Software to Analyse Ultrasonic Scanning Results

by

Thea Gegenberg

Master of Pure Mathematics, Waterloo, 2011
Bachelor of Science, SFU, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s):	Rodney Cooper, Master, Computer Science
Examining Board:	Przemyslaw Pochec, PhD, Computer Science, Chair
	Michael Fleming, PhD, Computer Science

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

September 2015

Abstract

The nuclear power industry is moving towards automated monitoring systems, which function during normal operation. UNB CNER's UTPro scanner is designed to search for flaws in active pipe infrastructure of nuclear power plants through ultrasonic waves. The author was tasked with designing software to read the output of the scanner, organize the inputted scans, analyze them, and produce reports based on automated data analysis. While the largest part of the project was straightforward, challenges were found in the development of the algorithms to automatically detect flaws and compare scans, as well as the creative task of designing an interface and reporting structure which was intuitive for the engineers. While the completed software was able to successfully find manually created flaws in pipes for testing purposes, the author was unable to determine how similar its performance was to a trained human technician manually searching for flaws.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Heavy industries, such as the nuclear industry, are increasingly moving to monitoring systems, such as pipe scanning systems, that function during normal operation.

The Canadian Nuclear Industry is trying to solve the problem of monitoring systems that function in the extreme environment of a nuclear power plant. Currently, in extreme environments, pipe scanning can only be done when the system is not functioning and this is not practical for the nuclear industry. The new scanning technology developed at the Centre for Nuclear Energy Research (CNER) at the University of New Brunswick will allow monitoring of pipe infrastructure to be done in the hot environment of an active nuclear power plant with liquids flowing through the pipes. Their scanner,

called UTPro, detects flaws in pipes using guided ultrasonic waves. The UTPro system captures data, displays it for trained technicians to interpret and outputs the scan data in text files. The functionality of this sophisticated hardware would be greatly enhanced by software that could automate the process. Automating the analysis and adding extra functions to the graphical display would produce results that could be monitored and interpreted by workers with minimal technical knowledge. The author's Masters project was the development of such software for CNER. The author signed a 5 year non-disclosure agreement with CNER that prevents any code from the project and any algorithm details from being presented here; as much detail as possible is available.

1.2 Project Objective

The objective of the project is to design and build a piece of software that can read the output files of the UTPro scanner, organize the inputted scans, analyze them, and produce reports based on the data analysis. The data analysis required will consist of automated flaw detection and automated comparison of two different scans. This will allow scanning for flaws to be done more frequently and with more ease than has been previously possible. The software will also organize scans based on location in a plant and allow the user to view scans, manually mark flaws, edit the comparison results and export results to Microsoft Word and Adobe PDF.

1.3 Report Overview

This report contains five chapters. The second chapter provides background information about ultrasonic flaw detection in general, and specifically about the UTPro scanner. It also includes an overview of the terminology associated with scanning and terminology unique to this software. Chapter three is a discussion of the process of designing the software. First, it discusses the technology used to build the software; second, it describes the user interface; third, it details the decisions regarding data storage choices and methods; and fourth, it describes the output of the software in the form of reports generated. The fourth chapter documents the process of implementation. In this project the implementation stage contained the most interesting challenges, and the most significant of these was the design and implementation of two algorithms. The fourth chapter is organized around the relative significance of the challenges involved; it begins with a discussion of the creation of algorithms to manage automated flaw detection and scan comparisons, and then discusses the other problems involved in the implementation, the creation of databases, the importation and display of scans, and the generation of reports.

Chapter 2

Background

2.1 Introduction

This chapter gives a basic idea of how the UTPro scanner works and includes terminology for technical terms involved with the scanning technology. This chapter also includes terminology used by the Reporting and Analysis Software. The UTPro scanner detects flaws by transmitting ultrasonic waves through a medium and measuring the time required for the waves to reflect back to the scanner.

2.2 Flaw Finding

Ultrasonic waves can be used to detect flaws in materials. This is done by using a transducer (Figure 2.1) to emit an ultrasonic wave inside a material

(Figure 2.1); any flaws in that material will cause a portion of the wave to be reflected back to the transducer (Figure 2.2). The transducer also receives and can measure the intensity of the returning wave over time.

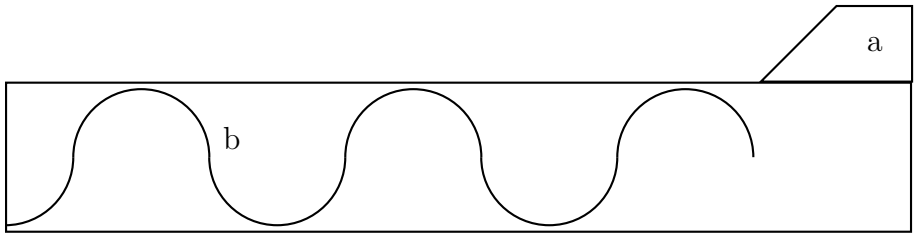

Figure 2.1: An ultrasonic wave in some material
a) ultrasonic transducer
b) ultrasonic wave

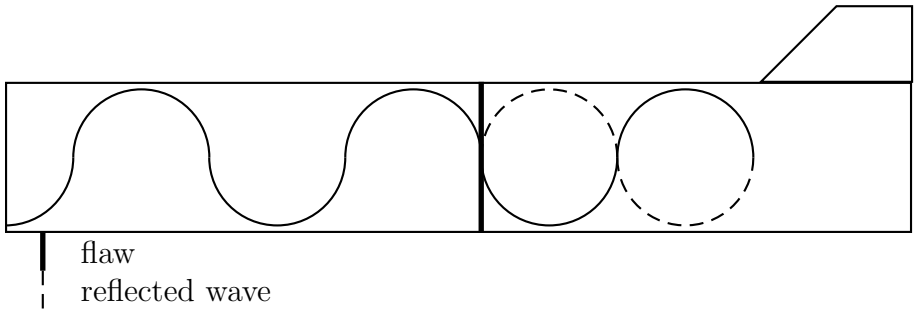

Figure 2.2: An ultrasonic wave reflecting in some material

By using the speed of the wave in the material it is traversing and measuring the return time of the ultrasonic wave, it is possible to accurately locate flaws using the formula

$$d = \frac{t \cdot v}{2},$$

where d is distance between sensor and flaw,

t is time between pulse emission and pulse return, and

v is the speed of the wave in the material.

Owing to the fact that creating the initial wave pulse will send some reverberations back to the transducer, the first few microseconds may not be recorded by the receiver.

Scan effectiveness is limited to a few metres in length because of energy loss in the wave and pipe features, which can obstruct or reflect the wave.

2.2.1 UTPro Scanner

The UTPro scanner was developed by Research and Productivity Council (RPC) for CNER and it uses ultrasonic waves to detect flaws in heated-fluid pipes. The UTPro scanner uses a ring of transducers around the pipe to locate flaws in the pipe. (Figure 2.3).

2.3 Terminology

2.3.1 Scanning Terminology

2.3.1.1 Ultrasonic wave

An *ultrasonic wave* is a sound wave with a frequency higher than humans are capable of hearing. These waves have a frequency of at least 20 kHz (20,000 wave peaks per second). The waves used by the UTPro sensor are around 1

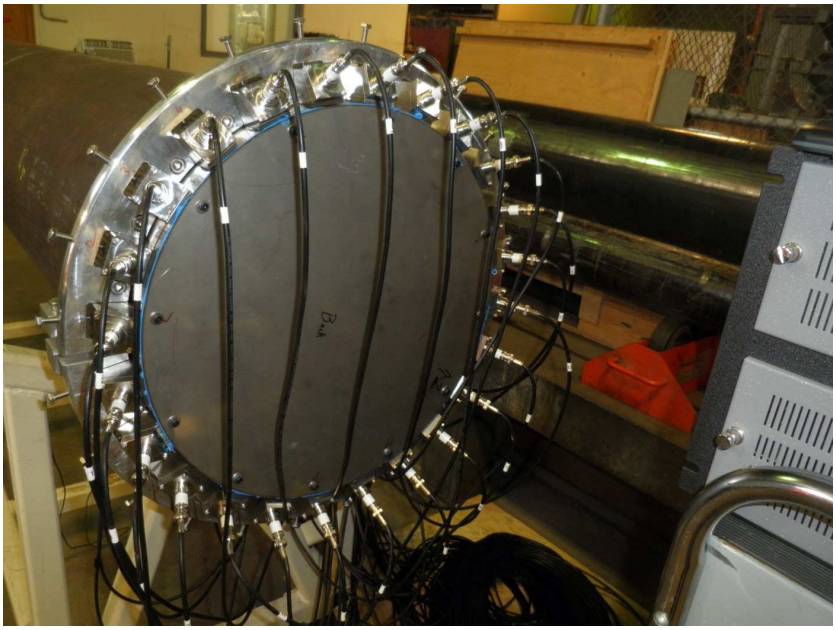

Figure 2.3: UTPro transducer set-up

MHz (or 1,000,000 wave peaks per second).

2.3.1.2 Transducer

A *transducer* is a device that can both emit and receive ultrasonic waves. They are usually made of a material that can be vibrated at high speed using electricity and whose vibrations can be read electrically.

2.3.1.3 A-scan

The results from a single transducer are called an *A-scan*. An A-scan is the measurement of the intensity of the returned wave over time. An A-scan will resemble the graph in Figure 2.4.

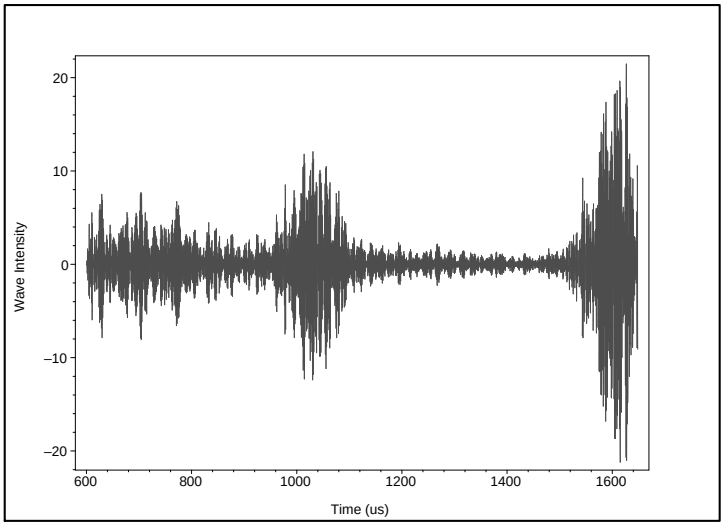

Figure 2.4: A sample A-scan

2.3.1.4 B-scan

If several A-scans are placed side-by-side, the result is called a *B-scan*. A B-scan is a three dimensional graph where the x, y, and z axes are time, transducer number (which is distance across the material), and wave intensity respectively.

A single B-scan from the UTPro sensor consists of multiple A-scans (one for each transducer), arranged so that every point along a given y-value is from the same transducer and every point along a given x-value reached the sensor ring at the same time. Since there are multiple transducers on the pipe, there are multiple points on the y-axis for each x-axis value, so to have a complete picture of the pipe the intervening y-values are filled in using linear interpolation. Figure 2.6 is an image of a B-scan where the z-axis values are represented using colour.

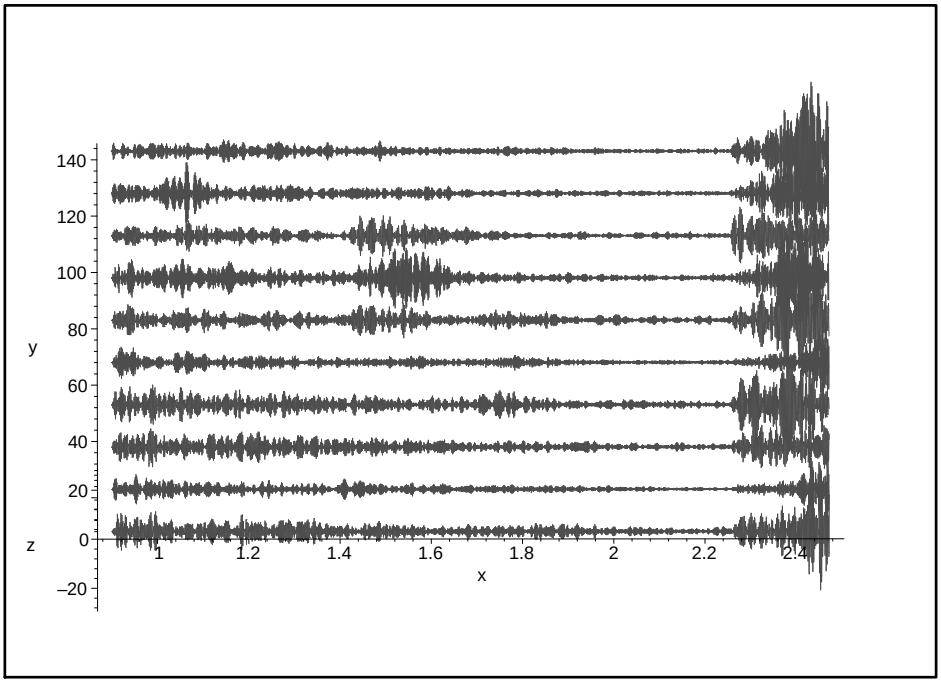

Figure 2.5: A-scans forming a B-scan

2.3.2 Software

Below is terminology used in the software to describe the pipe segments that are scanned and how they are categorized and organized.

2.3.2.1 Client

A *client* is a company using the UTPro sensor with one or many plants.

2.3.2.2 Plant

A *plant* is a discrete unit owned by the client, consisting of all or part of an actual plant which will contain one or more pipe systems.

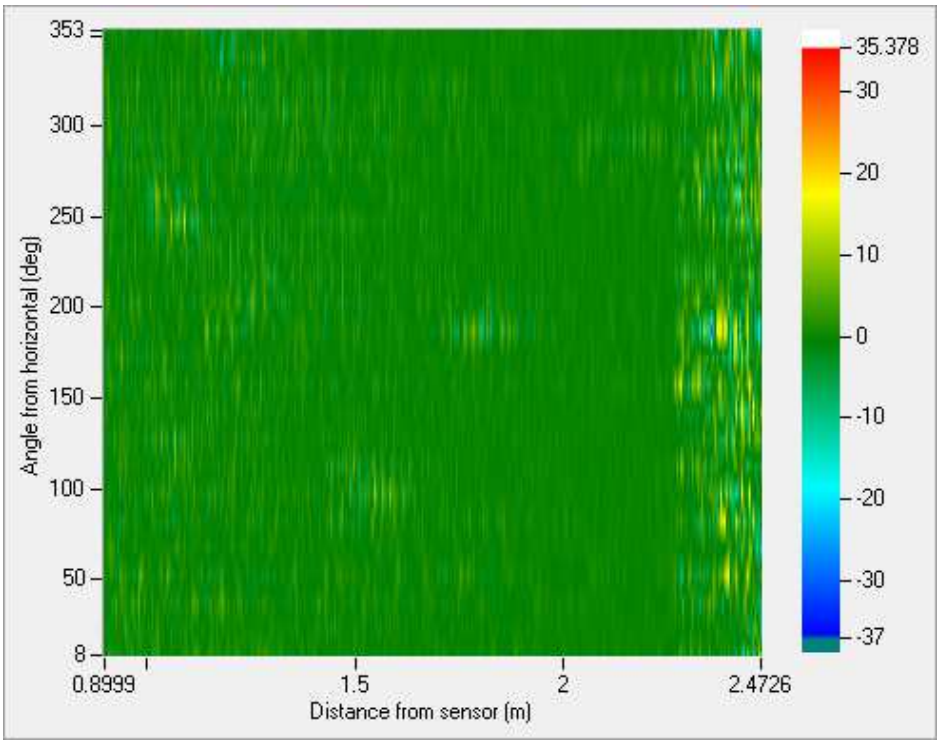

Figure 2.6: A sample B-scan

2.3.2.3 Pipe System

A *pipe system* is a connected system of pipes inside of a plant that will contain one or more pipe segments.

2.3.2.4 Pipe Segment

A *pipe segment* is a short length of pipe, typically a few metres in length, that can be scanned without moving the sensor. Every scan imported into the software will be associated with a pipe segment that was predefined.

2.3.2.5 Pipe Feature

A *pipe feature* is a material characteristic or fixture on the pipe that could affect scanning such as welds, hangers, valves, etc. Pipe segments can have one or more of these pipe features. One pipe feature will be designated a *landmark* which means that the location of every other feature and the sensor are given relative to it. This is to make it easier for users in the plant to consistently place the sensor in the same location so the sensor scans the same part of the pipe each time it is used.

Chapter 3

Design

3.1 Introduction

Software design is a critical part of software development because a good design both structures the process of implementation and determines the usability of the final product. A poor or implicit design can hamper the implementation by requiring redesigns during implementation and causes low usability which can negate product functionality.

A software design has three parts: frontend, backend, and data storage. This project is mostly frontend and data storage, because the software is centred around the analysis of the scans and there is no complicated control flow. Although it was not difficult to design, the user interface required the most attention in the design phase because it was necessary to learn how the engineers understood the task.

Working with the engineers at CNER was important to design the project. The author had no knowledge about power plants and how they are structured and organized, so it was necessary to work with the engineers to learn how the power plant engineers would locate and identify the pipes in the plant. This was important to the design so that the final software would have an intuitive feel for the people who would be using it.

This chapter discusses first the technological constraints that can affect design, and then discusses four design tasks in the project. The first two design tasks were the user interface and the report. Both of these came first because they informed the third and fourth tasks, the design of the databases and other data storage.

3.2 Technology

CNER chose to have the Reporting and Analysis Software written with Microsoft Visual Studio 2012 in Visual Basic.NET 4.5 for their own ease of use and maintenance.

The author created the databases using the Visual Studio database designer and they are stored locally using the default XML output. Since nuclear power plant data is of a sensitive nature it was felt that local storage was the most effective approach to deal with the inherent security issues of connecting to a remote database (i.e. by not having one). Using the Visual Studio database system also lessens the number of extra packages that need

to be installed with the software.

The National Instruments Measurement Studio 2013 for Visual Basic.NET was used to display the scans. Measurement Studio is a graphing and calculation package for scientists and engineers who are writing hardware control systems. Measurement Studio was chosen by RPC (the company developing the hardware) for the software that runs the UTPro hardware and is used for this project to ensure that displayed scans looked the same on both systems.

3.3 User Interface

The user interface was the first part of the software to be designed because the author needed to understand how to structure the software for engineers in the power plant industry. The author created the user interface with the engineers; through this process the author learned how they see power plants and the associated terminology. This allowed the author to design software that would be friendly to the target user.

Once the basic structure of the plant was inputted into the software the user has to be able to easily link the scan to a location in the plant, and direct the software to fulfil the primary task of data analysis. Thus, the user interface has two main functionalities: input of the plant data and structure, and display of the scans, analysis results, and reports.

In order to make it easy to compare two different scans done on the same piece of pipe and in order to generate useful reports, the software needed

to be able to associate scans with a segment of a pipe in the plant. The most intuitive way to organize the pipe segments was to organized them the way they would be organized in a plant. This intuitive organization is the schema of: pipe segments in pipe systems, pipe systems in plants, and plants belonging to clients. This schema lent itself to the creation of a dialog chain, with a separate dialog for each of these four elements that opens the dialog for the element below it. Since these elements correspond to real world structures that are unlikely to be changed; information about them will only need to be entered once and in an order that flows with the dialog chain structure.

The author decided that the majority of the software's main window should be used for the critical functionality: analysis and reporting. Since they are equally important, the author decided to use tabs for switching between flaw detection, scan comparison (called historical view in the interface), and reporting (see Figure 3.1 for a mockup of the main window). Importing scans, loading scans, creating reports, and loading reports are all done by single dialogs opened from the menu.

3.3.1 Flaw Marking

To indicate flaw locations on the B-scan it was decided to draw a circle around them and then list the coordinates of every flaw marked; flaws could be marked in two ways. Flaws could be marked automatically by the software, or flaws could be marked manually by the user. To manually mark a flaw the

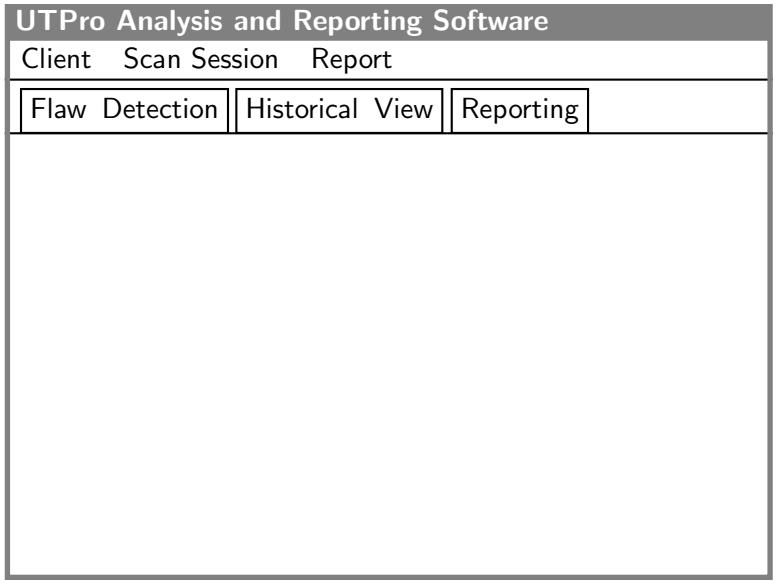

Figure 3.1: Main Window

user clicks on the B-scan and a circle appears where the user clicked. After placing the circle the user then presses a button to inform the software that there is a flaw where the circle was placed.

3.4 Report

The CNER required that the software have the capacity to generate written records so that text reports could be generated and filed as a permanent record for the organization. The design of the report capacity of the software was straightforward. It was based on the engineers' recommendations of what they need and the various elements of the data analysis that they would be examining.

The report was designed with five main parts: header, report and plant and pipe system information, pipe segments, B-scan images, and footer. The header contains the report number, report title and the page number. The report and plant and pipe system information parts contain all the information about the report, plant, and pipe system that CNER determined needed to be in a report. The pipe segment part contains information about each pipe segment included in the report and every flaw marked on that pipe segment. The B-scan images part contains the B-scan images of each B-scan with various display options for each pipe segment in the report. The footer part contains a description of the x and y units used in the report.

3.5 Databases

The databases hold all the information about the scans, all the information on the plants, all past flaws found by scans, and information on all the reports generated. The creation of these databases was straightforward, but a number of design decisions had to be made. First, where to store the databases; second, what kind of database to use; and third, how to interact with them. Regarding the first, there were two options, an external server, or local storage. The author chose the latter in order to avoid dealing with the network security protocols of the nuclear industry. Cyber-security is a serious threat in this industry and eliminating the network involved in an external server eliminates the threat of security breaches. The author was

careful to design the software so it would not be difficult to switch to an external server if this was desired at some point in the future. Regarding the second decision, of the kind of database to use, the author used the default Microsoft databases because it was adequate for the purpose. This avoided the use of any third party software which might then introduce glitches with future installations. Finally, regarding the interaction with the databases, there were two possibilities, either a direct interaction or through an intermediary class. The author chose the second because it would make it easier for someone at a later date to change the kind of database and the storage method.

The database required to manage the data was relatively straightforward to design. A database is founded on the relationship between various levels of data: plants, pipe systems, pipe segments, and pipe features; and the database needs to reflect this in such a way that manipulation and processing is facilitated. Three databases were required, one each for of the client, the scans, and the reports; each of which are detailed in a section below. Each database is stored locally using the default XML format when not in use.

3.5.1 Client Database

The client database does not contain most of the information that is directly about the client; instead it contains all information on plants, pipe systems, etc. The only information about the client that is stored in this database are the client logos, which are associated with a dummy plant called "client"

that is created by the software.

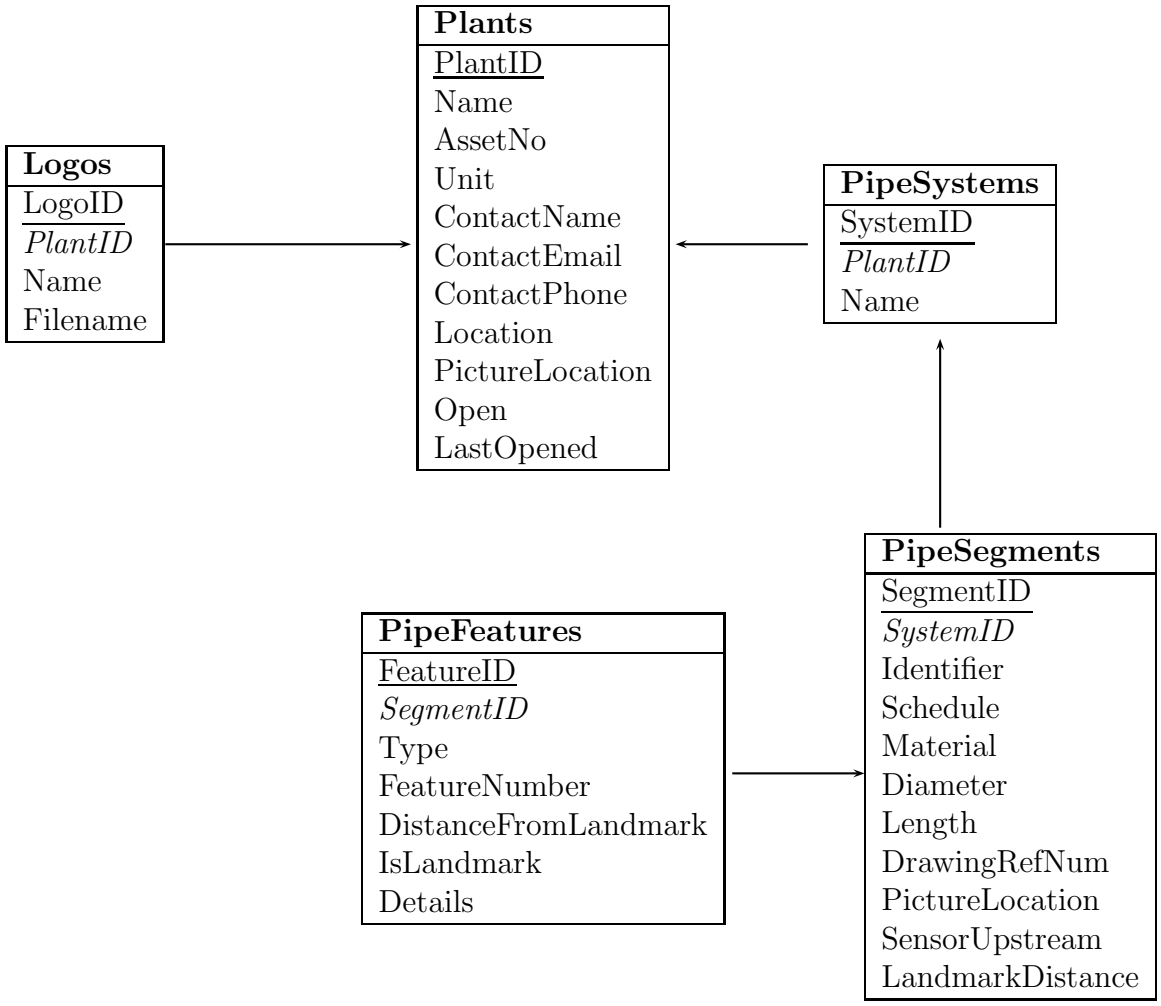

Figure 3.2: Client Database Model

3.5.2 Scans Database

The scans database contains the location of the scan data, connections to associated pipe segments in the client database, all marked flaws, scan com-

parison results, and information about the particular scanning device used (in the event that a client has more than one).

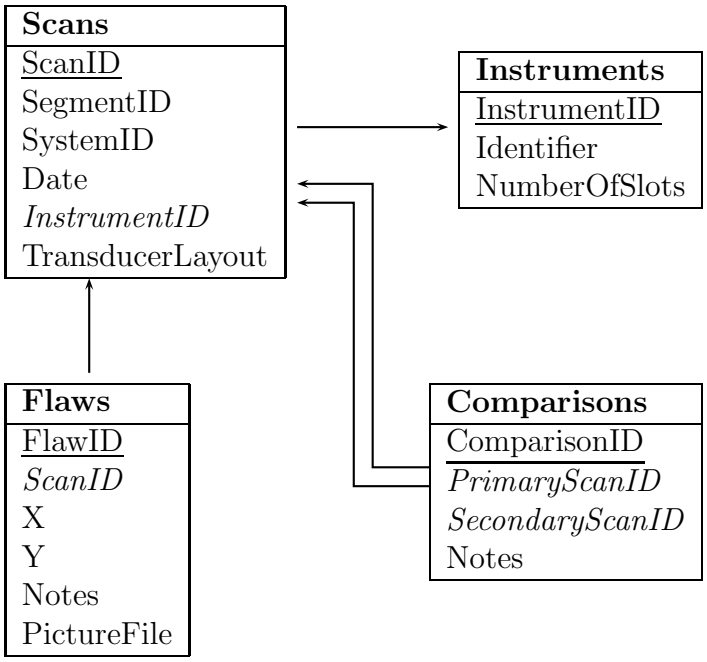

Figure 3.3: Scans Database Model

3.5.3 Reports Database

The reports database is organized so each tab in the reports dialog has a corresponding table in the database and a table containing report number prefixes. For each report there will be exactly one row in each table that corresponds to a tab except for the PipeSegments table which contains a row for each pipe segment in each report. These rows are linked by a unique value for ReportID which is an automatically incrementing primary key in

the Reports table and both a primary and foreign key in the other tables.

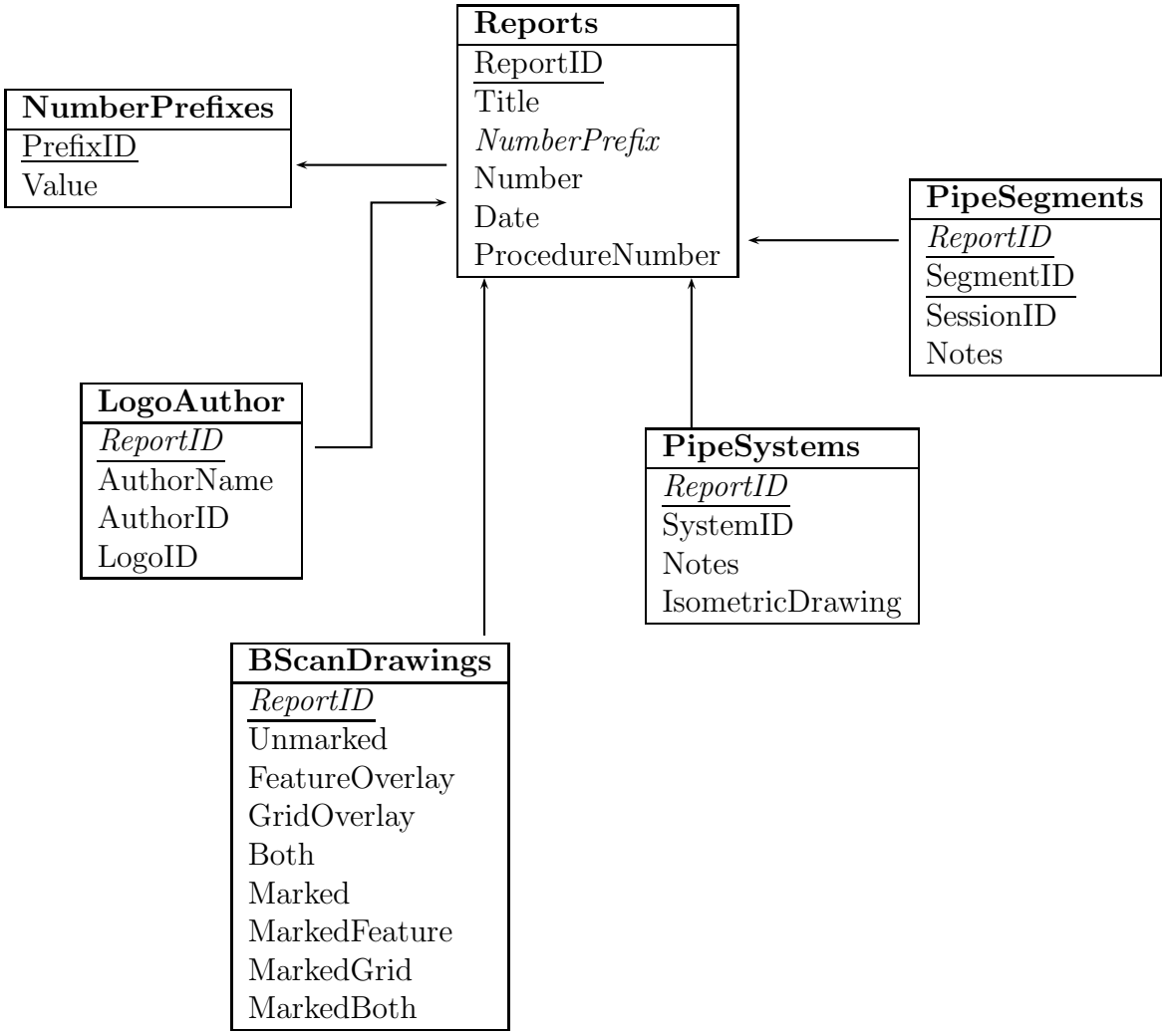

Figure 3.4: Reports Database Model

3.6 Other Data Storage

Two kinds of data were not stored in the databases: data for which only one instance was required, since a table with only one row is pointless, and data which was so large that loading it into active memory would be prohibitive. Data of the first type, single instance, includes client information, a list of possible valve types, settings for automatic flaw detection, and settings for the display of B-scans. Data of the second type, really big data, includes any images uploaded by the user and the B-scans themselves.

3.6.1 Client File

Information about the client, such as contact information and location, are loaded into a unique instance of the Client class from an XML file with the format shown in Figure 3.5.

```
<Client PlantID=[PlantID of dummy plant in client database]>
  <Name>[client name]</Name>
  <Address Count=[number of lines in address]>
    <Line Index=[the line number]>[value of the line]</Line>
  </Address>
  <Contact>
    <Name>[contact name]</Name>
    <Email>[contact email]</Email>
    <Phone>[contact phone]</Phone>
  </Contact>
</Client>
```

Figure 3.5: Client XML Format

3.6.2 B-Scan Data

The UTPro hardware saves its data using a plain text format, and since scans consist of thousands of points this is not space efficient. Since the software is designed to have many scans saved with it, it was important to create a more space efficient format for saving the B-scans. After a B-scan is loaded into the software from the UTPro .bsc format, it is saved by the software as a .bin file in a binary format outlined in Figure 3.6.

⟵ 4 bytes ⟶	⟵ 4 bytes ⟶	⟵ 4 bytes ⟶	⟵ 4 bytes ⟶
number of points	total number of transducers	frequency of point capture	
speed of wave		time delay before capture start	
amount of circumference scanned		minimum z-axis value to display	
display type	number of transducers used	transducer 1, value 1	transducer 1, value 2
···	transducer 1, last value	transducer 2, value 1	···
transducer 2, last value	···		last transducer, last value

Figure 3.6: B-scan binary file format

3.6.3 Other Data

All pictures that are uploaded by users for any use are saved with the data as JPEG's. The automatic flaw detection settings, colour options for B-scan display and the valve types are saved in binary files. Automatic flaw detection settings and colour options have fixed length files. Valve types are written in one line with a special character separating different types.

Chapter 4

Implementation

4.1 Introduction

The implementation of the software contained four significant challenges. The major challenge in this project was coming up with the algorithms for the main functions of the software. A second challenge was designing the software to display the B-scans in such a way that allowed flaw locations to be visually indicated. The third challenge was the report generation, and the fourth, minor, challenge was the creation of the databases. The implementation also included more routine tasks such as the user interface, and the author has omitted the details of such routine tasks.

4.2 Algorithms

4.2.1 Flaw Detection

The capacity to detect flaws is the central function of the software. It was the critical objective from the point of view of the client because it allowed them to avoid the cost of hiring specialists to do this work. It was also the most challenging part of the project because the flaw detection had to mimic human cognition. There are elements of flaw detection which appear to be more art than science; an analogy would be the interpretation of x-rays. The work of scan analysis was done by an experienced technician who was able to interpret the visual scan patterns and draw conclusions about the state of the pipes. The author's task was to create software that could automate this process of recognizing and evaluating flaws in the pipes. Automating the discernment involved in human visual capacity is difficult. Recognition of flaws is, at first glance, easy because a flaw presents itself as a spike in the wave, but other factors also have the same effect so the software has to be able to evaluate the likelihood that a flaw, and not another benign factor, has caused the spike in the wave.

As previously mentioned, flaws can be marked in two ways: manual or automatic. Manual flaw detection is done by users clicking on the B-scan image to mark a flaw or automatically done by the software.

Manual flaw detection takes 5 steps which are listed below.

1. Click on the "Start Marking" button

2. Click on the B-scan image until satisfied that the circle is in the correct location

3. Click on the "Mark Flaw" button

4. Enter any notes into the flaw dialog

5. Click the "OK" button

To aid users in pinpointing flaws a zoom function was added to the B-scan display. The zoom function, displayed in Figure 4.1, allows users to zoom in on a section of the B-scan by entering bounding coordinates into the relevant boxes and clicking the "Limited Region" button; to reset the zoom users click the "Full Region" button which resets the values.

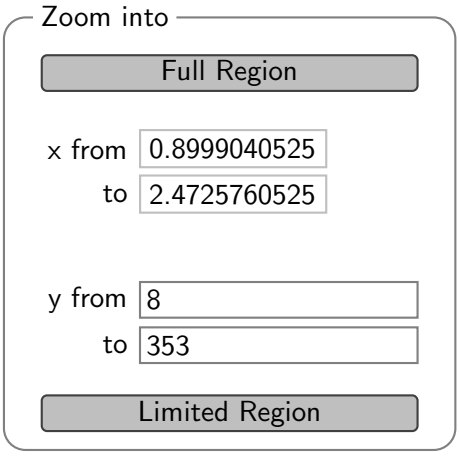

Figure 4.1: B-scan zoom function

Automatic flaw detection is performed by the software and is controlled by seven user-defined parameters; these parameters are detailed in Figure 4.1.

Users can edit these parameters by clicking on the "Auto Locate Settings" button which displays a dialog. Users initiate automatic flaw detection by clicking the "Auto Locate Flaws" button; they will then be asked if they wish to clear all currently marked flaws before automatic flaw detection begins.

Parameter Name	**Parameter Type**	**Parameter Function**
EstimatePipeEnd	Boolean	Determines whether or not the software should estimate the width of noise at the end of the scan caused by a pipe end or a pipe feature that prevents further scanning
PipeEndValue	Integer	User entered value for the width of the end of pipe noise in centimetres
AveragePeakHeight	Boolean	Determines if the software should use average peak height as minimum flaw height
PeakHeightValue	Single	User entered value for minimum flaw height
PeakRadius	Integer	User entered value for a radius within which two peaks are considered to be within the same flaw, measured in millimetres
IgnorePipeFeatures	Boolean	Determines if the software should ignore peaks "near" pipe features
PipeFeatureRadius	Integer	Radius around a pipe feature to ignore peaks within, measured in centimetres

Table 4.1: Automatic flaw detection parameters

4.2.2 Scan Comparisons

This algorithm was slightly easier to create than the previous one because it used the flaw finding technique developed for flaw detection to measure flaw sizes. The task was to, as elegantly as possible for the user interface, display what had changed between the scans. This allows the engineers to detect new flaws, recurrences of a flaw in the same location, and the worsening of existing flaws. The challenge for software design is that two scans to be compared may not have easily comparable variables because the scanner might not have been in the same place when the scans were done, or temperature difference may have resulted in different speeds and different wavelengths. This requires that the software be able to convert data to make the comparisons valid.

Several methods were discussed for what factors to consider when comparing two scans. Initially there were several options available; comparing only peaks in both scans, comparing changes in marked flaws from one scan to the other, or both. For simplicity's sake it was decided to just have one comparison algorithm available. The resulting algorithm compares all flaws marked in the previous scan to the highest peak in the same location in the more recent scan. The only parameter the comparison uses is the minimum peak height of a flaw. The comparison dialog is passed this value by the historical view control in the main screen which takes it from the display parameters the user has set to view the B-scans being compared. The steps of the comparison algorithm are outlined below.

1. Choose a flaw in the previous scan that has not been analysed.
2. Draw a box around it based on minimum peak height.
3. Draw the box found in Step 2 in the recent scan.
4. Find the highest peak in the box drawn in Step 3.
5. Draw a box around the peak found in Step 4 based on minimum peak height.
6. Compare the height of the flaw chosen in Step 1 and the peak found in Step 4 and record the results.
7. Compare the length and the width of the box drawn in Step 2 and the box drawn in Step 5 and record the results.
8. If not all flaws have been analysed, then return to Step 1.

The algorithm results are formatted in sentence form and output to the results textbox in the dialog. Users can then choose to edit these results if they choose.

4.3 Importing, Displaying, and Marking B-Scans

A software package, the National Instruments Measurement Studio, was included in the RPC scanner software. This package included several possible

ways to display the B-scans. The default was a 3D plot, which made marking flaws very difficult. Instead, the author used an intensity graph, a 2D display, making marking much more simple.

The second issue with the B-Scans was the storage format. The format that the scans were saved in was text-based and was a space hog. It was important that the software be space efficient, given its use by a large power plant with many pipes and thus scans. The author created a new space-efficient binary format for saving scans.

For space efficiency it was decided not to load all of the B-scans into memory during runtime. Instead when users decide to import a B-scan the .bsc file is read and its contents are loaded into memory and then are saved using the space efficient format described in Section 3.6.2. Every time users open a B-Scan after the initial load the data is read from the binary format.

4.3.1 Displaying B-Scans

The software that runs the UTPro hardware used a 3D graphing class from Measurement Studio to display the B-scan. Using this method caused two problems in the analysis and reporting software. Firstly, flaws needed to be able to be marked on the B-scan display and no reliable method was found for doing so. Secondly, when attempting to publish the software for installation, the 3D graphing package did not appear to be easily includable and no workaround was found within the project timeline. To solve the second problem, another class, called an intensity graph, was used. The

intensity graph would plot a set of 3D data by using colours to indicate the z-axis values which looked identical to the 3D graph laid flat.

A more complicated solution was needed for the first problem. Since it was not possible to draw on the intensity graph itself, and the 3D graph proved impossible to draw on visibly when lain flat, a custom B-scan display class had to be created. The custom class contained a Measurement Studio intensity graph and used the Visual Basic.NET picture class for display. The intensity graph class has a method called "DrawToBitmap" which draws the visual contents of the intensity graph onto a bitmap object passed to it. The custom display class called this method and passed to it the background image property of the picture object. The image property of the picture object could then be drawn on and erased without effecting the B-scan display. Using a custom class allowed for much better control of the display as well.

The custom display class was connected to the Scans database which raised an event whenever a flaw was added or deleted so that the display could update instantly. The custom class made the use of grid and pipe features overlays much easier to implement, as they could be controlled by a property in the class. The custom display class also handled showing the flaw dialog and the automatic flaw detection since it contained the B-scan data that was loaded into memory.

4.4 Report Generation

The task of report generation required the adaptation of an existing system so that it was more functional for the user. The default Microsoft program produced reports in the form of tables of data pulled from the database; however it has limited usability. A report of the complex pipe system in the plants needs to be able to be flexible enough to report on as many or as few pipe segments as the engineers need to see, each segment represented in the report with a variable number of possible flaws. A good option would be to have a separate table for each segment, but Microsoft's program did not allow for this because report designs had a hardcoded number of tables. The author's research on alternate programs found that many people used Crystal reports in such cases; however, the author rejected this option because of the potential for third party programs to produce glitches in subsequent installations. Instead, the author chose to work with the Microsoft system and tweaked the program to produce a more functional reporting system for the project.

The author's adaptation of Microsoft's program used a system of nested subreports. In all, there were four different report setups used: main report, pipe segments, flaws, and B-scan images. In Microsoft Reports each table is connected to a database table and the rows of the table will be repeated for each row in the database. Each table can have conditions that restrict the number of rows in the database — such as $SystemID = 1$. For example, to

have a subreport repeated for each pipe segment in a particular pipe system there would be a table with two rows, one containing the relevant subreport and one containing a row to act as a spacer (see Figure 4.2). The table would be linked to the PipeSegments table of the Client database with the condition $SystemID =$ <value of SystemID for chosen system>. For each row of the PipeSegments table that satisfied the condition — i.e. a row where the SystemID was desired — the two rows of the table in Figure 4.2 would be repeated.

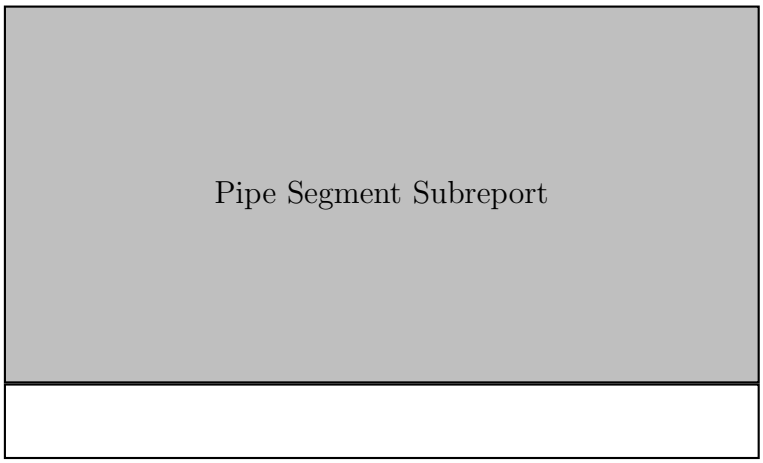

Figure 4.2: Subreport table

The main report setup has seven different parts, which are: header, logo and report information, plant information, pipe system information, pipe segment report, B-scan images report, and footer. The header appears on the top of every page of the report; it contains the report number and title on the left and the page number, formatted "<page number>of <page count>", on the right. The logo and report information contains the logo

chosen by the user and all information that is about the report itself, such as report title, author name, etc, this section also contains a space for the author's signature. The plant information section contains the name of the client, the plant, and the plant's asset and unit numbers. The pipe system information contains the pipe system name, any notes that the user input, and an isometric drawing of the pipe system if included by the user. The pipe segment report section contains the table that has the pipe segment subreport in it. The B-scan images report section has the table that contains the B-scan images subreport. The footer section appears on every page of the report except for the first and contains a brief description of how the x and y units being used in the report are measured and a helpful diagram.

The second report setup is for pipe segments. The first part contains information about the pipe segment and the scan that is being displayed. The second, a table with one row for each flaw marked on the scan. Each of these rows contains a subreport, which is passed the flawID of the relevant flaw.

The third report setup is for flaws. The first part of this report contains information about the flaw in a table. The second contains a picture box with an uploaded picture of the flaw, or is invisible if no such picture has been uploaded.

The fourth report setup is for B-scan images. The user can choose to display any combination of unmarked B-scans, B-scans with the grid overlay, B-scans with the feature overlay, B-scans with both overlays, B-scans with

flaws marked, B-scans with flaws marked and the grid overlay, B-scans with flaws marked and the feature overlay, and B-scans with flaws marked and both overlays. The B-scans images report setup contains eight tables: one for each of these options. If the user has chosen a particular option then that table will be displayed for each of the pipe segments included in the report.

4.5 Databases

The databases are created using the database system internal to Visual Studio. This choice was made because it reduced the number of extras that would be needed to be installed with the software and since it was decided to store the databases locally, a system like SQL was not required. The databases are stored using the default XML format and are loaded into memory at runtime. They are only saved to the hard drive when the program closes.

Each database has an associated control class, of which there is exactly one instance. The control class for each database contains an instance of the database which is loaded at runtime and methods for searching, adding to, editing, and deleting from the database itself. The control class also has backup and restore methods for the database which are used to revert the database to a previous point if changes were made to the database and then canceled.

Chapter 5

Conclusion

The project was a contract for CNER to develop a piece of software to go with their new specialized UTPro scanning hardware that would do automated analysis and reporting of ultrasonic scans to detect flaws in pipes. This was needed because otherwise the analysis must be done by hand, which is expensive. The author designed and built a stand-alone windows application to accomplish these goals.

Much of the software development in this project was fairly straightforward; for example, the user interface and the creation of the databases posed little challenge. The challenges were found in the development of the algorithms, the creative task of designing what would be an intuitive and versatile tool for the engineers, and the implementation of said design. The project required the development of two algorithms: one for finding flaws in the pipes, and the other for comparing scans of pipes.

Two parts of the software design were extremely challenging in terms of implementation. First, the component that would allow the engineers to display the scans of the pipes and mark flaws found. This problem was solved by creating a custom class that would allow for display and marking. The second part was the format of the reporting, which was designed by the author and the engineers in concert. An issue emerged because the modularity needed by the engineers was hard to achieve. Crystal Reports was suggested as a solution, but there are drawbacks to integrating third party software, so instead the author created a system consisting of a series of nested subreports that would allow the level of modularity desired by the engineers.

The two most significant and interesting parts of the project were the algorithm for flaw detection and the algorithm for comparing two different scans. The details of these algorithms are protected by a NDA.

Finally, one of the key challenges facing the author was the implementation of a design which followed the way the engineers organized information about the powerplants. This required the author to learn the language and work habits of the engineers who would be using her software, in order to map those to her design.

In the final result, it was not possible to determine how close the software came to the capability of a human technician. Unfortunately, the project did not have the resources to run sufficient tests to determine how close the software came to human capability. However, the software was able to pick

out flaws on pipes manually created for testing purposes.

Bibliography

[1] M.J.S. Lowe, D.N. Alleyne, and P. Cawley, *Defect detection in pipes using guided waves*, Ultrasonics **36** (1998), 147–154.

[2] M.H.S. Siqueira, C.E.N. Gatts, R.R. da Silva, and J.M.A. Rebello, *The use of ultrasonic guided waves and wavelet analysis in pipe inspection*, Ultrasonics **41** (2004), 785–797.

Vita

Candidate's full name: Thea Gegenberg
University attended (with dates and degrees obtained):

- University of New Brunswick (Master of Computer Science, October 2015)
- University of Waterloo (Master of Mathematics, May 2011)
- Simon Fraser University (Bachelor of Science in Mathematics 2009)

Conference Presentations:

- "Introduction to Special Lagrangian Submanifolds of $\mathbb{C}^n$", General Relativity Seminar, University of New Brunswick, July 14, 2010
- "The Special Lagrangian Differential Equation", Working Geometry Seminar, University of Waterloo, June 30, 2010
- "Calibrations and Special Lagrangian Submanifolds", Working Geometry Seminar, University of Waterloo, May 26, 2010