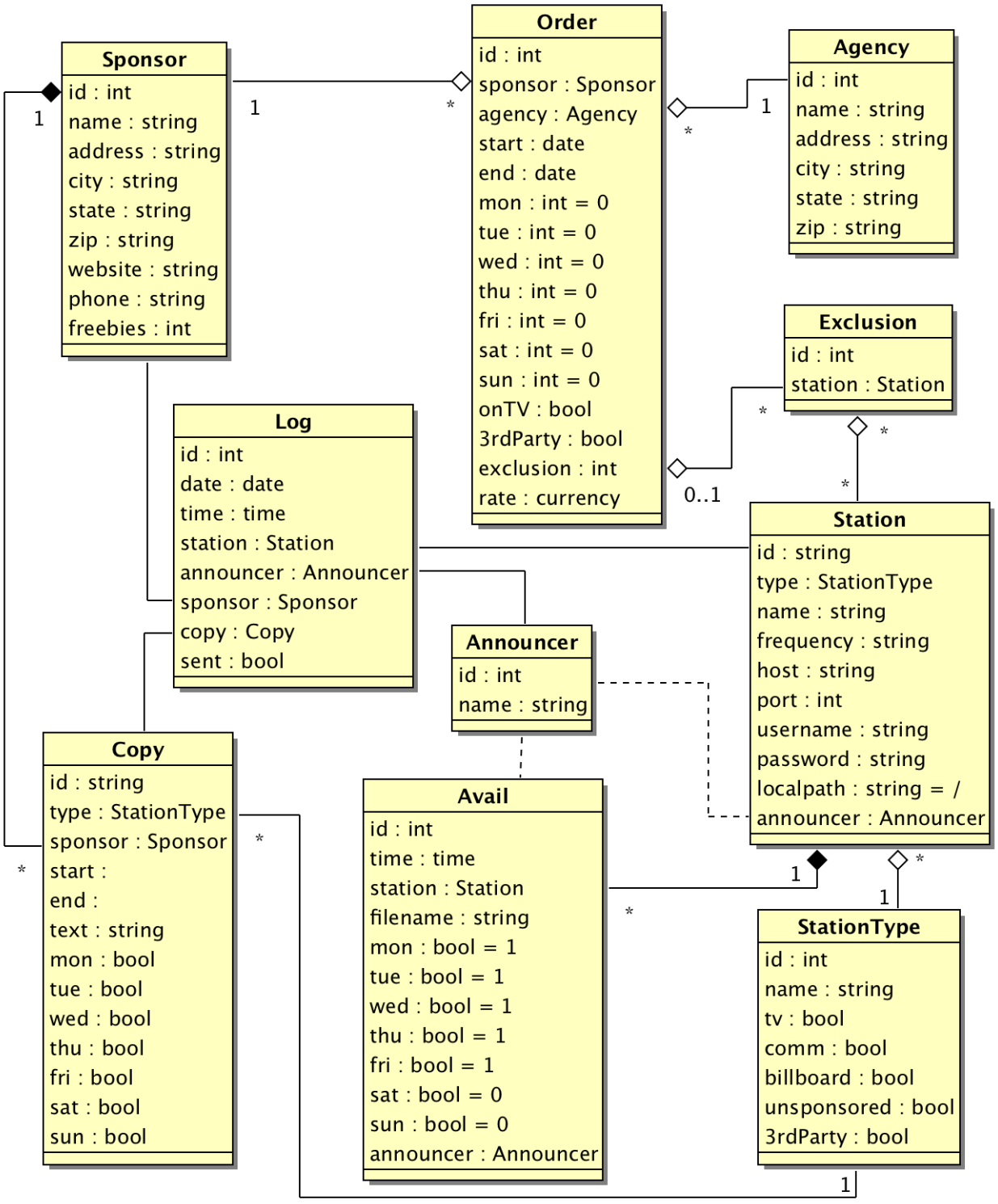

Figure 4: MySQL database structure

To complement the database server, the author developed client software called Traffic Dealer to retain the same functionality and visual interface as the original Traffic Main-screen from the Access database. By separating the reporter's function from the manager's function of updating the database, and extra layer of security was introduced; that is, a reporter can no longer manipulate the database directly without additional authorization into the web interface.

Traffic Dealer was written in C++ using wxWidgets [2] and the middleware, giving it the capability of being recompiled to run on all major operating systems. wxWidgets has also been used in the GNU Radio project [3] (described in detail in a later section) to enable multi-platform support. An important feature of wxWidgets is that it is designed to produce binaries that are system-specific, in that their user-interfaces blend in with the operating system on which they run. This is accomplished not by brute-force emulation (e.g. rendering each widget manually to appear identical to the host OS), but by wrapping and abstracting the APIs of each operating system. Instantiating a wxButton compiles down to a call to CreateWindow within the Microsoft Windows API, or to a call to the corresponding XWindow or GTK+ toolkit method on Linux, or on Mac OS to call the Carbon C++ interfaces.

A major factor in the success of the new client/server system was strict adherence to user interface design guidelines [4] such as:

- Allow the user to review what has been done and whether it was successful
- Keep the sequence of operations and the words used consistent with the user's view of his/her actions and terminology

- Errors can be corrected easily without having to re-enter ‘good’ data as well as ‘bad’ data.

To keep things consistent with the user’s perspective, ease-of-use was designed into the original system from the visual layout - putting the most prominent and pertinent information in the center and in larger font - down to the input device interaction - using the scroll-wheel and page-up / page-down buttons for time-slot selection. Every such element was retained from the old interface in designing the new application. Although the old interface was adequate, room for improvement still remained. While the old interface would bring up a black console window when sending the report showing the progress-bar output of pscp, a utility from the PuTTY [5] suite that securely copies a file to a remote SSH server, the new interface captures the output of a spawned pscp process and displays it in the lower status-bar of the frame. During file transmission, the border color blinks on and off to indicate activity. After file transmission is completed, the border color is set to blue, and the "transmission complete" status is remembered. The ability to select a font for the main text areas was also added since screen resolutions may vary from installation to installation.

There were also functional changes. Since the database resides on a remote host, the application is designed to "phone home" at startup and request today's published log. Although graceful error detection and handling is generally important in software design [6], the new client software was not programmed to deal with the event in which the host cannot be contacted; the application's behavior is simply to fail and abnormally terminate. This is a good thing from the standpoint that it is the only possible failure scenario upon opening, and can therefore be used to prevent piracy or misuse of

the software. In the event that the software does fall into the wrong hands, server passwords can be reset, permanently locking out all distributed versions of the client software. The proper course of action would be to recompile the software with new credentials, and ask announcers to reinstall the software on their workstations from the secure web server.

When the application is closed, the list of sent reports is sent back to the server using pscp, and the server uses this information to mark the "sent" value in the Log table for each successfully sent report.

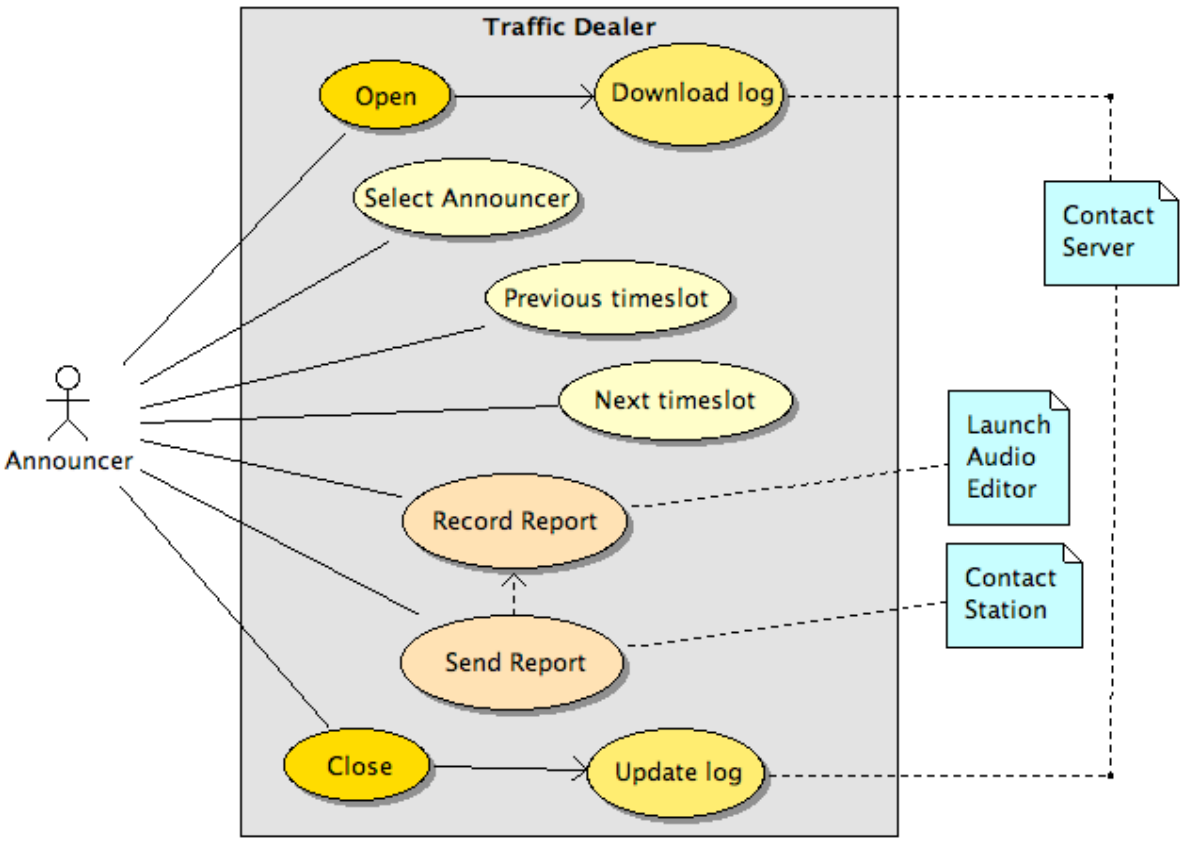

Figure 5: Traffic Dealer use cases

3.3 DOCUMENTATION

Once the operations management software reached a mature phase, it became advantageous to document the undertaken effort. Although no formal UML models were constructed before or during the development process, the author and operations manager did rely on several hand-drawn outlines and diagrams to clarify the software's logic. In order to facilitate future development and communication regarding the software, the author undertook the effort of analyzing the existing code and determining a suitable representation. The process of log generation is the most complex piece of the puzzle, and its UML flow models are shown in Appendix A, along with figures representing the company's conceptual network architecture.

4. Software Radio

4.1 WHAT IS SDR?

Software-Defined Radio (SDR), or simply Software Radio, is a field of study that focuses on a computer's ability process signals being carried via electromagnetic radiation. Rather than relying on traditional radio hardware, which is generally engineered specifically for dealing with only one or two types of signals, SDR attempts to offload demodulation into software to allow more flexibility. Instead of only being able to tune either AM, FM, or GPS data, SDR is theoretically able to receive and interpret any type of signal modulation.

Applications of SDR have foundation in the military and defense sector, with the SpeakEasy project being one of the first such deployments [7]. SpeakEasy established a system for communicating with ten different radio systems from a single device. One fatal error that was made in this project was that it was implemented as a hardware-dependent solution, using machine-specific code. Having learned a valuable lesson in the development process, the project moved more in the direction of more flexible, portable software [8]. Joseph Mitola coined the term “Software Radio” to define the shift in functionality from 80% of the functionality being provided by hardware, to 80% of the functionality being implemented in software, and was one of the pioneers working in the sector to develop this research [9].

The software radio project known as GNU Radio has been maintained mostly by a small group of people, of which the most prominent figure is Eric Blossom. According to Blossom, the goal of GNU Radio is to "get the software as close to the antenna as is feasible" [10]. GNU Radio provides a framework in which to develop signal-processing

code in a reusable form by providing a separation between low-level algorithms and higher-level functionality.

USRP stands for Universal Software Radio Peripheral, and is a device that works via USB to enable computer access to the electromagnetic spectrum. It is an open source hardware platform, which means that the designs and schematics are publicly available. However, since it is necessary for someone to actually build and market this component, Matt Ettus of Ettus Research [11] is responsible for sales of the USRP.

The USRP consists of several components. The fundamental unit is the motherboard. It handles USB connection, power distribution, and controlling signal flow. The motherboard has four slots for daughterboards - two for transmission, and two for reception. Daughterboards can be purchased, each of which specialize in receiving or transmitting a specific frequency range.

In order for the computer to acquire and deal with a signal from a daughterboard, which communicates in the analog domain, the signal must be digitized by an analog to digital converter (ADC). Part of this process involves decimating the signal (approximating its analog representation using a finite series of digits). For the transmission element of the USRP, the inverse is used - a digital to analog converter (DAC). The USRP consists of two ADCs and two DACs connected to the FPGA.

On its motherboard as the central processing unit, there is a field programmable gate array (FPGA), which is a small processor that can be reprogrammed quickly during initialization. It is not as powerful as a modern x86 processor, but it is suitable for controlling the flow of data to and from the ADC/DACs.

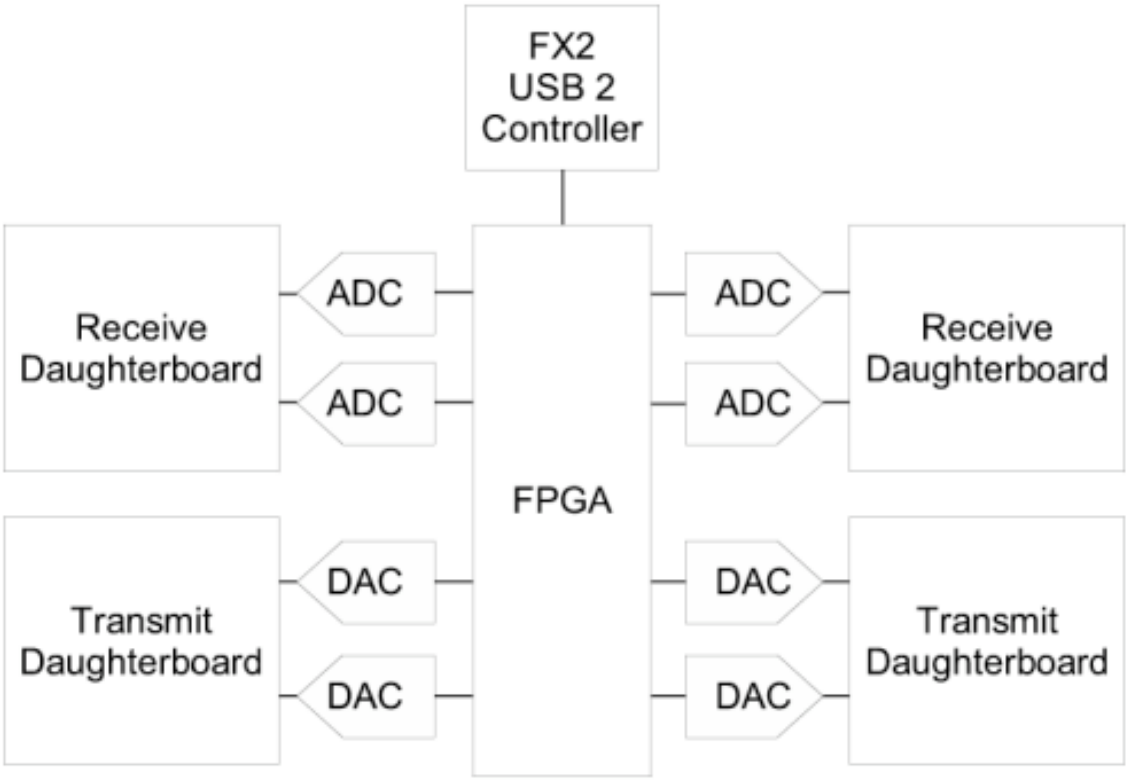

Figure 6: USRP schematic

4.2 WHY SDR?

To provide context to the situation of broadcast radio, it is necessary to consider the time in which these technologies became available. Because there is a nationwide shift from traditional analog bandwidth allocation to digital trunking frequency allocation, it is very attractive to have the capacity to remotely reprogram a receiving device, and to have a standardized hardware device that doesn't need to be replaced for each markets' radio transmission system.

In addition to the capability to change modalities dynamically, SDR offers a more robust platform and tighter integration with existing software components than would be possible with traditional receivers, which have a tendency to encapsulate all

control/status communications with the device using RS-232, and requiring separate cabling to connect the radio to the audio-card to receive the signal. When a device's firmware is not accessible, it is not possible to exceed any limitations imposed by its manufacturer. With the USRP specifically, its firmware is readily accessible and easily loaded into the system at runtime. Modifying the firmware for the USRP however, has not been necessary, as its standard firmware is more than adequate for most practical purposes. Indeed because of its simplicity, most of the signal processing and control programming is high-level and intended to run on the host machine's CPU.

4.3 COMPETITORS IN THE TRAFFIC REPORTING MARKET

Metro Networks is one of Traffic on Demand's primary competitors, and they have existed in this market since 1974. In 1999 Westwood One, a major broadcaster in the radio market, acquired them. Their methods are similar, in that they rely on information gained through contact with 911 operators and county dispatch. In 2000 Westwood One acquired SmartRoute Systems, which became their main hub for traffic information provided by local and state governments. As an established market force, they are also able to afford helicopters for an aerial view of traffic issues in larger markets.

Ascertaining the competition in smaller markets such as Wilmington, NC is more difficult. Because media giants such as Metro do not have a presence, local information companies can gain foothold, creating a more fragmented national market. In smaller markets too, traffic reporting is often performed by the broadcasting affiliates rather than by third parties.

4.4 *Alternative Technologies and Approaches*

In the past, the approach of deploying a remote police scanner involved purchasing police scanners at local retailers, and connecting them to the deployable Linux server that would broadcast an MP3 stream via Internet, which could then be listened to by reporters locally. This was viewed as inefficient because the nature of most consumer-level radio scanners is that they are either not remotely programmable, or they require proprietary software to perform the programming via RS-232. It was also considered inadequate because these scanners are not able to report back to the listener what frequency they are currently hearing. Another inefficiency arose noting that the MP3 streaming requires about 20 seconds of latency from start of broadcast to the time the listener is able to hear the feed.

FlexRadio is another SDR technology that was investigated. Although their technologies offer the robustness of a software-defined radio solution at a reasonable price ($1599 for an entry-level unit), they rely upon Microsoft operating systems in order to run. The goal of this research is to avoid commercial software when possible to eliminate the licensing fees and to avoid the imposition of planned obsolescence [12].

4.5 *Strengths and Weaknesses*

The main strength that is offered by SDR functionality in this application is that it is possible to remotely program not only the frequencies, but also the mode of reception of the radio in the field. With the USRP, although its cost is high, it more than pays for itself considering the multiple trips that could be required in the situation of a faulty or improperly configured traditional radio, or the time required of the remote radio station's

engineer to assist in troubleshooting the or reprogramming the receiver. By developing a more manageable and flexible solution, expansion becomes less costly overall.

A remaining weakness of this approach is that although the radio can be configured with multiple daughterboards, it is possible that a daughterboard deployed in the remote location may not be able to receive all frequencies that are broadcast in that area. It still will be necessary to perform research before attempting to deploy the USRP with the appropriate daughterboards. The possibility that the equipment could need replacing due to an electrical spike at the receiver's antenna input is also a concern that must be considered.

4.6 Experience with GNU Radio and the USRP

The GNU Radio project wiki has typically been the de-facto source for documentation. During the first attempt to install GNU Radio on Windows XP, there arose very tedious problems because the wiki's instructions no longer applied to the current state of the project at the time. In order to run on Windows, a Linux abstraction and development layer such as Cygwin or MinGW must be installed. Both of these development kits are geared towards compiling POSIX-based software code to run on Windows. Using Cygwin seemed to work relatively well, as indicated in the wiki, but when tuning into a signal, it would often lose synchronization with the host device after a few seconds. This was partly due to the fact that only an older version of GNU Radio could be compiled under Cygwin. Getting MinGW to work seemed less tedious, and actually would tune in a signal without losing connection to the USRP, but there were still obstacles similar to those experienced with Cygwin, such as some libraries not being found without extra post-installation. The SDL library enables connection to the

accelerated graphics adapter for video streaming, but it too was not readily accessible when compiling on the Windows platform. Given that the project focuses on a deployable Linux server, these platform-specific issues can be ignored effectively.

Installing GNU Radio on Linux was relatively painless. The Linux distribution chosen was Arch Linux, a Slackware [13] derivative with a decent community, standardized text-based configuration mechanisms, and very up-to-date software packages. Although there did not yet exist specific instructions regarding installation on Arch Linux, it was not difficult to make use of existing instructions for installation on other Linux distributions.

Within GNU Radio, there is a separation of levels of detail. Low-level code requiring the most efficiency such as the signal-processing blocks that perform demodulation and frequency-domain transformations are written in C++. For code that does not require low-latency, such as the GUI framework and the code that connects the blocks together into a directed graph, is coded in Python. There is an intermediate library called SWIG (Simple Wrapper and Interface Generator) that is used to resolve correct bindings between the different languages. This abstraction makes it possible for the developer to focus on the task at hand, rather than needing to consider all possible connections between the hardware, the signal processing units, and the software interface. The part that the developer will focus on immediately will be the Python interface that allows scanning and receiving, but it may become necessary to code a block or two in C++ for dealing with trunked radio systems. Figure 7 shows the general architecture of GNU Radio [14].

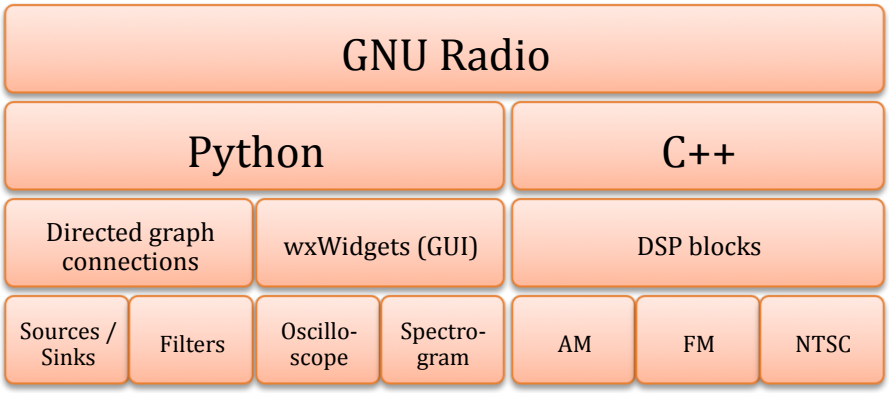

Figure 7: Conceptual diagram of the GNU Radio architecture

4.7 PACKAGING AND DELIVERABLES

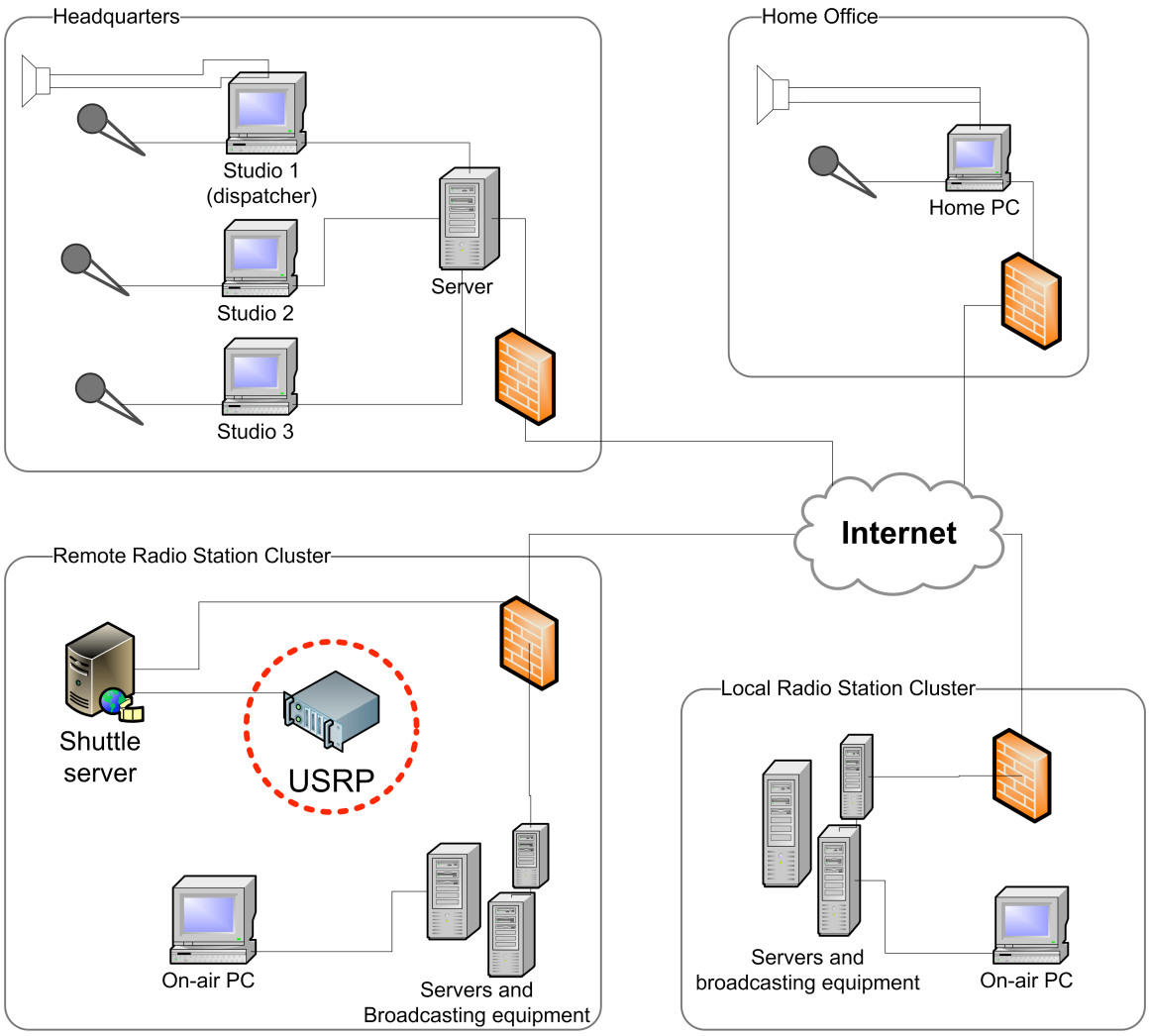

Figure 8: Network including the deployment of the USRP

The overall goal of this project is to produce a radio scanner system that can be deployed in a remote location with ease. In order to package this system, it would be desirable to incorporate all hardware into one “black box” server, ideally rack-mountable. In situations where cost-performance is a greater constraint, it would be possible to use a small form-factor PC such as Shuttle as has been used for reporter workstations, and bundle the USRP either within or alongside the unit.

It is preferable to have control over subsystems in text-mode so that they do not require a stable graphical layer, which would increase load times. Because graphics adapters can generate heat, their long-term stability should be considered marginal and unnecessary for essential services such as networking, firewall, and streaming audio.

Because latency may be a factor in broadcasting an audio stream via the Internet, a buffer may be needed to store which frequencies have been locked onto at a given time. The user would see this buffer on their client software to see synchronous information about what is being tuned.

An important concern is ensuring that the squelch is not set too high or low. If the squelch is set too low, it will send excess noise to the dispatcher. The squelch parameter keeps the audio feed silent when there is only noise and no incoming signal. Many existing scanners do not address the issue of being able to adjust the squelch in software; they simply have a physical knob to adjust the squelch parameter. With the new system, all adjustable parameters will be written in software.

Another important issue to address is usability. The simplicity with which the interface can be driven is a key factor; the dispatcher must be able to add or remove frequencies from the scanning range. An additional feature which is less important but

still attainable, is a simple graphic equalizer consisting of bass / mid / treble knobs. These signal processing blocks will be applied before Internet retransmission takes place, and will serve as a filter to increase the quality of the rebroadcasted signal.

4.7.1 Key deliverables

- The USRP must automatically scan frequencies using the TVRX board, ranging between 50 and 860 MHz.
- There must be an interface for controlling the scanner:
  - Adjustable squelch.
  - List of frequencies to be scanned.
- There must be an interface showing which frequency is currently tuned.
- The audio must be rebroadcasted via the Internet.

4.7.2 Key benefits

- Ability to remotely monitor police and emergency communications
- Ability to remotely adjust the scanner
  - Adjustable squelch
  - Frequency list reprogrammable
  - Ability to adapt to different modes of transmission
  - Full remote manageability

4.7.3 Additional objectives

- Investigate simultaneous utilization of both daughterboards
- Attempt streaming via Asterisk VoIP system to reduce latency
- Introduce bass/mid/treble filtering

4.7.3.1 Graphical User Interface

Whether the interface for controlling the scanner will be a graphical or a text-mode user interface is an important consideration. Because of the latency, bandwidth requirement, and overhead that a graphical interface would demand, it may be more realistic to implement the interface in text-mode. However, the possibility of creating a graphical interface has not been completely ruled out, since there already exist several examples of working interfaces. To show which frequency is currently being tuned, the console mode will work quite well due to the minimal latency and bandwidth

requirements of the text-driven console. Making the interface attractive and intuitive will be high priority. Appropriate use of text coloring will be attempted in order to alert the user to different events within the system. If a graphical user interface is to be constructed, it may look like Figure 9.

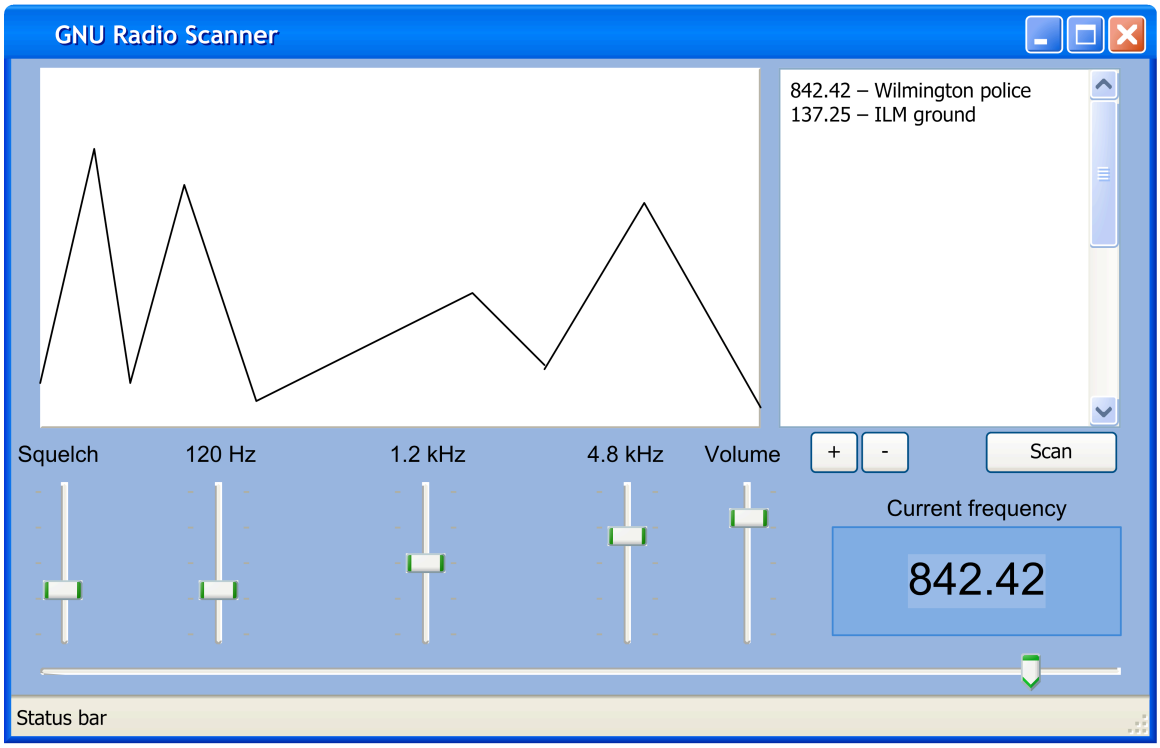

Figure 9: Possible GUI layout

4.7.4 Concerns

- Working with digital trunking systems may require additional research
- Adequate bandwidth at deployment location will be necessary

4.8 PROJECT STATUS

At this point, the author has identified and taken the necessary steps to create an interface with adjustable squelch, frequency list and current frequency indicator, and audio route to the sound card. Nullsoft's Shoutcast server [15,16] monitors the sound card's capture device, and the audio is streamed in MP3 format via HTTP. In its

simplest form, it will work best with high-speed connections on both ends, as VNC can be used for controlling the GNU Radio tuning system. Abandoning this approach due to the high latency of the system, the author devoted time towards the VoIP solution.

With the Asterisk [17] VoIP system version 1.6 installed, the author has developed a GNU Radio Python program that streams signed-integer data to a named FIFO pipe that is consumed by the Asterisk server as a “music on hold” (MoH) process. This interaction is extremely desirable, as it allows direct communication between the RF receiver and the streaming server, requiring no intermediate re-sampling between the output of GNU Radio and the input for MoH, keeping latency to a bare minimum. It uses the User Datagram Protocol (UDP), which is more suited towards real-time streaming than Transmission Control Protocol (TCP), having the overhead of flow control and error correction. In addition, there is separation between the streaming program and the invocation that Asterisk uses to access that stream. This situation is very effective at enforcing runtime stability. Because either service can run independent of the other, it does not become necessary to restart both servers simultaneously in the event that the developer needs to make modifications to the streaming and control program. For the listener, this means that there will be brief moments of silence as the stream restarts, rather than presenting the listener with an error message such as “Disconnected” or “Service unavailable”. In most situations, the listener will not notice.

The control interface works over an SSH pipe as a simple shell script that binds to named pipes within the system, and the user can connect to it using PuTTY [5] in Windows, or OpenSSH [18] in Mac OS or Linux. Making use of ANSI control characters in presenting the user with information about the state of the radio scanner allows for a

colorful, dynamically updating display. It accepts commands in plain text such as `list`, `vol` for volume, `sq` for squelch, and of course `help` for a list of all supported commands. There are three modes in the program for display: one that leaves the cursor in place and prompts the user for action while continuing to scan, one that scrolls down for each successfully tuned frequency, and one that pauses scanning and stays tuned to the current frequency. Switching between the modes is accomplished by pressing the Enter key when no command has been entered, allowing the user to quickly pause or resume. Commands can also be abbreviated by using the first letter of the command, allowing experienced users to navigate the interface more quickly.

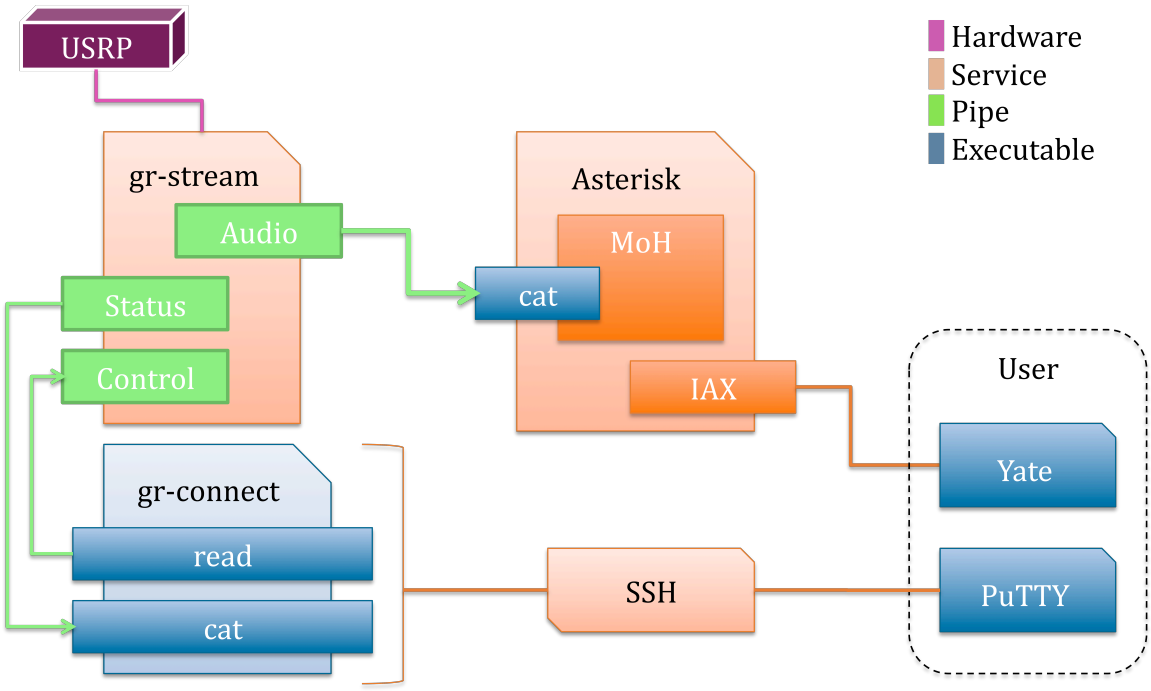

Figure 10: Conceptual outline of the USRP to Asterisk and user's perspective

An attempted idea involved designing a control interface using the messaging features of Session Initiation Protocol (SIP) [19] to communicate via instant messages between the user and the scanning software. While this sounds good in theory for simplicity, the reality is that it does not provide as dynamic an interface as an SSH

connection can, and there doesn't seem to be much support in the current version of Asterisk for SIP messaging. Compounding this problem, SIP is not intended to traverse NAT boundaries, so its use as a protocol for connecting end-users is only appropriate when the Asterisk server is located on the same local subnet as the user. Instead, the Inter-Asterisk Exchange (IAX) Protocol [20,21] is used to connect listeners to the stream via soft-phones such as Kiax or Yate [22,23]. Although IAX does not support instant messaging, using the numeric keypad to issue commands may be very possible by reconfiguring the extensions.conf file in `/etc/asterisk`, and the `/usr/local/bin/gr-agi` PHP script which can communicate between Asterisk and GNU Radio via the Asterisk Gateway Interface (AGI).

5. Methodology and Timeline

The author's involvement with the company's various projects has varied through time. This section will quantitatively describe the level of involvement the author undertook from the company's inception to the present day. The following dimensions of involvement will be shown: Website development, training the users to perform operations with the new systems, analysis performed by the author, server installation or configuration, maintenance of broadcasters' file-receiving capabilities, expansion efforts into remote markets, and research involving the software radio.

The Unified Process Life Cycle can be used to describe the methodology and approach used by the author in developing the systems. At any given point, the author assumed whichever role was most appropriate at the time, with the primary objective being optimization of the company's operations. During inception phases, ideas were informally communicated between company managers and the developer. During elaboration, the developer took careful consideration to the implications of the project to be undertaken, and did preliminary testing to assure the project could be completed successfully. After elaboration, the construction phase, during which the developer put forth the greatest amount of effort. At the transition phase, the developer was able to shift focus away from the project.

The following diagram describes which projects occupied the developer at points in time over the last four years. Higher points in the vertical dimension indicate greater involvement.

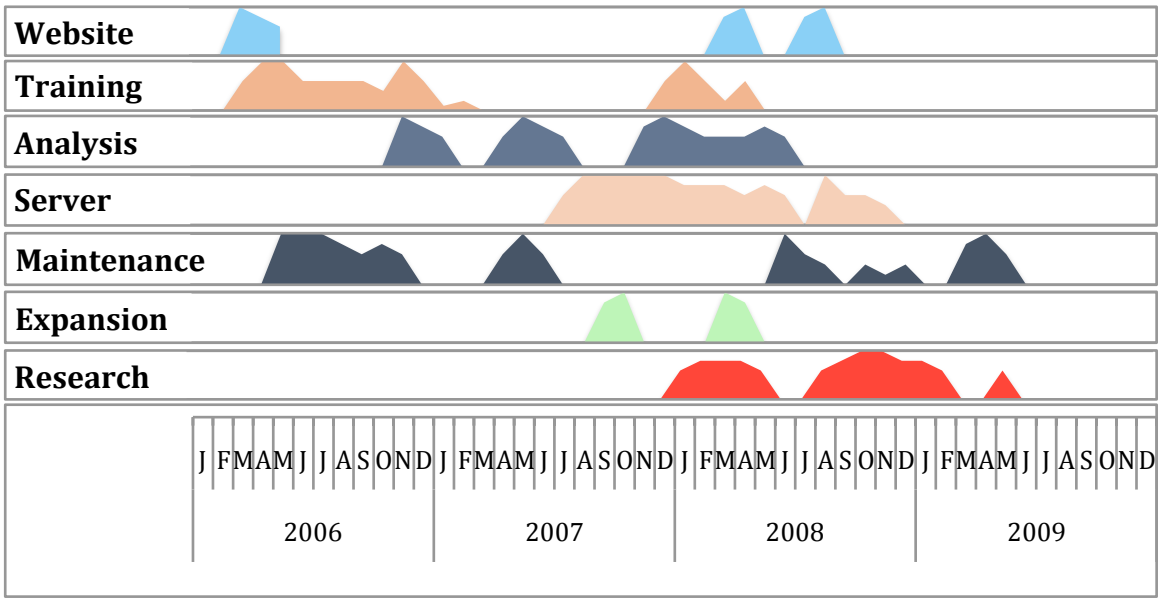

Figure 11: Involvement graph

5.1 *INITIAL TIMELINE*

2008

January	8	Start directed independent study of GNU Radio and USRP
February	1	Start research with USRP and GNU Radio
March	28	Trip to Grand Rapids, Michigan for server installation
April	29	GNU Radio / USRP demo presentation
August	22	Begin writing proposal
September	25	GNU Radio / USRP colloquium
October	7	Begin writing IEEE paper
October	16	Proposal defense
October	31	Deadline for publishing to IEEE SoutheastCon

(Proposed)

November	1	Start implementation of GNU Radio scanner

	December	12	Completion of a working GNU Radio scanner prototype
2009			
	January	15	Capstone defense

5.2 ACTUAL TIMELINE

2009

April	18	Start implementation of GNU Radio scanner with GUI
May	1	Successful GUI implementation with squelch and scanning
September	29	Resume work on GNU Radio scanner, GUI abandoned
October	26	GNU Radio to Asterisk / SSH control prototype completed
November	10	Capstone defense

6. Conclusion and Future Work

It has been demonstrated that the USRP hardware and GNU Radio software can be used in tandem with real-time streaming services such as Asterisk to produce a remotely reprogrammable scanning radio. Such a device will improve traffic reporting companies' ability to expand to remote locations by facilitating the transmission of more up-to-date information.

Future directions for this project include experiments with multiple USRPs tied to one server, adding multi-modal scanning functionality, extending the list capabilities to include per-frequency squelch and volume gain parameters, and scaling the host system down in size and processor speed to reduce power consumption. Another direction this project could take is to enable multi-frequency reception using fast scanning routines, multiple buffers, and interpolative signal processing to compensate. Such a system would have the ability to effectively tune in more than one frequency using one daughterboard, at the expense of audio quality. The time that it takes to switch from one frequency to the next will be an important factor to consider in such an implementation.

With this prototype, there exists a caveat such that only one simultaneous connection is available to the control interface. Although the control interface can easily support more than one incoming control connection, the readout functionality is limited to one viewer due to the fact that the output stream from the streaming service is tied directly to a FIFO named pipe. In order to allow simultaneous viewers, a multi-threaded server will be necessary to attach itself to the output stream's named pipe, and redirect this data to multiple connected streams or sockets.

Appendix A - Additional UML

UML diagrams describing the log generation functions
and the company’s conceptual architecture

7. Appendix A - Additional UML

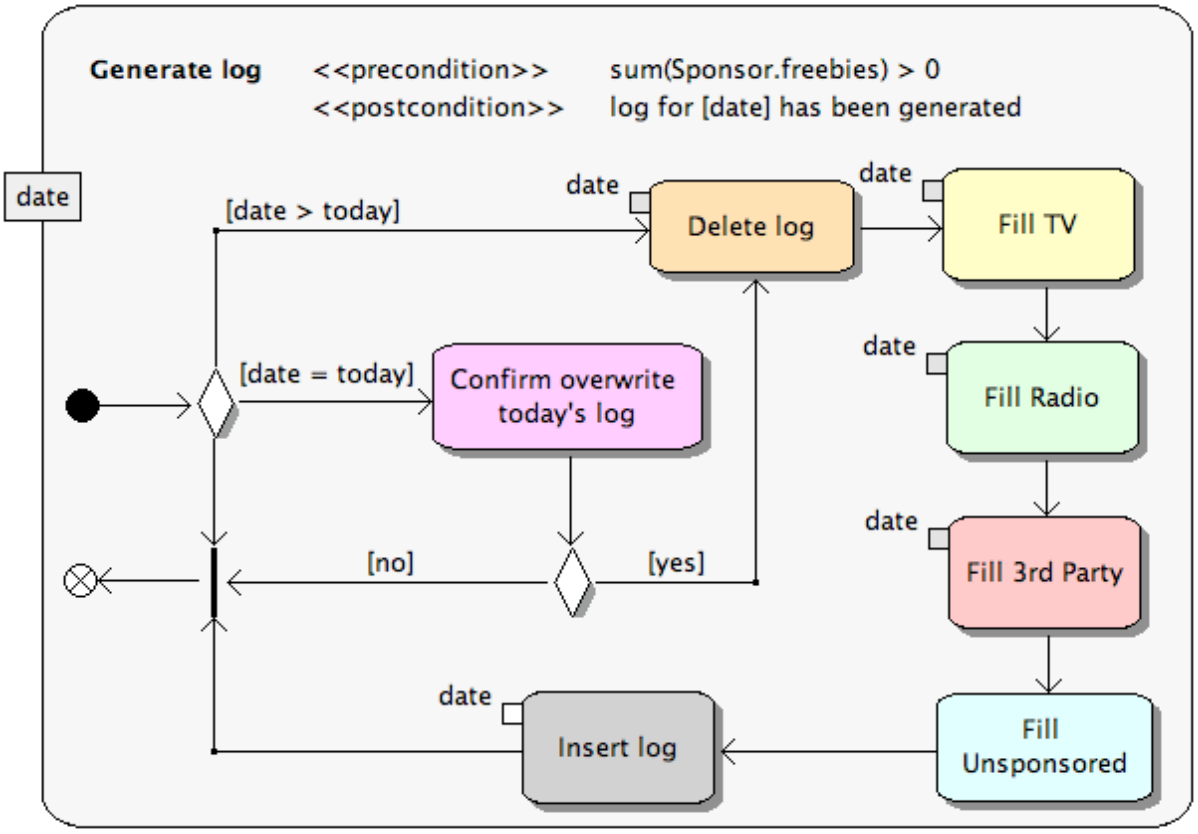

Figure 12: General overview of the log generation process

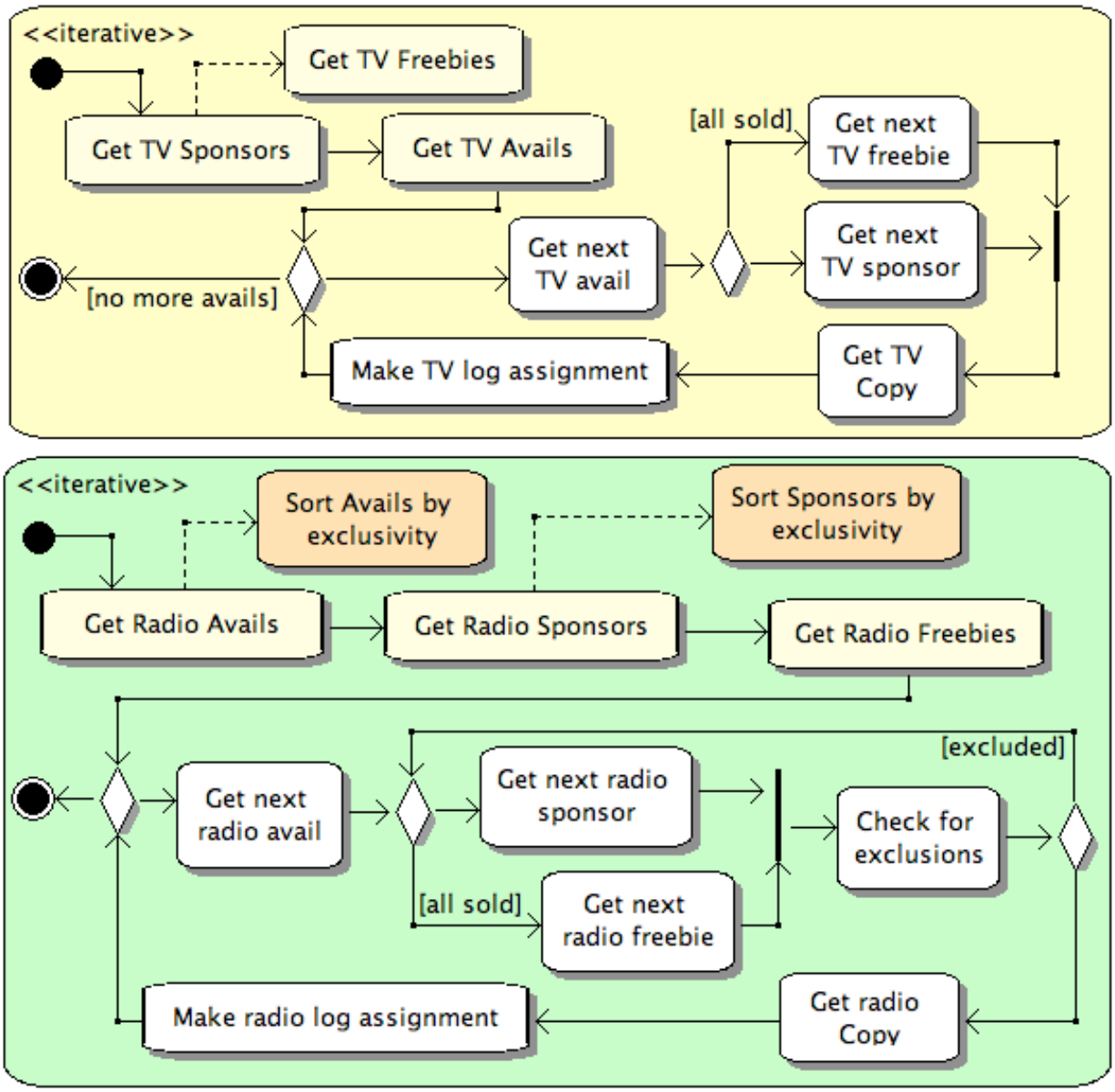

Figure 13: Flow diagram for TV and commercial radio log generation process

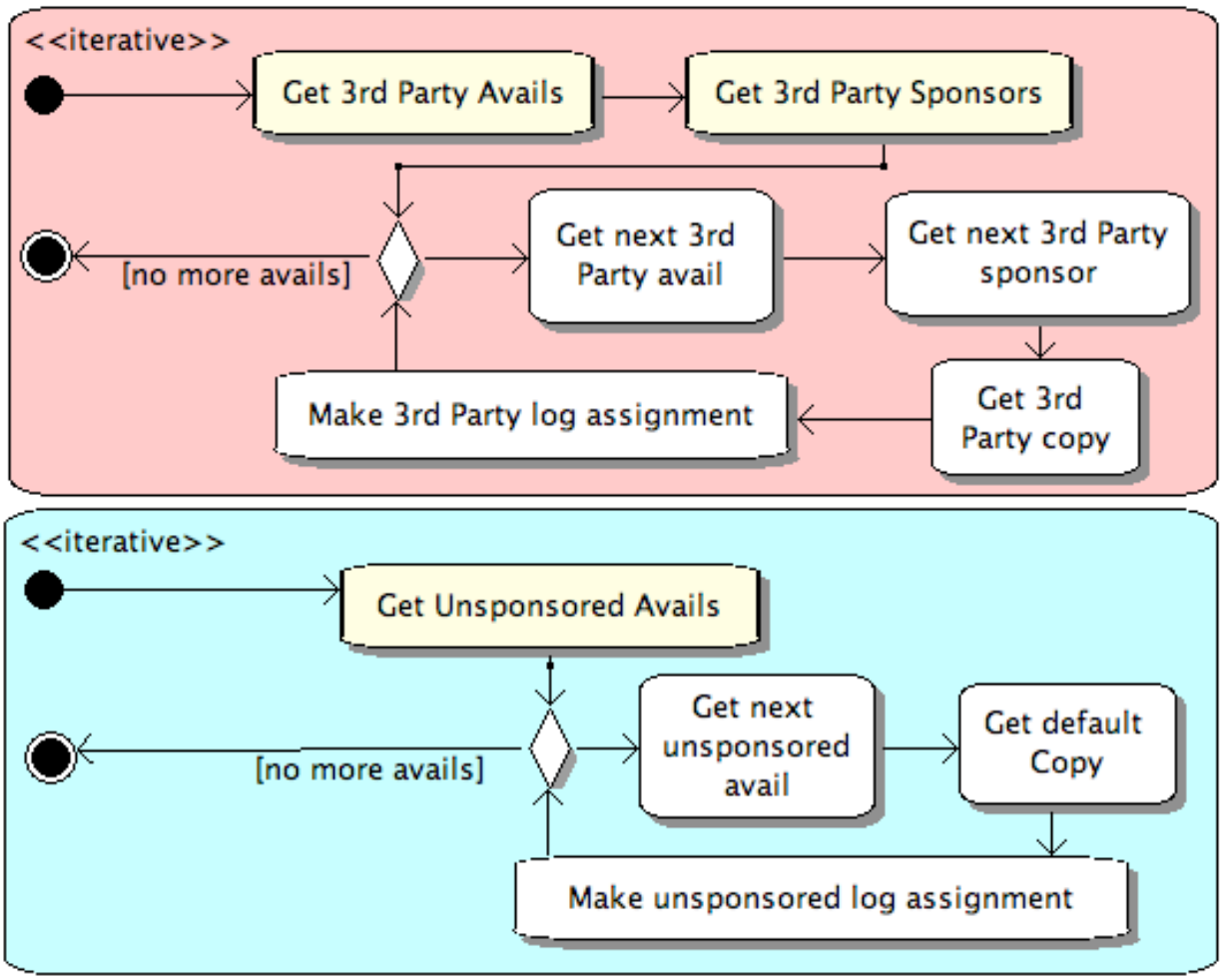

Figure 14: Flow diagram for non-commercial radio and unsponsored log generation process

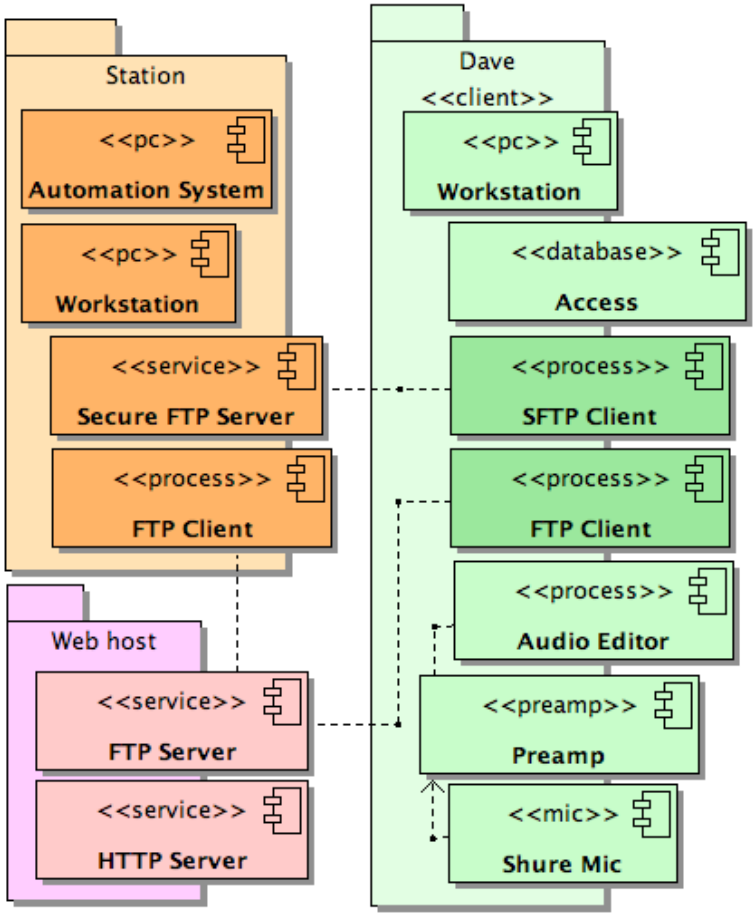

Figure 15: Original architecture using Access database

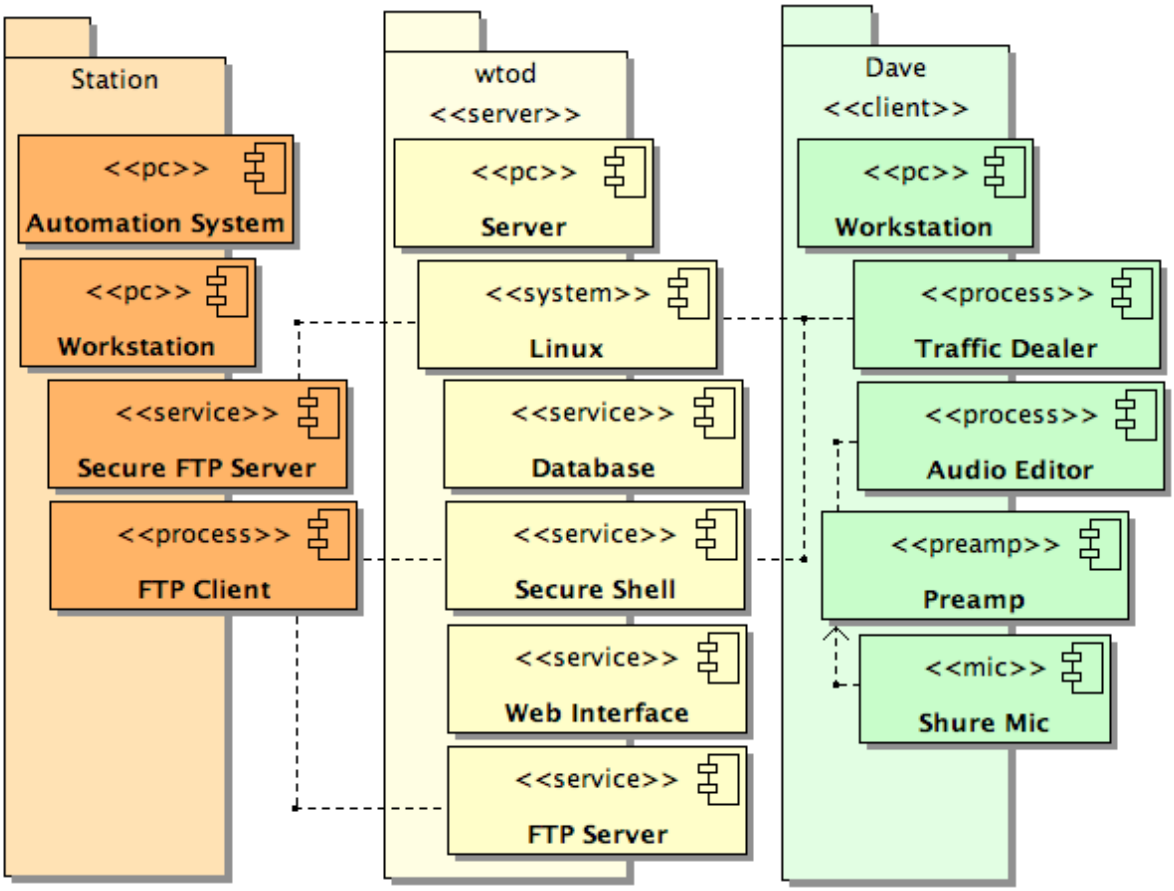

Figure 16: Refined client/server architecture

Appendix B - Simulation

8. Appendix B - Simulation

The author undertook a quantitative analysis of time-cost reduction provided by the new system of operations. By automating the process of originating and transmitting the report, the announcer is left with the responsibility to follow the interface and scroll to the next scheduled report, perform the [ideally] brief task of recording and producing the report, and pressing a button the transmit the report to the broadcaster. This compares with the previous method of requiring the reporter to follow a paper log of station report times, locate and identify the correct copy document in Word, navigate using Windows Explorer to the location of the recorded file, and connect to the proper server using an FTP client. Not only does this save time, it also simplifies the selection and training process in hiring new reporters by requiring less technical expertise.

8.1 HYPOTHESIS AND SCENARIOS

The main hypothesis of this simulation projects was as follows: Computerization and automation have greatly improved routine operations undertaken by the company. The main question is “by how much have things improved?” In this simulation, four scenarios were considered:

1. Manual operations performed by two reporters.
2. Manual operation performed by three reporters.
3. Automated operation performed by three reporters.
4. Automated operation performed by two reporters.

All simulations were performed on the same volume of demand, that is, the number of reports to be delivered remained constant throughout. The process of log generation was estimated based on the experience of the operations manager, who

manually performed this process using pen and paper before the database system was in place. Indeed, as much as possible, realistic figures were used to estimate the time required to perform each task.

8.2 EXPERIMENT DESCRIPTION

In the context of simulation, the reporters were considered as resources to be occupied by the duty of performing voice recordings. The demand for a given report was considered as an entity within the model as the task to be performed, and were generated by a scheduling mechanism that closely and accurately resembles a working shift at a point in time with a total demand of 160 reports per day to be distributed to reporters using first-available queuing. Another entity was considered: an incoming radio signal to represent the task of acquiring information via police scanner or telephone calls, these tasks occupying between 5 and 20 seconds, and being generated pseudo-randomly. The final entity considered is generated once daily at the end of the working day and represents the need to perform the normally time-consuming task of generating a log for the next day.

The logic of the model is as follows: Attending to demand for a traffic report takes first priority. When demand arrives, it is sent to an available reporter. If no reporters are available, it is stored in a queue and its time is recorded. The reporter must scan the log for the next report, locate the copy script to read from, record the report, find the correct file to send, and finally wait for the report to completely transmit to the broadcaster. Some steps' times are cut down to zero by computerization / automation, represented in the third and fourth scenarios.

When a radio signal arrives, it is ignored unless one of the reporters is available to listen in. This information can be relayed verbally or via instant-messenger to other reporters if necessary, but this is not noted in the model as it can be allocated to free time.

Several outputs of the simulation were considered as final metrics: The time by which a report is delayed due to the reporter being occupied with another previous report - higher is worse because it represents a deadline missed, and ideally no queuing should occur; the time required to generate a log is considered and is crucial in demonstrating the overall time savings of the automated system of manual processes; and the time from start-to-finish of originating a report is considered and is proportional to the company's ability to complete more tasks with the same number of reporters. Since there were multiple scenarios with variable time and a varying number of announcers, controls (additional independent variables) were not needed. Thirty replications of each scenario were simulated - a good number because it represents roughly a month's worth of work.

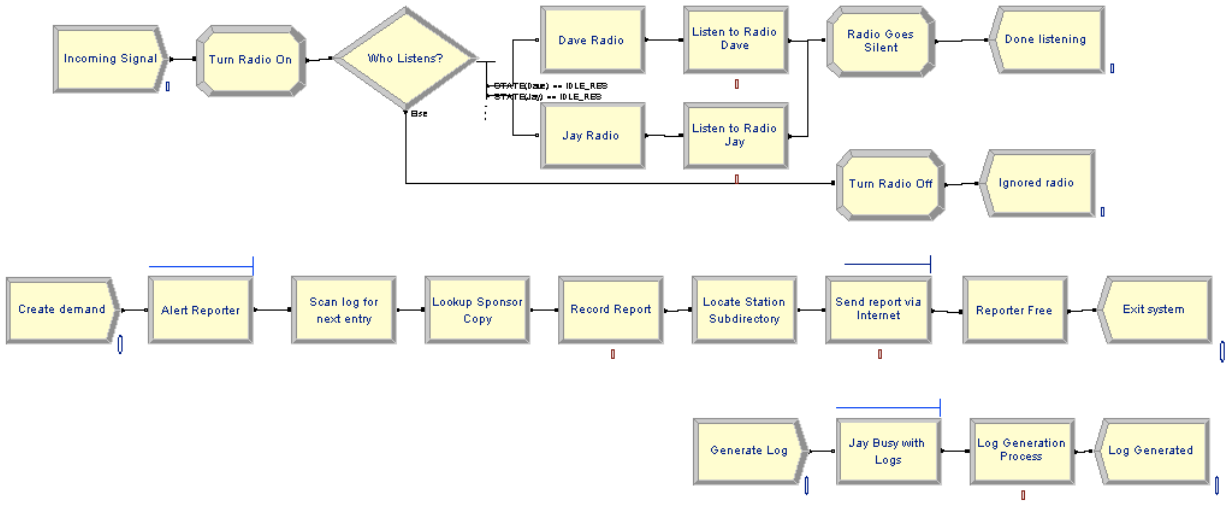

Figure 17: Simulation model

8.3 RESULTS

The results for the four scenarios follow in Table 1, all times in minutes:

Table 1: Results of the simulation

Scenario	Delay time	Log generation	Report time
1	0.265	151.692	2.041
2	0.029	146.061	1.814
3	0.009	0.027	1.269
4	0.073	0.027	1.323

By adding more reporters (the difference between scenarios 1 and 2), report times are only slightly reduced, but the duration of any delay in sending reports drops off to a matter of seconds. The log generation process doesn't change very much by adding another reporter in the simulation, but in all reality there would be extra time introduced into the log generation process, as the person generating the log must consider how to split up the reports (either by station or by time of day). The time saved in log generation alone is a huge improvement, as it frees the operations manager by two-and-a-half hours per day. For the operations manager this means the difference between going home at 6:30pm and staying as late as 10:00pm.

These results also demonstrate the benefit of implementing a computerized system of automation versus hiring additional announcers. The delay time drops even further with the automated system, and the report times themselves take less time, saving a cumulative 11 minutes per day per announcer. By allowing more time for the announcers to breathe, the quality of reports is improved by the reduction of stress levels, providing a higher-quality work environment overall. For the organization it means that they can expand their affiliate network more efficiently, requiring fewer announcers.

9. Appendix C - Budget

The following describes the costs associated with deploying a single USRP with one server into a market. In deployment, there may be several servers and/or USRPs required depending on the available signal strength at the locations. Since there may be a few variables to consider in regard to the power and speed of the system, a minimal baseline configuration is considered alongside a more expensive and robust configuration. Also considered is the possibility that either automotive or airline travel expenses would be required for the engineer to perform on-site installation.

As these budgets represent current prices, they are subject to change.

Capital investment	Baseline	Premium	w/Airline
Server hardware	300	1100	
USRP	700	1000	
Daughterboard(s)	100	800	
Hardware total	1100	2900	
Configuration	1000	1500	
Travel allowance	0	100	1000
Total capital for USRP	2100	4400	5400
Recurring expenses	**Monthly**	**Annually**	
Internet bandwidth	80	960	
Server maintenance	300	3600	
Total recurring for USRP	380	4560	

Also worth considering is the fact that deployment of the company's infrastructure into a new market may require the hiring of additional reporters. In general, we will assume that two additional reporter will be involved, and that at least one of them will require a workstation to produce and transmit reports. We will assume that the reporter will pay for and utilize his/her own Internet connection. Although travel is much less likely when configuring workstations, we will nevertheless consider the expenses associated with travel, in the event that it becomes necessary.

Capital Investment	Baseline	Premium	w/Airline
Workstation base	200	400	
CPU	150	300	
Memory	30	100	
Monitor and peripherals	160	400	
Operating system	0	150	
Workstation total	440	1350	
Configuration	0	200	
Travel allowance	0	100	1000
Total capital for workstation	440	1650	2650

10. References

1 Arch Linux - http://www.archlinux.org/

2 wxWidgets - http://www.wxwidgets.org/

3 GNU Radio - http://gnuradio.org/trac

4 Wade, J. 1984. Practical guidelines for a user-friendly interface. In *Proceedings of the international Conference on APL* (Finland, June 11 - 15, 1984). APL '84. ACM, New York, NY, 365-371. DOI= http://doi.acm.org/10.1145/800058.801122

5 PuTTY: a free telnet/ssh client - http://www.chiark.greenend.org.uk/~sgtatham/putty/

6 Weimer, W. and Necula, G. C. 2008. Exceptional situations and program reliability. ACM Trans. Program. Lang. Syst. 30, 2 (Mar. 2008), 1-51. DOI= http://0-doi.acm.org.uncclc.coast.uncwil.edu/10.1145/1330017.1330019

7 RJ Lackey and DW Upmal, "Speakeasy: The Military Software Radio," IEEE Communications Magazine, May 1995.

8 Bose, Vanu, “A Software Driven Approach to SDR”, http://www.cotsjournalonline.com/articles/view/100056

9 Joseph Mitola III - http://web.it.kth.se/~maguire/jmitola/

10 E. Blossom, “GNU Radio: Tools for Exploring the Radio Frequency Spectrum,” Linux Journal, 2004 - http://www.linuxjournal.com/article/7319

11 Ettus Research LLC - http://www.ettus.com/

12 Woolley, M. 2003. Choreographing obsolescence - ecodesign: the pleasure/dissatisfaction cycle. In Proceedings of the 2003 international Conference on Designing Pleasurable Products and interfaces (Pittsburgh, PA, USA, June 23 - 26, 2003). DPPI '03. ACM, New York, NY, 77-81. DOI= http://0-doi.acm.org.uncclc.coast.uncwil.edu/10.1145/782896.782916

13 Slackware - http://www.slackware.com/

14 Tucker, D. C., “Prototyping with GNU Radio and the USRP”, paper #72 in the proceedings of IEEE SouthEastCon 2009, IEEE Press

15 SHOUTCast Radio - http://www.shoutcast.com/

16 nullsoft.com - http://www.nullsoft.com/

17 Asterisk - http://www.asterisk.org/

18 OpenSSH - http://www.openssh.com/

19 Session Initiation Protocol - http://www.voip-info.org/wiki/view/SIP

20 IAX - http://www.voip-info.org/wiki/view/IAX

21 IAX versus SIP - http://www.voip-info.org/wiki/view/IAX+versus+SIP

22 Forschung Direkt: Kiax - http://www.forschung-direkt.eu/projects/kiax2/

23 Yate - http://yate.null.ro/pmwiki/