Final Report

Mobile Inspection Assistant (MIA)

A wearable computer prototype for bridge inspection
Table of Contents

1. Introduction and Overview of the MIA system 1
   1.1 Purpose
   1.2 Background
   1.3 Group Members
   1.4 Report Overview

2. Conceptual Design
   2.1 Problem Definition 4
      2.1.1 Problem Scenario
      2.1.2 Key System Requirements
   2.2 Initial Solution Concepts 6
      2.2.1 Table of Selected Technologies
      2.2.2 Visionary Scenario
   2.3 Conceptual Design 9
      2.3.1 Solution Scenario Selection Criteria
      2.3.2 Product Design Specifications
      2.3.3 Hardware Architecture
      2.3.4 Software Architecture
      2.3.5 Mechanical and Industrial Concepts

3. System Tutorial and Usage Scenario
   3.1 Screen Independent Fields 16
   3.2 Summary of Integrated User Interface 19
   3.3 Usage Scenario 22
      3.3.1 Using MIA
      3.3.2 System Tutorial

4. System Implementation
   4.1 User Interface Design 51
      4.1.1 User Interface Design Issues
      4.1.2 Example Bridge Inspection Reports
   4.2 Software Design 52
      4.2.1 Software Object Architecture
      4.2.2 Software Tools and API
      4.2.3 Software Tool Evaluation Studies
   4.3 Electronics Design 64
      4.3.1 Electronics Architecture
      4.3.2 Hardware Subsystems
      4.3.3 Electronic Subsystem Evaluation Studies
   4.4 Mechanical and Industrial Design
      4.4.1 Mechanical Design Concept
      4.4.2 Support Vest
      4.4.3 MIA Container
      4.4.4 Additional Components
      4.4.5 Materials
      4.4.6 User Feedback
      4.4.7 Future Design Goals
   4.5 Conclusions 74
      4.5.1 Requirement Feature Table
      4.5.2 Summary of Key Design Issues
      4.5.3 Lessons Learned
   4.6 Design Evaluation 76
5. Project Management
   5.1 Implementation Phase Results 77
   5.2 Ranked Issue List 78
   5.3 Suggestions for Class Improvement 78

Appendix A: Software Tools Object Model 79
Appendix B: Bridge Inspection Report 96
Appendix C: Mechanical and Industrial Design 105
Appendix D: MIA User Guide 109
Introduction and Overview of the MIA System

This document is the final report on the work completed by the Wearable Computer Project Design Course at Carnegie Mellon University, on the Mobile Inspection Assistant (MIA) project. The purpose of the project was to develop a wearable computer for bridge inspectors, to use while they are in the field.

1.1 Purpose

The purpose of the Mobile Inspection Assistant is to reduce the amount of paperwork and redundant data entry necessary for filing inspection reports, while providing more convenient access to previous inspection reports. The main goals of the project include minimization of the amount of time spent on paperwork, so that the inspectors can spend more time on the actual inspection, providing a user interface that can be easily used and providing near hands-free operation.

1.2 Background

There are nearly 590,000 bridges in the National Bridge Inventory, that are over 6.1m in length which must be inspected at least once every two years. There are many other bridges with spans less than 6.1m, that agencies must still maintain at a cost to taxpayers. Although advanced bridge management systems are being created, deployed and used to collect this data for bridge inventories, bridge inspectors in the field are still primarily using paper-based notes to support and record the results of their inspection processes. Consequently, the inspectors are currently limited in terms of the material they can reference, while in the field and the tools they can use to capture the nature and location of defects found on the bridge elements.

The computing needs of bridge inspectors cannot simply be addressed by giving them a laptop computer with desktop-oriented software to use in the field. This approach has already been tested and proved unsatisfactory.

In order to properly support a bridge inspector in the field, the inter-disciplinary team from Carnegie Mellon University, researched and designed an integrated system of hardware and software that recognizes the field-oriented nature of bridge inspection and provides inspector-friendly support for field data collection and decision making. A first prototype of the intended product has been developed.

The prototype was developed in three phases viz. conceptualization, design and implementation phases. In the conceptualization phase, field studies were conducted. The team interviewed bridge inspectors, attended a field inspection and a seminar on bridge inspection methods. Based on the requirements derived from talking with and observing bridge inspectors, the team created a visionary scenario of the product, and set out to develop a wearable computer for bridge inspection purposes in the next two phases.

In the detailed design phase, a design scenario was developed, by systematically walking through every step involved in using the product. During this process, several issues with respect to design and implementation were identified and resolved. Towards the end of this phase, the implementation of the software, hardware and housing of the product were done individually.

In the implementation stage, each of the units were tested and integrated to finally arrive at a working prototype. Regular feedback was obtained from the bridge inspectors at the end of each phase, to arrive at an efficient design.
1.3 Group Members

The team was divided into four subgroups, who worked hand in hand, to arrive at the first prototype.

Software Group
Amy Roch
Anitha Balasubramanian
Rebecca Buccheit
Bikram Baidya
Eric Seshens

HCI Group
Justin Hilderbrandt
Jason McDowall
Jirapon Sunkpho
Neeraj Bansal

Hardware Group
Gautam Kharkar
Barry Huie
Chris Lumb
Rob Migliore
Pace Lin

MEI Group
Julie Rodriguez
Lyren Brown
Alex Lozupone
Robert Slater
1.4 Overview

The remainder of this report is a description of the final design and implementation of the MIA project.

Section 2.0 summarizes the work performed in the conceptualization phase. It gives an overview of the requirements that influenced the system design.

Section 3.0 is a description of how to use the system, presented in the form of a tutorial. Illustrations of the user interface are provided wherever possible.

Section 4.0 documents the system design, as it is currently built. Section 4.1 describes the user interface design, which details the process of determining how the user interface will interact with the system. Section 4.2 contains the software design, which describes the software tools that were chosen and how they were implemented. Section 4.3 deals with the electronics design, which enumerates the hardware components used and the reasoning behind their implementation. Section 4.4 constitutes the mechanical and industrial design, which explains the physical shape of MIA and the research behind it.

The concluding section of the report shows how the design maps to the requirements chosen in the problem definition. In addition, it enumerates possible areas for improvement.
2. Conceptual Design

This section gives an overview of the key requirements obtained as a result of field studies conducted in the conceptual phase, and a preliminary design scenario developed on the basis of these requirements.

2.1 Problem Definition

Most often, bridge inspectors have to perform their monitoring and inspection tasks in harsh environments. In order to collect bridge condition data, the bridge inspectors must walk and climb on bridges. Due to the nature of the inspection, inspectors need to have their hands free to support climbing and inspecting activities, and cannot carry reference manuals, previous inspection reports or other information with them onto the bridge. In addition, field inspectors typically prepare sketches to guide their inspection and to facilitate the recording of what they see in the field. The inability of an inspector to access supporting information on field, influences the efficiency and quality of the collected data.

The objective of the project is to develop a wearable computer prototype to help bridge inspectors collect information on the field.

2.1.1 Problem scenario

The problem scenario is the summary of the findings from the field evaluation work in a concise, simple format in order to ensure everyone understands and is solving the same problem. The problem scenario is part of the user centered design process and it is used to generate the system requirements. The following are the seven steps in problem scenarios for bridge inspection.

Scenario 1: Retrieve, Review

Our friendly bridge inspector, Joe, is sitting in his office, on a fine morning. He makes himself a nice cup of coffee and looks up the bridge schedule that needs to be inspected next. After determining the bridge, he walks down to the file cabinet room where all the bridge files are sorted according to the bridge identification number. He looks for the file and signs it out. He then returns to his office and reviews previous reports, schematics and layout plans of the bridge. He identifies problem elements and components and proceeds to make a list of them. He also looks at specific comments by the inspector who did the previous report. He makes some notes about critical elements. He is now ready to make an action plan.

Scenario 2: makes action plan

The inspector is now at his desk and is ready to prepare an action plan for the bridge inspection. He finds his favorite clipboard and maps out a plan of what he wants to accomplish based on his results from scenario 1. He prepares some sketches of elements that he wants to look up and arranges them in a priority order. He ensures that he knows what equipment he will need, and is now good to go.

Scenario 3: goes to site

The inspector now gets in his van, and ensures all the required equipment is there. He finds the map with the bridge location. He looks at site pictures from the previous inspection. Coupling that with the map, he drives to the site.

Scenario 4: The Inspection
The inspector is at the site. He systematically proceeds through his action plan and the
form. He inspects and make sure the roadway is good, then moves systematically through the
elements of the bridge. He annotates any damage on a component, takes a photograph if
necessary, or makes a sketch. Then he may work on any measurements. He finds a missing
bolt, and sprays on it to ensure that it is possible to easily identify it at the time of repair. He
measures cracks and records measurements with a piece of chalk. He works his way through
the entire bridge in this manner.

Scenario 5: Preparing the Report

The inspector is now in the van, and he looks through all of his data sets. Then he
completes the rest of the form, followed by a quick run back to ensure any additional data he
might need, or just to verify what he is entering. He makes the appropriate references to
to pictures and sketches. Finally he enters all the statistical data. The report is complete and
ready for filing.

Scenario 6: Identify items for repair

The inspector is in the van, and he finds a repair maintenance form. If there is an
emergency, he will call the police to block a section or the whole bridge, for safety purposes.
He fills out the form with a description of the location and nature of problem. He submits the
form when he returns back to the office.

Scenario 7: Filing Report

After the pictures are developed, he compiles the entire report, files it into the
corresponding bridge file, and submits it to the archive. This concludes the inspection.

2.1.2 Key System Requirements

The problem scenario gives an overview of the existing bridge inspection methodology.
From the problem scenario, certain key system requirements were identified, which must be
necessarily provided in the prototype, to facilitate bridge inspection.

- The inspector must be able to record damage information with words, sketches, and photos.
  Annotation of pre-made drawings of the key bridge elements should be designed. In addition, he
  should be provided with a tool that graphically displays the results of component monitoring.

- A digital camera that allows pictures to be reviewed would be extremely useful, because it ensures that
  a good image has been captured.

- Input devices should allow for the fact that the inspectors wear heavy work gloves.

- Handwriting recognition should not be made the primary mode of annotation, owing to the previous
  unsatisfactory experience with a poor version of the Fujitsu software. However, it should not be ruled
  out as a back-up mode of annotation.

- Speech Recognition would be the ideal mode of annotation since they ensure near hands-free operation
  and avoids typing. However, these systems would have to withstand high noise environment.

- A touch screen that uses a magnetic pen is important for input. A touch screen that accepts all touch
  input would probably not work well in a dirty environment, especially if the user is wearing gloves.

- Wireless communications is required to link the home office from a site.

- The inspector requires backup capability in case of an accident.
• The inspectors desire a tool that will help reduce time spent on paperwork and increase time spent on inspection. Their goal is to have the bridge report done when they leave the site.

• The device must be impact resistant and able to withstand jostling and bumps. The device is required to withstand rain and possibly underwater immersion. The device should be capable of withstanding outdoor temperature variations of the Pennsylvania climate.

• The device may be tethered to the body to allow freedom of movement and also near hands-free operation.

• The screen must be legible under a variety of lighting conditions ranging from quite dark to bright and direct sunlight.

2.2 Initial Solution Concepts

The system requirements that were identified based on the problem scenario, served as a basis for deciding upon a set of target technologies, which were simultaneously addressed in accordance with the visionary scenario, and finally integrated to arrive at the prototype design.

2.2.1 Table of Selected Technologies

From the system requirements, a list of target technologies were identified [Table 2.2.1].

<table>
<thead>
<tr>
<th>TARGET TECHNOLOGIES</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pen Input Devices</td>
<td>To allow user input</td>
</tr>
<tr>
<td>Touch Screens</td>
<td>To allow user input</td>
</tr>
<tr>
<td>LCD Screens</td>
<td>To display application</td>
</tr>
<tr>
<td>Wireless Networking</td>
<td>Communication of multiple bridge inspectors. Communication between client/server units.</td>
</tr>
<tr>
<td>Digital Camera</td>
<td>Annotation of bridge reports</td>
</tr>
<tr>
<td>Storage Devices</td>
<td>Storage of all necessary data</td>
</tr>
<tr>
<td>Handwriting Recognition</td>
<td>Annotation of bridge reports</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td>Annotation of bridge reports</td>
</tr>
<tr>
<td>Sketch Software</td>
<td>Free hand sketching on site</td>
</tr>
<tr>
<td>Monitoring System</td>
<td>Monitor damage on any given bridge component, graphically</td>
</tr>
<tr>
<td>Databases</td>
<td>Organization and storage of inspection forms</td>
</tr>
<tr>
<td>Photograph Editing software</td>
<td>Enhance photographic data (addition of comments, contrast, etc.)</td>
</tr>
<tr>
<td>File transfer protocol</td>
<td>Transfer of bridge files from van/office unit onto the device</td>
</tr>
</tbody>
</table>

2.2.2 Visionary Scenario

The Visionary Scenario describes the same work effort addressed in the Problem Scenario, with modifications made, based on the identified target technologies. The following
is the visionary scenario that was developed in the Conceptualization Phase. Suitable modifications were made to the scenario during the design and implementation phases.

**Scenario 1: Retrieve & Review**

**Situation:** Joe, a 35-year-old inspector, is sitting in Dunkin Donuts and preparing for the next inspection at Miller’s Run bridge. He has MIA with him.

He is now looking at a summary of Miller’s Run bridge which includes information about the bridge such as the deck rating condition code and the superstructure condition code. He can also retrieve the last two inspection reports of Miller’s Run Bridge from this summary. He tells MIA to give him the superstructure portion of the last inspection report that has worst condition rating. Then he can take a detailed look into what is happening to the superstructure. MIA gives him a list of elements applicable to Miller’s Run Bridge, which is a 2-girder single span bridge. When he is satisfied with his review, he makes a note to remind himself what he needs to do during the inspection. This note is created on his virtual space, which is analogous to a clipboard.

He goes back to the summary report and asks MIA to show him which elements have monitoring systems. MIA shows him the monitored elements which are the left bearing at pier 1 and cracking on beam L4. He reviews the monitoring system note of L4, which includes graphs showing the crack width versus time. He makes a note to check element L4 and to take a photograph of L4.

Joe checks the equipment on the summary report to determine what equipment he will need to use for this inspection. He finds that he needs a 12-ft midstream ladder to inspect this bridge. Again, he makes a note to check its reliability.

Joe looks through his notes again and he feels he is ready to go to the bridge. He leaves Dunkin Donuts and heads to his van.

**Scenario 2: Prepare for inspection**

**Situation:** Joe is in his van on the way to the inspection site. MIA rests in a charging cradle.

Joe tells MIA to make a copy of the last inspection report file including comments, sketches, and pictures. He asks MIA to read him the note he prepared at Dunkin Donuts. In his note, he wants to make a sketch for Girder G1 that is not included in the last inspection so he has to create a sketch template for G1.

Joe listens to the rest of his note, and he is reminded that he needs to set a monitoring system for the crack on the back wall number 1. He asks MIA to set up a monitoring system template where all the information to be monitored is on the template such as crack location and length. He decides everything he will need is available on MIA.

At a red traffic light, Joe looks at a marked map to locate the site. The map is linked to domain pictures of the site that gives him a good idea of how to find it. He arrives at Miller’s Run bridge. He is now ready for the inspection.

**Scenario 3: At Site (Inspecting Channel)**

**Situation:** Joe is now at the Miller’s Run Bridge. He has MIA with him.

He checks the note compiled by MIA and visits the channel. He notices debris in the channel. He thinks this might harm the channel productivity of the bridge so he uses a voice annotation to make verbal comments on channel in the inspection report. He goes to the Maintenance Needs Data Form and updates the form on “Scour-Erosion Control” section that debris under the bridge should be removed. MIA goes to sleep while Joe walks around the
bridge to see the overall condition. After the first pass, he goes to the first element on his to-do list.

**Scenario 4: At Site (Inspecting Element with monitoring system)**  
**Situation:** Joe is now at the bridge.

He asks MIA to give him the list of monitoring elements and one of them is B1, which has movement that should be measured. He locates bearing B1 by looking at the drawings on the clipboard on MIA. Joe goes to B1 and measures the displacement. He inputs the displacement of B1 to the monitoring system file. He measures the temperature with an external device, and the temperature is also recorded into the file.

MIA updates the data and generates the graph of the displacement of B1 over time. This allows Joe to view criticality of the situation. Next, he uses MIA to take digital photographs of B1 and annotates the photograph with information about the element using his push-talk microphone. He brings up the sketch and marks the damage on it. He also draws on the captured digital image for details. He brings up the form and adds comments to the form. He completes the annotation and decides to move to the next element.

**Scenario 5: At Site (Inspecting normal element)**  
**Situation:** Joe is at the bridge, inspecting element L4 which is a floor-beam.

He selects the superstructure and then floor-beam sub item from the last inspection report. MIA displays the last inspection report on floor-beam which shows that floor-beam has crack in the middle of L4 from the last inspection report. He zooms the picture to take a closer look at it and compare it with the current condition. He may also take a look at a 3-D visual of L4 but he decides not to do so. He finds that the crack has not changed significantly from the previous inspection so he indicates "no change" which forwards the comments from the previous inspection into the form for this inspection. He decides not to make the sketch of L4. However, he takes the picture of L4 to update the record and again he annotates the date and any information about the element. MIA compiles and saves the information of L4.

Joe moves to the next element of the bridge. He does the same thing for other elements of the bridge. An element that requires maintenance or replacement will be updated in the Maintenance Needs Data Form.

**Scenario 6: Generate Report**  
**Situation:** Joe is now done with the inspection work. He goes to the van and sits MIA in its cradle.

He tells MIA to generate the report. MIA generates the report with appropriate annotations, and references. Joe makes sure that sketches and references are all linked correctly. MIA also indexes the report, so that it is easily navigated. Joe reviews the report and edits some comment.

Joe is satisfied with the final report. So he saves the report onto MIA. He asks MIA to print the report in paper form to be stored in the district office. MIA also produces a BMS file that can be transmitted to Harrisburg. The report is done and sent to the district office for archiving.
2.3 Conceptual Design

2.3.1 Solution Scenario Selection Criteria

2.3.1.1 Software

The software group was primarily concerned with three objectives for each software package:
- meeting the required features for each data object
- integrating the tools that implement these features
- staying within reasonable limits for hardware requirements

An object oriented design was adopted in the software design, thus ensuring modularity. The different objects were first identified [Table 2.3.1.1], a preliminary search for a suitable software tool for each of these objects was done, and feasibility studies conducted on each of them.

Each software product was tested against the requirements for the data objects, which were derived from the visionary scenario. The tools that met those requirements were compared against each other to find any obvious problems that might result during integration of these tools. Finally, the hardware requirements were evaluated for the tools that satisfied the first two tests.

Table 2.3.1.1 Data objects and their purpose

<table>
<thead>
<tr>
<th>Object</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>To store the information obtained on site</td>
</tr>
<tr>
<td>Sketch</td>
<td>To enable the inspector to make free-hand sketches in the field</td>
</tr>
<tr>
<td>Photo</td>
<td>To enable the inspector to dow-load and view photographs,</td>
</tr>
<tr>
<td></td>
<td>taken with the digital camera</td>
</tr>
<tr>
<td>Annotation</td>
<td>To enable the inspector to enter damage information</td>
</tr>
<tr>
<td>(speech recognition, handwriting recognition, virtual keyboard)</td>
<td></td>
</tr>
<tr>
<td>Monitoring System</td>
<td>To graphically observe variation in parameters associated with</td>
</tr>
<tr>
<td></td>
<td>a bridge component</td>
</tr>
</tbody>
</table>

The database and spreadsheet software were produced by the same company, in addition to others. To resolve integration issues, it was decided to use products from the same company for both these objects. Since all of the products were fairly similar in terms of capabilities and hardware requirements, the main selection criterion was familiarity with the product. This was done to reduce the learning curve associated with using the products during the design and implementation phases.

Only one sketching tool met all of the basic requirements for the sketch object, the hardware requirements, and the integration constraints. Fortunately, the tool is extremely powerful and useful, and meets the feature requirements well.

Two digital photography tools met the basic requirements for the photograph object, the hardware requirements and the integration constraints. There were tradeoffs associated with these two tools. While one of them was very easy to use, it lacked the facility for free hand sketching, which is a key requirement. The other option required a great deal more time and effort to produce, and lacked a crucial "undo" capability.
Interviews with inspectors brought to light that speech recognition would be the most sought after mode of annotation. More than one product satisfied the basic requirements for the speech object, the hardware requirements and the integration constraints. However, some of the speech recognition tools did not support recognition of continuous speech. The final product was selected, owing to its ease of use, its reliability and most importantly, its capacity to be trained for user-defined voice commands, which would then ensure completely hands-free operation at the field.

Although speech recognition was considered to be the chief mode of annotation, the possibility of handwriting recognition was not ruled out. This mode of annotation was considered as a backup, which could be used, if the speech recognition system were to fail on site, for some reason or the other. In addition, it would serve as the chief mode of annotation for those who prefer not to use speech recognition. A third form of input that was considered was the "virtual keyboard." The virtual keyboard would consist of a layout of the alphabets and numerals, where the inspector would choose letters by tapping on them with his stylus.

The handwriting recognition software and the virtual keyboard were selected based on their ease of use, and the ability to integrate easily with the other software tools.

2.3.1.2 Hardware

It was decided to use a 9.6" screen display because it is roughly of the same size as the clipboard. A color display was selected since the bridge inspectors prefer color display screens. A 586 133MHz CPU and 32MB to 64MB RAM was chosen to support the software that would be used on the device. A large capacity hard drive was also a consideration because the software and operating system that were selected, required a significant amount of hard drive storage space. The safety of data was also a consideration. A non-rotating hard drive would not be affected by torque as much as a rotating hard drive. The probability of the bridge inspector accidentally tumbling down the river-bank, causing torque on the hard drive was considered high. If the hard drive were to crash or malfunction, the bridge inspector could lose hours of work. Hence it was decided that wireless communication would be installed for backing up the data from the wearable unit to the van unit. This would reduce the likelihood of the user losing hours of work and valuable time.

Both stylus pen and touch screen were considered for the input device. The stylus pen was preferred over the touch screen because the inspector may be wearing heavy gloves.

Ideally, battery life should be about four hours because the inspectors work for roughly four hours before and after lunch breaks. The inspector can change the batteries during the break and charge the used batteries.

For speech recognition, a throat-mounted microphone or a noise cancelling microphone were considered as candidate technologies. A throat-mounted microphone can directly pick up the user’s voice from his vocal chords which may eliminate more background noise from traffic, running water, and other inspectors, in comparison with a noise-cancelling microphone.

For the camera technology, digital cameras were selected because it is easy to import photographs to the report and is also easy to annotate comments on the pictures. Digital cameras also allow the user to view the pictures right after he takes the photograph.

2.3.2 Product Design Specifications
• Display 9.6” color or monochrome display screen
• Input stylus pen or touch screen, microphone
• Cellular modem/Wireless LAN communication with backup unit to store and retrieve necessary information.
• Performance 586 133MHz processor with 32 – 64 MB RAM
• Speech to text recognition through microphone, text to speech playback through ear piece
• Graphical display of information through sketches, digital photographs, and graphs
• Textual and sound representation of information through stored voice clips and speech to text recognition
• Storage Flash Drive or EIDE Hard Drive (PCMCIA)
• 350 MB to store software and previous inspection reports
• Weight monochrome display weighs approximately ½ pound, color display weighs approximately 1 pound
• Battery Life approximately four hours
• Cost less than $4000 for hardware and software
• Operating Environment varies from 0 °F to 100 °F, humid, may be submersed in water, will be used in unstable terrain

2.3.3 Hardware Architecture

The hardware architecture [Figure 2.1] contains three parts viz. office support, van support, and the wearable computer or mobile support. At the office, a server is to be set up that will have either a physical connection or a cellular modem or PCMCIA modem with the laptops or Fujitsu units that PennDot currently owns, which would enable bridge inspectors to download the necessary data they need from office to the van.

For the van unit, the laptops or the Fujitsu units can be equipped for database storage and back up. The laptops have a 586 CPU, a 2 to 4 GB EIDE hard drive, and 32 MB to 64 MB of RAM. Wireless communication will integrate with the laptops (e.g. spread spectrum receiver), so the van unit can communicate with MIA and a backup of the data can be created from MIA, while at the field. A charging cradle is to be set up for MIA. The charging cradle will allow the bridge inspectors to download and upload the data from MIA to the laptops during breaks between inspections or halfway through a large inspection.

The wearable computer will have a 9.6” inch color or monochrome display unit that will accept input from a magnetic stylus pen or a touch screen. The display will be connected to the main body of MIA via the VGA port. MIA contains a mother-board with a 586 CPU, 32MB to 64MB of RAM, and integrated sound, video, and wireless chip sets. Two flash hard drives (solid state) will be installed inside MIA for software and storage. There will be a RS232/Serial port to connect a digital camera, a microphone socket for the microphone, and a
sound output port for the ear piece. A rechargeable battery that will hold a four hour charge is to be provided.

2.3.4 Software Architecture

The software architecture diagram [Figure 2.2] shows where the various software applications will reside and how the different units connect. All three units, mobile or wearable, base unit, and server, will each have all of the individual software components installed. This is feasible as disk space is more than sufficient on all three units. The mobile or wearable unit connects to the base unit via a wireless spread spectrum connection. The connection between the base unit and server will require a modem.

**SW Architecture**

<table>
<thead>
<tr>
<th>Mobile Unit</th>
<th>Base Unit</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Access</td>
<td>Microsoft Access</td>
<td>Microsoft Access</td>
</tr>
<tr>
<td>Dragon Systems</td>
<td>Dragon Systems</td>
<td>Dragon Systems</td>
</tr>
<tr>
<td>NaturallySpeaking</td>
<td>NaturallySpeaking</td>
<td>NaturallySpeaking</td>
</tr>
<tr>
<td>CICHandwriting</td>
<td>CICHandwriting</td>
<td>CICHandwriting</td>
</tr>
<tr>
<td>RecognitionSystem</td>
<td>RecognitionSystem</td>
<td>RecognitionSystem</td>
</tr>
<tr>
<td>Pen Windows</td>
<td>Pen Windows</td>
<td>Pen Windows</td>
</tr>
<tr>
<td>Macromedia SmartSketch</td>
<td>Macromedia SmartSketch</td>
<td>Macromedia SmartSketch</td>
</tr>
<tr>
<td>Microsoft Graph</td>
<td>Microsoft Graph</td>
<td>Microsoft Graph</td>
</tr>
<tr>
<td>Microsoft PaintShop Pro</td>
<td>Microsoft PaintShop Pro</td>
<td>Microsoft PaintShop Pro</td>
</tr>
<tr>
<td>Microsoft Windows95</td>
<td>Microsoft Windows95</td>
<td>Microsoft Windows95</td>
</tr>
</tbody>
</table>

![Figure 2.2 Software Architecture Diagram](image-url)
2.3.5 Mechanical / Industrial Concepts

Two main concepts were looked at while designing the mechanical housing for the MIA. The first was separation – how many units it was made of. The second was placement – where exactly the main unit should go.

2.3.5.1 Configuration

Four different configurations were explored for the separation aspect. The configuration we eventually settled upon had a central unit with the batteries, processor, and data storage, and a satellite unit containing the display [Figure 2.3]

2.3.5.2 Placement

Five options were explored in placement of the display. The placement we eventually settled on was an attachment to the waist [Figure 2.4]. This was chosen partly because inspectors are already used to wearing tool on their belts, partly because of the work of previous designers, and partly due to the team’s perspectives on the handling of the device during actual use.

Figure 2.3 Product Configuration
Figure 2.4 Product Placement
3. **System Tutorial And Usage Scenario**

A set of key requirements were identified in the conceptualization phase, which served as the basis of arriving at an inspector-friendly user interface, software that provides as many required functionalities as possible, a hardware platform to support the software, and finally the product housing, shape, size and placement.

The User Interface was designed around a concept of encapsulation and simplification. Great care was taken to masterfully ensure the intuitiveness and flow of the system. The project was envisioned to add value to the bridge inspection process and utilize technology to impact and optimize their operations. It is critical to achieve this mission that the system usability is congruent with the capabilities of the users. Keeping that in mind the HCI group spent a lot of time going through the details of how users interact with the system and how information should be captured and presented. Thus 4 main components of the actual interface were identified. These tabs represent actual physical states of a 'Bridge File'. A bridge file consists of anything related to the bridge, including previous reports, current reports, sketches, images, etc. In the future video information and collaborative information may also be stored. The only identification with such a file is the Bridge Identification Code, a 14 digit code assigned by PennDOT.

The next 'state' dependent information is the generic toolbar at the bottom of the screen. These are actions that need to be performed at any given time of interaction. Hence this toolbar can be accessed at any time.

There is a section on the left that allows for navigational properties across whatever information is being dealt with in the work area. For instance when the report tab is highlighted, the navigation window will display all the components of the report that can be accessed, like abutments, superstructure, sub structure etc.

### 3.1. Screen Independent Fields and User interface concepts

The user interface [Figure 3.1] can be divided into four main areas.
The "tab bar" helps the user to navigate to various types of information that an inspector needs to access while they perform the inspection. This information includes current report, previous report, general bridge information such as drawings and maps, photo and sketch album, and monitoring system.

The "tab specific menu" is used in helping the inspector to navigate through various parts of the information corresponding to the active tab bar. For example if the previous report tab bar is active, the tab specific menu will include the items which is in the previous report such as date stamped on the report, superstructure item, deck item, and etc. [Figure 3.2].

The Tab specific Menu Items for each Tab Bar are described [Table 3.1.1].

The table [Table 3.1.1] shows the tab specific menu item for a specific Tab Bar (Note: Each item of tab specific menu item is the top most level of the hierarchy of the corresponding Tab Bar.)

Table 3.1.1 : Tab bar functions

<table>
<thead>
<tr>
<th>TAB BAR</th>
<th>Tab Specific Menu Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Report</td>
<td>Forms of current report e.g. D450-A,B</td>
</tr>
<tr>
<td>Previous Report</td>
<td>Date Stamped on the Report</td>
</tr>
<tr>
<td>Bridge Information</td>
<td>Drawings, History, Map, Traffic Data etc.</td>
</tr>
<tr>
<td>Sketch Template</td>
<td>Sketch template by inspection date</td>
</tr>
<tr>
<td>Monitoring System</td>
<td>Flat list of inspection element</td>
</tr>
<tr>
<td>Album</td>
<td>Album by inspection date and element</td>
</tr>
</tbody>
</table>

The "general action tool bar" is used to perform the functionality of the system such as creating a sketch, making a comment, etc. and to perform some basic operations such as copy, cut and paste.

The explanation for each General Action Tool Bar is described [Table 3.1.2]

Table 3.1.2 shows the functionality of each General Action Tool Bar
Table 3.1.2: General Action Tool Bar function(s)

<table>
<thead>
<tr>
<th>General Action Tool Bar</th>
<th>Function(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Bring up the bridge file window</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy picture, sketch, comment</td>
</tr>
<tr>
<td>Cut</td>
<td>Cut picture, sketch, comment</td>
</tr>
<tr>
<td>Paste</td>
<td>Paste picture, sketch, comment</td>
</tr>
<tr>
<td>Insert</td>
<td>Insert sketch template, sketch, picture, bridge element</td>
</tr>
<tr>
<td>Undo</td>
<td>Undo last action</td>
</tr>
<tr>
<td>Search</td>
<td>Search items corresponding to given keywords and return links to those items</td>
</tr>
<tr>
<td>Backup</td>
<td>Save the report to the van unit</td>
</tr>
<tr>
<td>Print</td>
<td>Print the report of selected item</td>
</tr>
<tr>
<td>Quit</td>
<td>Quit the program</td>
</tr>
</tbody>
</table>

- **Working Area**

The "working area" is the area where the information according to the active tab bar and tab specific menu item is shown. For example, if the *previous report* tab bar and *stringer* tab specific menu item are active, the working area will show the condition information about the stingers from a specific previous report [Figure 3.2]

![Figure 3.2, An example of user interface screen](image)

### 3.2 Summary of Integrated User Interfaces

When the user wants to add a sketch, photo or a monitoring system, the corresponding tool is brought up in the working area, and a few additional buttons and features may be
available to the user, depending upon the tool launched. The interface is therefore modified to accommodate these tools.

3.2.1 Sketch Tool

The sketch User Interface has 3 main buttons in addition to the buttons (icons) given on the sketch tool itself, SmartSketch. [Figure 3.3]. The lowest button (Toolbar) is to show and hide the buttons common to MIA at all times (cut, copy, paste, undo, etc.) The other two buttons and icons are explained below:

- **Clear:**

  The clear button is only active when the open sketch is unlocked and can therefore be edited. The clear button will clear the entire sketch. This action can be undone by pressing the Toolbar button and then the Undo button. A dialog box will appear asking the user if they are sure they wish to clear the sketch. The user can respond by either hitting the ‘Yes’ or ‘Cancel’ button.

- **Close/Done:**

  The Close/Done button’s caption (either Close or Done) depends upon the status of the opened sketch. If the sketch is unlocked, then the caption on the button is ‘Done’, otherwise, the caption is ‘Close’. If the user is finished editing or viewing the sketch, they may close the tool by clicking on the Close/Done button. This will bring up a dialog box asking them if they are sure they wish to quit out of the sketch tool.

  If the file is locked, then the user will have the choice to either close the sketch (given by a ‘Yes’ response), or cancel the close request (given by a ‘Cancel’ response). If the file is unlocked (Done), then the user will have the choice to one, close the sketch and save any changes (given by a ‘Yes’ response), two, close the sketch and do not save any changes (given by a ‘No’ response), or cancel the done request (given by a ‘Cancel’ response).

![Figure 3.3 Sketch tool user interface](image)

3.2.2 Photo Tool
The photo User Interface has seven buttons and a text box. The lowest button is for the toolbar to show and hide buttons like cut, copy, paste, etc. The other six buttons are specific to the photo controller. The photo user interface has been built along the same lines as the sketch tool user interface. The clear and close/done buttons provided on the comment box associated with the photo tool perform the same function as those in the sketch tool.

![Photo Tool User Interface](image)

**Figure 3.4 Photo Tool User Interface**

### 3.2.3 Monitoring System

Upon clicking on the Monitoring System tab, the user is brought to the Monitoring System. Here the format is a little altered from. There are two buttons in addition to the “Toolbar” button located on the screen--“Add Element” and “Add Values”. Still on the left of the screen, the tab-specific menu is a short list of monitored elements. Directly above the list is a label signifying which monitored element is currently being displayed. Directly below the list is the "Add Element" button which is used to add a new monitoring system for an element. Below this button is the "Toolbar" button, that is accessible to the user at all times. To the right of the "Add Element" and "Toolbar" buttons is a table for displaying the variables, dates, and all data associated with the monitored element for the last four inspections. The center of the screen is occupied by a line graph for displaying the data found in the table. The right side of the screen has a comment box for the inclusion of any additional comments to the monitoring system. Finally, directly beneath the comment box and above the table is the "Add Values" button. This button allows the user to enter values for the current date to the monitoring system.

Selecting an element in the tab-specific menu at the right of the screen will make that monitored element active. That is to say, the label above the tab-specific menu, the data in the table and graph, and the text in the comment box will all change to the information stored for that monitored element. In addition to reviewing the information, the user can take three actions to change the data: Adding Comments, Adding Values, and Adding New Elements.
3.2.4 Handwriting

The handwriting recognition software is running in the background at any point of time. In order to use handwriting recognition, the user merely has to click on the text box, and start writing. No special interface has been created to incorporate handwriting recognition.

3.2.5 Speech Recognition

Speech Recognition allows near hands-free operation. The user can annotate damage information and also navigate through the user interface using voice commands. A special speech interface has been created, such that it functions as an overlay to the main user interface. Every action on the main user interface has a corresponding voice command. In effect, the voice commands can be visualized as a talking stylus.

In order to annotate damage information, a voice command brings up the editing window or comment box, which displays the comments spoken by the user, in the form of text. [Figure 3.6] shows the comment box that would appear at the time of annotation. The complete list of voice commands in the speech interface has been provided in the user's guide [Appendix A].

Figure 3.5 Monitoring System User Interface

Figure 3.6 Annotation - Speech Recognition Interface
4. System Implementation

The sections that follow give a detailed description of the individual component design. The first sub-section deals with the user interface design concepts, followed by the object oriented software design. The subsequent section describes the hardware platform that has been developed to support this software, followed by the design concepts involved in fabricating the shape and size of the product. The concluding sub-section enumerates fields of possible improvement to this first prototype for bridge inspections.

4.1 User Interface Design

4.1.1 Design Issues

The User Interface is designed around simplicity and availability of information, and functionality. There are many features that can be envisioned to ensure better availability of these resources. The issues that were pending in all previous phases have been successfully resolved. An interesting enhancement would be the introduction of some intelligent searching that allowed bridge inspectors to find information and mix and match information from different bridge files. As it stands MIA, can integrate information from one bridge file, but multi bridge file collaboration is not an easy task. The interface could possibly allow that at some later stage.

The other issue is that the learning curve and intuitiveness may still not be as trivial as possible. The teams have put in considerable effort to simplify the user interface, but limitations such as screen size, speech reliability and so forth make ease of use challenging.

4.1.2 Example Bridge Inspection Reports

4.1.2.1 Current Format

The current report system is a collection of forms generated by the Pennsylvania Department of Transportation. Each of these forms contains information about a different aspect of the bridge. For example, form D-450A contains information about site data while D-450C contains information about abutment data.

The current form system is linear and does not allow for concurrent access to previous data and integration of multimedia in an automated fashion. This is more a drawback of the system that is archaic and relies on tacit knowledge of the inspector.

One of the primary disadvantages of the current system is that the inspector is not able to easily write a separate comment for each different element of the same type. This is due to the fact that there is only one place that the inspector can make a comment for a particular element type. For example, there is only one place to make a comment about girders, even though the inspector may want to give a separate comment for the girder in span 1 and the girder in span 2.

Another disadvantage of the current paper-based system is that the space provided on the form may be too big or too small. This leads to wastage of space. These two disadvantages make the inspection process inflexible and inefficient.
4.1.2.2 Generated by MIA

The new report, is digital and automatically generated from the forms. The format, in terms of flow of information will mimic their current reports. Hence the system will be able to automatically generate a report that will look like an official PennDOT document, and will be transparent to any non user of the system as to the origination of the report. The advantage however lies in the fact that the inspector is able to manipulate and integrate multimedia objects into the report dynamically. Moreover the database stores all relevant information and the inspector can add information at any point of time, without worrying about constraints like space, organization or prioritization.

The report generated by MIA provides greater flexibility to the inspector in the organization of the report. Each element of the same type will have a separate comment field, allowing the inspector to decide the granularity of the inspection. The inspector can print out the report element by element by using a print command. For example if the selected item is "Girders/Beams", MIA will generate the report of Girders and Beams. [Appendix B] provides an example of the report generated by MIA when the selected item is "Current Report".

4.2 Software Design

An object oriented approach was adopted in the software design, thus ensuring modularity. The different objects were first identified, a preliminary search for a suitable software tool for each of these objects was done, and feasibility studies conducted on each of them. Detailed functionality use cases and object models were created for each object to facilitate implementation. Finally the individual components were integrated to arrive at the final software implementation.

4.2.1 Software Object Architecture

The software architecture diagram [Figure 4.1] illustrates the data flow through the software, as well as the interaction of each component in the subsystem, with each other.

Every action performed by the user on the main interface sends a message to the controller, which decodes the message, launches the respective tool, and also sends data to the tool, to be displayed. Consequently the controller serves as the medium of communication between the main interface and the commercial software tools. Each tool has its own interface, which comes into play, once the corresponding tool is launched.

In addition to the main interface, the speech interface is super-imposed on the main interface, in such a way that there is a one-to-one correspondence between every voice command and every possible action on the main user interface.
Based on the object oriented approach, the software was divided into many modular components such as database, speech, annotation, sketch, photo, and main controller.

4.2.2 Software Tools and API

The software tools that were selected did not have any associated Application Programming Interfaces. Consequently, for the purpose of integrating the tools with each other, certain functions were written which would perform the same actions as those built internally within the tool.

Two development environments were considered:
Microsoft Visual Basic 5.0 and Microsoft Visual C++.

The biggest constraint involved was the short amount of implementation time allotted to the prototype. Owing to this, the most important criteria for selection was the ease of learning the usage of the environments. Several people in the group had worked with one or both of the environments, but none had worked with both. In addition, the functionality available with both environments as well as the ability to control other tools i.e. the sketch and photograph editing software, was also considered. Visual C++ offered more functionality, but Visual Basic was able to control the tools better. A comparison of the two environments along the above-mentioned lines, led to the choice of Visual Basic as the development environment. It was felt that this posed a flatter learning curve, and in addition offers a better interface for controlling the other software tools.
The following sections give the design and detailed functionality for the database, each of the tools (database design, sketch tool, photo tool, monitoring system, speech recognition, handwriting recognition) and the main controller, as it is currently built. The object code in pseudocode format, for each of the above-mentioned tools and the controller is given in Appendix A.

4.2.2.1 Database Functionality

The data stored by the database can be classified into a number of groups. These are enumerated and described below.

- **Administrative data**
  The database stores administrative data in the *authentication* table and in the *worklog* table. The *authentication* table stores user-names and passwords. The *worklog* table keeps track of the time spent on each inspection by each inspector.

- **Bridge and Inspection Data**
  Time independent data about each bridge is stored in the *bridge* table. Each bridge is assigned a unique identification number by the database. Additionally, the controller module assures that each bridge is unique given a combination of *district*, *route*, *segment*, and *offset*. Each bridge can have multiple inspections. Time dependent data about each inspection is stored in the *inspection* table. Each inspection is unique given a bridge identification number and inspection date. The inspection date is formatted as <year><day of year> (i.e., 4/27/98 would be formatted as 1998117).

  Each bridge can also have multiple monitoring systems. Time independent data about each monitoring system is stored in the *monitoringSystem* table. Each monitoring system is assigned a unique identification number by the database and may only be associated with one bridge. Time dependent data (i.e., measured values for a particular inspection) is stored in the *monitoringSystemValues* table. Each record in this table is unique by monitoring system identification number and date.

- **Inspection Form Data**
  Each remaining element of an inspection form is stored in a table that is unique, given a bridge identification number, inspection date, and element identification number. An inspection can contain multiple elements of the same type. The element tables are *abutments*, *appraisals*, *approachRoadway*, *channels*, *culverts*, *deck*, *diaphrams*, *expansionJoints*, *floorbeams*, *girdersAndBeams*, *paintCondition*, *piersAndBents*, *portalsAndBracing*, *signage*, *stringers*, *superstructure*, *trafficSafety*, *trussMembers*, and *wearingSurface*. Six of these tables are “children” of the *superstructure* table: *diaphrams*, *floorbeams*, *girdersAndBeams*, *portalsAndBracing*, *stringers*, and *trussMembers*. They differ only in the form of their element identification numbers.

  Each element identification number consists of a 2-character identifier and a two digit number. Thus, the element identification number for the first abutment would be “AB00.” The child elements of the *superstructure* table have their element identification number appended to their parent superstructure identification number. The complete element identification number for the first stringer of the second superstructure element would be “SP01ST00.”
• **Condition Ratings**

The condition codes used in the form are stored in the `conditionCodes` table. This table is simply a listing of the available condition codes that is used to generate part of the `conditionRatings` table. The `conditionRatings` table is a time dependent table that stores the condition ratings for a given bridge and inspection date. Each record in `conditionRatings` is unique given a bridge identification number, inspection date, condition identification number, and condition name.
Figure 4.2  E-R diagram
Figure 4.3 E-R diagram
• **Multimedia Objects**

Multimedia files are not directly stored in the database. Instead, the database stores the complete pathname which can be used to retrieve the file in the `multimedia` table. Each multimedia object is unique given its filename.

The relationships between these classes of data can be illustrated through an E-R (entity-relationship) diagram [Figures 4.2, 4.3, 4.4]
4.2.2.2 Sketch Tool Functionality

The software design of the sketching capabilities of MIA was simplified by the fact that the sketch tool, SmartSketch, did not have an available API. Therefore, the sketch tool has been left in tact and has been setup such that the user can interact directly with the SmartSketch interface. Additional capabilities or shortcuts were added to keep the functionality consistent with the other software tools. For example, the main toolbar of the interface contains buttons such as Cut, Copy, Paste, Undo, and Print. In order to provide interaction with the sketch tool via the main interface, various Visual Basic modules were written which used control character sequences in order to execute the commands. Some actions required multiple character sequences, but that was the highest level of complexity required within the software design. Other commands accessible through the interface such as closing a file or clearing the screen were designed in the same manner.

Additional Visual Basic modules were designed to facilitate interactions between the sketch tool interface and the main controller. For example, the opening and saving of a sketch, or the opening and closing of the comment box corresponding to the sketch. The opening and saving of a sketch was executed by the use of control character sequences, whereas the opening and closing of the comment boxes was accomplished by the use of code from the main controller since these actions directly involved the database.

Finally, because of the fact that the sketch tool’s interface was left untouched, some error checking had to be incorporated into the design. For example, if the user closes or minimizes the sketch tool for any reason, the Visual Basic code will not be able to execute the sketch commands. For this reason, various checks were placed throughout the code in order to prevent any possible disasters. Therefore, if the user has closed or minimized the tool, a warning box will appear warning the user that closing the tool is not a recommended action. In addition, it automatically restarts the tool and opens the previously opened file, although any previous changes will have been lost.

4.2.2.3 Photo Tool Functionality

The software design of the photo tool of MIA is very similar to that of the sketch tool. The photo tool used is PaintShop Pro, which did not have any API available. Consequently control character sequences were used to implement the actions of the toolbar buttons. Once the photo tool is launched, the user interacts directly with the tool. Activating any of the toolbar buttons, sends control character sequences, which enable the main interface to communicate with the photo tool.

However, this functionality leads to run time errors if the user accidently hits the minimize button or the capslock key. In order to avoid this, an error check has been provided in each of the Visual Basic modules. Therefore, if the user has closed or minimized the photo tool, an error number is internally generated. The error check identifies this number and transfers the control from the photo controller to the main interface. An error message is displayed to the user indicating that closing the tool is not a recommended action. In addition, the tool is automatically restarted and the previously opened file is reopened, although the previously made changes have now been lost.
4.2.2.4 Monitoring System Functionality

The Monitoring System was wholly implemented in Visual Basic. The integratable chart and flex grid object types, available as inclusions in Visual Basic itself, made this possible. The chart object is MSGraph 8.0 which is fully controllable through Visual Basic commands. In addition to the ability to display the data in graphical or table format, the Monitoring System has the functionality to index and add elements as well as add values.

The Monitoring System interacts with the database through a set of subroutines. The subroutines allow the addition of an element, retrieval of monitored elements, and retrieval of all data associated with a particular element. The underlying Visual Basic code for the Monitoring System prompts the user for appropriate information and updates and displays the information. Some error checking has been implemented in the system. In the case of adding values, the system would force the user to enter numerical values. It would also prompt the user for confirmation if the system is about to overwrite existing values for the current date.

4.2.2.5 Speech Recognition Functionality

The speech recognition software design involves two parts - annotation and command mode.

The annotation mode is concerned with addition and editing of comments. Since API was unavailable, the speech recognition tool has been left intact and additional voice commands have been incorporated in the speech tool which opens the edit window of the tool itself. Once this window is opened, the user interacts directly with the speech recognition software. The tool converts speech to text and displays the text on screen. Upon completion, a voice command transfers the current contents in the annotation window onto the main interface display. The text is now displayed at the current location of the cursor.

The speech interface has been built in such a way that it forms an overlay on the main interface. For every action on the main user interface, a voice command has been built within the speech tool. These voice commands can be visualized as a talking stylus equivalent. The speech recognition tool scripting language, which is similar to Visual Basic, has been used to design the individual voice commands. A voice command executes multiple control character sequences to implement the corresponding action on the main user interface. To facilitate ease of use, the voice commands have been built in such a way that speaking the caption on a button activates it, thus avoiding a special user guide for the speech interface.

4.2.2.6 Handwriting Recognition and Virtual Keyboard Functionality

The handwriting recognition software functions as an overlay on the PenWindows interface and is transparent to all applications. Consequently, it was not necessary to develop any special interface for handwriting recognition. The CIC handwriting recognition software that was installed identifies the existence of a text box, and adds a writing pad over it. Consequently, the user merely has to use his stylus to click on the text box, in order to activate the software.

The virtual keyboard software is a component of the PenWindows interface and is transparent to all applications. Every time the software identifies a text box, it inserts a virtual keyboard icon. When the user clicks on the icon, a virtual keyboard is displayed on screen and the information can be entered by clicking on the keys using a stylus.

4.2.2.7 Main User Interface Functionality
The design of the main user interface functionality was based on authentication of the existing information. Consequently the controller has been built in such a way that only selected actions are permissible in any given window. For example, when the previous report tab is highlighted on the tab bar, the 'Cut' toolbar button is greyed out and the action is made inaccessible.

Similarly, if the user is in the sketch template and needs to insert a sketch, the insert button opens an Insert options dialog box. The functionality provided is such that only the insert sketch option is available, thus ensuring that unessential actions are performed by the user, leading to accidental system crash.

The functions have been written in Visual Basic, with adequate error checking at every stage. Warning messages are displayed if the user is about to perform an action that is not recommended, such as cut operation on an entire bridge element.

The complete object model in pseudo code format has been provided in Appendix C.

- **Filing structure and naming conventions**
  The uppermost directory structure contains the MIA database and one directory for each bridge file. Each bridge file directory contains a directory each for sketches, photos, templates, and inspection data associated with the bridge. Each inspection directory contains a directory each for sketches, photographs and templates associated with that inspection.
  The controller module assigns each photograph and sketch a unique filename.
  The naming conventions are as follows (x represents a digit from 0-9):
    - If a photograph or sketch is associated with the bridge only and not a page in the inspection report, the complete pathnames are of the form
      /bridgexxxxx/photo/photo-x.jpg for photographs
      /bridgexxxxx/sketch/sketch-x.ssk for sketches
      /bridgexxxxx/template/template-x.ssk for templates
    - If a photograph or sketch is associated with a page and a specific inspection of the bridge, the complete pathnames are of the form
      /bridgexxxxx/insxxxxxxx/photo/elementID-x.jpg for photographs
      /bridgexxxxx/insxxxxxxx/sketch/elementID-x.ssk for sketches
      /bridgexxxxx/insxxxxxxx/template/elementID-x.ssk for templates
4.2.3 Software Tool Evaluation Studies

4.2.3.1 Database Feasibility Studies

The two main criteria for choosing a database were its ability to interface with the development environment and the speed in which it can be implemented. Since two of the group members had considerable experience with Microsoft Access, this was selected as the best solution. It provides all of the functionality that is required for storing the inspection data and for transferring it between MIA and the office and van computers. In addition, it integrates seamlessly with Visual Basic. Visual Basic provides excellent support for Access data structures.

4.2.3.2 Sketch Tool Evaluation Studies

Initially, the three sketch software tools under consideration were MacroMedia's SmartSketch, DLR-Draw, and MT-Draw. Trial demos of each tool were downloaded, and carefully examined. Both DLR-Draw and MT-Draw were found to not satisfy many of the requirements given in the sketch use cases. For example, neither tool was capable of free-hand sketching. Instead, they only allowed for the drawing of predefined shapes such as circles, squares, lines, etc. Therefore these two tools were removed from consideration. The final tool, MacroMedia's SmartSketch was very user friendly and did not contain an excess of unnecessary tools. The tool allows free-hand sketching in addition to access to predefined shapes. Moreover, the tool has straighten and smooth modes which may be turned on or off. Therefore, if the inspector is working in a tight space and the ability to draw a straight line is hindered, the straightening capabilities of the tool can assist the inspector in obtaining a better sketch.

Another use case for a sketch object is the ability to select various areas of the sketch for editing or moving. SmartSketch supports the typical select mode in which the user stretches a box to include the area they wish to select. In addition, SmartSketch contains what is called a lasso tool. This tool allows the user to free-hand sketch any shape which includes the desired area of selection. It is not restricted to be a circle or square or any other given shape. This is especially nice when the desired selection is a small piece of the sketch. Finally, the tool integrates very well with other Window products. The save format is .ssk, but the user can export the file into a Windows Meta File which can then be opened by any Windows based software tool.

4.2.3.3 Photo editing Tool evaluation studies

A preliminary search for photo editors led to nine products. These were mainly based on search conducted on the internet. The initial selection was based on the details that came with these web sites. The various functionalities that the product claimed to do were compared with the use cases that had been prepared for a photo editor ideally suited for this project. The products selected were LivePix, PictureIt, Digital Camera Suite and PaintShop Pro. A careful study was then conducted on the demo versions of these packages.

LivePix was eliminated because of its complexity. It also did not posses features like creating an album, which belonged to the key functionalities being provided.

Digital Camera Suite provided better facilities for album generation. It has a good catalog system and also has the facility of arranging a picture show. Thus a sequence of pictures can be organised in an album and the player can be called for to show these pictures sequentially. This may in some way serve as a substitute for a video. The other advantage of selecting this is that it has a very small memory requirement. The storage space required for this is just 15MB. The serious setback that it suffers is that it is not very user friendly. A user must be quite well-versed with the tool in order to use it effectively. The most important drawback with this tool, is that it does not allow free hand sketching on the photograph, which would be a valuable asset in the photo tool that is to be provided to the bridge inspectors.
PictureIt was much more user friendly and surpassed the facilities offered by Digital Camera Suite due to its ease of use.

However, the ideal solution was found in PaintShop Pro, which allowed for freehand sketching, annotation of damage information on the photo itself, and the generation of a photo album. In addition, this software is capable of performing basic operations such as cut, copy, paste, rotate, etc. and is compatible with the chosen Windows 95 operating system. The photo editor can read all standard formats for pictures (GIF, JPG, BMP) and can also read from any standard digital camera.

### 4.2.3.4 Speech Recognition Tool Evaluation Studies

Speech Recognition was identified as a possible substitute for written or manually typed out forms. A preliminary survey of the various speech recognition products available in the market recognized few brandname, whose features were compatible with the use cases, identified with the speech object.

The identified products were

- Phillips Speech Magic
- Dragon Systems Naturally Speaking
- Dragon Dictate 3.0
- IBM Via Voice Gold
- IBM Simply Speaking Gold

Dragon Dictate 3.0, a discrete-speech recognition product was first tested. Although speech recognition was reasonably effective, it was essential to provide pauses between words. A similar problem was identified with IBM Simply Speaking Gold, another discrete-speech recognition product. While on site, the bridge inspector would like to speak out his comments naturally, and hence discrete speech was ruled out in favour of continuous speech recognition products.

A preliminary survey of the system requirements for IBM Via Voice Gold (requires additional hard disk space of 125MB) proved beyond doubt that this product cannot be used in the proposed prototype, due to the disk storage limitation imposed by the hardware group, during the conceptual design stage.

Detailed tests were performed using Dragon Systems Naturally Speaking personal version, a continuous-speech recognition product. It is extremely easy to use, and allows the user to speak naturally into the microphone. Some of the notable features are:

- True continuous speech (upto 160 words per minute)
- There is no need to pause between words
- Immediate recognition, words appear on the screen immediately
- Has a large vocabulary, words specific to our task can be added to the vocabulary
- Words can be trained to suit the user's pronunciation
- Words can be orally spelt in the natural way, instead of the military usage (alpha, bravo, etc), as was the case in other products that were tested
- The comments can be edited orally at any time.

On the whole, the product is efficient in speech recognition and is extremely easy to use.

Dragon Systems Naturally Speaking comes in 3 versions-personal, preferred and deluxe. The additional features provided by the deluxe version are:

- Dictation of text into any windows based application
- Provides text to speech conversion
- Provides dictation playback
• Powerful macros can be built to add words to the command list

Although the personal version satisfies many of the use cases for the speech object, the deluxe proved to be extremely advantageous for integration with the prototype. The facility to add voice commands proved beneficial in designing a prototype that saves time and which can be operated using voice commands alone, thus resulting in completely hands-free operation.

**Noise Cancelling Microphone:**

The Dragon Systems Naturally Speaking personal version was tested using the VXI microphone and Andrea Noise Cancelling headset.

For effective speech recognition, it is necessary that the microphone have noise cancelling hardware. The environment usually encountered at the time of inspection was simulated and speech recognition test was carried out amidst sounds of flowing water, thunderstorms, and train whistles.

The Andrea noise cancelling headset was found to be extremely effective in eliminating surrounding noise.

**4.2.3.5 Handwriting Recognition Software Evaluation Studies**

CIC Handwriting Recognition System (HRS) works very well as a continuous handwriting interpreter. Many HRS products only translate single characters at a time, but CIC HRS allows the user to write without interruption. Another notable feature is that CIC HRS stays resident as long as the user is in the Windows95 environment. This means that the program doesn’t have to be loaded every time the user wants to write a command. It also allows the user to have handwriting recognition available for more than just one program window at a time. Additionally, CIC HRS converts handwriting to text no matter where it is written on the screen, as opposed to other programs that only recognize pen strokes made within a specific window.

However, a drawback associated with CIC HRS is that its installation procedure requires manual editing of the Windows95 Registry. Since this is applicable only at the time of installation, the issue was overruled.
4.3 Electronics Design

4.3.1 Electronics Architecture

Throughout the design process we arranged for backup solutions in case our primary solution did not workout. Thus, an ideal hardware architecture and a proposed hardware architecture have been developed.

The ideal hardware architecture consists of three detachable components and a wearable computer engineered to meet a bridge inspector’s on-site needs. The wearable computer houses a 586 class processor, 32 Megabytes of memory and a 1.6 Gigabytes of rotational disk storage. A sound and video chip is integrated on the system motherboard, and a wireless spread spectrum radio card resides in the available PCMCIA slot. Connected to the computing unit is a display, headset and camera.

The display is a handheld Sharp 6.25" TFT screen with a touch screen overlay. This device would connect to the computing unit with a detachable custom cable.

The headset is an Andrea unit with built-in noise canceling hardware for the microphone and a comfortable earpiece for sound reproduction. It is connected via standard 1/8" jacks.

The camera is an Epson PhotoPC 600 with high-resolution imaging (up to 1024 x 768 megapixel) and easy to use features. It is connected via a detachable standard serial cable.

In the van is a laptop with an access point for the wireless MIA connection. This creates a local area network with reliable transfer rates. However, this requires an antenna to be mounted to the inspection vehicle for the optimal reliability. The laptop would act as a storage and backup device for old bridge files as well as an Internet gateway to the office files, as a user attempts to check out a bridge onto his or her system. A charging station for MIA’s batteries would also reside in the van.

For a number of reasons, that are yet to be explained, it was not possible to proceed with the ideal hardware architecture leading to difficult design decisions. The final proposed hardware architecture uses a computing device, identical to that in the ideal model, which incorporates a 6.4” Black & White screen, rather than an independent screen and computing device.

4.3.2 Hardware Subsystems

4.3.2.1 Computer Platform

The hardware architecture platform has been illustrated in [Figure 4.5]

- Cardio Spread-out platform:

Cardio is an integrated circuit board with 586 133MHz CPU, 32MB, IDE controller interface, serial, parallel ports, and PCMCIA connection. Its footprint is about credit card size and 0.25" thick. The cardio platform is a bigger version of cardio circuit board. All the serial, parallel, video and PCMCIA ports are accessible for testing purposes. Additional sound and network cards are plugged into the ISA slots for software testing. A separate LCD display with touch screen integrated is also connected to test. As a result, the purpose of cardio spread out platform is to easily perform any hardware and software testing before we integrate all the hardware into the final housing.

- TIA-P:
TIA-P is the final platform with 586 133MHz CPU, 32MB, IDE controller interface, serial, parallel ports, video and sound chip-set, and PCMCIA slot integrated in a rugged housing. The display with touch screen on it is also integrated within the housing.

It has a switch on the side to turn the machine on or off. Next to the screen, there are four buttons which are resume, contrast -, brightness, contrast +, right mouse. The resume button is to wake up the system when the system is in the sleep mode. In sleep mode, it is possible to save some power so that the battery could be last longer. Contrast -, contrast +, and brightness buttons are used to adjust the contrast and brightness of display and the right mouse button emulates as right button of mouse.

There are two jacks on the case. One is for a speaker and the other one is for a microphone. On the other side of the case, there is a serial port for connecting to the digital camera. The wireless radio card goes into the PCMCIA slot so that data can be backed up to the computer in the van.

**Figure 4.5 Hardware Architecture Platform**

4.3.2.2 Wireless
After considerable effort, the first stages of wireless communications have been completed. These consist of the system from Symbol Technologies utilizing spread spectrum communications. Two factors led to the choice of this system: recommendations from past studies in similar projects, and availability of systems in the laboratory. The use of spread spectrum is also considered to be more robust in terms of signal reception.

The system consists of a wireless PCMCIA card which acts as a node, and an access point which connects the systems and any other adjoining network. It was at the access point that most of the problems occurred. The initial plan was to proceed with a PCMCIA based access point card installed in a laptop to simulate the van unit. It was found, after great effort and consultations with members of previous groups, that the access point card has never functioned properly in the current setup. Consequently, an access point box has been connected to the laptop. Though this is a considerably bulkier option, the access point box has been shown to work with the wireless card in the wearable unit (currently a TIA-P).

It is important to remember in setup that the drivers for the wireless card be installed properly, and that in order to create a local network the IP address and Subnet Mask be set correctly. For the IP address both the wearable and van unit should have all address numbers the same except for the last three digits; all digits in the subnet should be the same.

4.3.2.3 Camera

A wide array of digital cameras have entered the market in the past year, with new models being introduced at a staggering rate. The decision of selecting possible candidates for evaluation thus necessarily had to include key criteria which would make the units most effective in the wearable project. From these candidates a suitable example would have to be decided upon and thoroughly tested to ensure compatibility with the rest of the system.

The first of these was available resolution, which would play a key role in the final decision. This spanned from sub-standard 320x240, through VGA 640x480, up to so called megapixel resolution of greater than 1024x768. By taking similar images with the different candidates, the need for higher resolution could be assessed.

Another consideration was the availability of an on-camera color LCD. Sizes for these screens vary in size up to 2.5 inches, and the functionality they provide was considered to be necessary. These include the ability to view images immediately upon capture, delete unwanted images, and the use of the LCD as a viewfinder for capturing new images.

Form factor was another important consideration. It is believed that the digital cameras will be used in difficult situations, often while wearing gloves. A suitable packaging was required in order that Mechanical and Industrial Design group would be able to incorporate it into the rest of the system with minimal adjustment. Sizes ranged from palm sized mini-cams to video camera size professional units.

The last, but ultimately most limiting, factors were cost and availability. The necessary tradeoff between functionality and its monetary worth needed to be made. Once a judgment was made on this count, a vendor had to be found that not only had the cameras in stock, but was able to work with the purchasing department of Carnegie Mellon University.

Two systems were ultimately purchased from CompUSA, Monroeville. The first was a Minolta DC which had a maximum resolution of 640x480, a 1.8 inch LCD screen, 4MB of internal memory, and a retail cost of $399.99. The second was the Epson PhotoPC 600 which had a maximum resolution of 1024x768, a 2 inch LCD screen, 4 MB of internal memory, and a retail cost of $699.99. Both cameras used jpeg compression to store the images. The Epson PhotoPC 600 has been currently used with MIA prototype.

4.3.2.4 Microphone

Considering the environment that the MIA system will operate in we felt that a throat based microphone system would be the best option. The reasons for this are many. First of
all a throat mike is sealed and thus is less prone to damage from water, dirt, etc. In addition because a throat mike picks up the vibrations of the vocal cords, it is much less susceptible to background noise interference. The microphone that we chose was the Genesys T-05. In addition to the normal advantages of a throat mike, it contains noise canceling hardware to further reduce background noise. The original list price for the microphone was $84.95 however, we discovered a supplier called Amateur Electronic Supply in Milwaukee for the price of $49.95. However when it came down to testing the throat microphone with the Dragon naturally speaking, it was found that it did not work well enough to be reliable. A ear microphone was also tested but that too did not give reliable results. In the current system, a standard noise canceling microphone headset has been used.

4.3.2.5 Display

The design process for the LCD display began with a listing of the options that would be available to support a display. The design was limited to the options that could be supported by the TIA-P and TIA-0. These platforms were based on the Cardio 586 processor platform and the SPC8110 video card, which meant that only a limited subset of the LCD panels on the market would be available. In addition, the TIA-P platform did not contain the full set of display outputs that were on the Cardio, which further limited the choice.

The type of output that can be provided from these platforms is of three types: analog RGB, digital RGB, and VGA (analog). The easiest way to connect a Cardio board to a display would be through digital RGB, but the full set of signals needed for a standard digital screen is not available on the TIA-P. The pre-installed screen on the TIA-P uses a digital RGB signal. However, removing that screen would not be feasible. The second option was to use a screen that would accept an analog RGB input, something that could be provided with the TIA-P/0. SMOS, the maker of the Cardio platform, was contacted for assistance and a search for such a screen was started. The results of the search were less than satisfactory and such a screen could not be found. The final option was to use a screen that would accept a VGA connection. This required a VGA signal, which could be provided with a bit of electrical work inside the TIA-P platforms. However, virtually all of the LCD panels that accepted a VGA input were LCD monitors that required a 120V display and weighed close to 20 pounds. A solution to this problem was found when it was discovered that a company called Allus Technology manufactured a card that would convert a VGA signal to a digital RGB signal that could be used by an LCD screen. The primary drawback to this solution was the price of a display using this technology. The price tag for the card, a 10.4” screen, and the touch overlay would cost over $2000.

A more reasonable solution was found when it was discovered that a company with ties to the wearable computing group had recently integrated a Sharp 6.25” color LCD display with a Cardio platform for a 3-dimensional camera. This company was contacted and technical information about the solution was received. The largest problem with this solution is to obtain the LCD screen itself. The Sharp LCD used is an older model and has been discontinued by Sharp. One possible solution would be to use a similar Sharp LCD screen, however, the results could not be guaranteed and would require further testing.

Current Implementation

The detached monochrome display was chosen because it was an alternative that was already developed and working with the Cardio platform. As it turns out, it provides a display that is similar in size and quality to the TIA-P display, making it an option that does not change the system quality drastically when added. The screen can easily be connected to the display output of the TIA-P and TIA-O systems with some wiring to a connector. The only customization required is an interface board to change standard input voltages to what is required by the display and backlight. The interface board requires input voltages of 5V and
12V DC and converts those voltages to 40V and 1100V respectively. SMOS provides a kit containing the LCD display, backlight, cables, and the two power supplies that use the 40V and 1100V inputs.

The TIA-P and TIA-O both contain all of the signals needed for a standard VGA display. The TIA-P does require an internal expansion connector to allow a connection to those signals, but the TIA-O makes those signals externally available. To connect the monochrome display to the TIA-P, the input cable of the display must be connected to the VGA output of the TIA-P through an expansion connector. The expansion connector is not required for the TIA-O and the display can be connected directly. A 5V and 12V power source is then required for the interface board. The outputs of the board are then connected to the power inputs of the display and backlight.

The pinouts for the TIA-P and TIA-O follow:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>TIA0</th>
<th>TIA0</th>
<th>TIA0</th>
<th>TIA0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PANELDOT</td>
<td>PANELDOT</td>
<td>PANELDOT</td>
<td>PANELDOT</td>
</tr>
<tr>
<td>1</td>
<td>PANELVTIM</td>
<td>PANELVTIM</td>
<td>PANELVTIM</td>
<td>PANELVTIM</td>
</tr>
<tr>
<td>2</td>
<td>PANELHTIM</td>
<td>PANELHTIM</td>
<td>PANELHTIM</td>
<td>PANELHTIM</td>
</tr>
<tr>
<td>3</td>
<td>W AUDIO R</td>
<td>W AUDIO R</td>
<td>W AUDIO R</td>
<td>W AUDIO R</td>
</tr>
<tr>
<td>4</td>
<td>W AUDIO RSHLD</td>
<td>W AUDIO RSHLD</td>
<td>W AUDIO RSHLD</td>
<td>W AUDIO RSHLD</td>
</tr>
<tr>
<td>5</td>
<td>W AUDIO M</td>
<td>W AUDIO M</td>
<td>W AUDIO M</td>
<td>W AUDIO M</td>
</tr>
<tr>
<td>6</td>
<td>W AUDIO MSHLD</td>
<td>W AUDIO MSHLD</td>
<td>W AUDIO MSHLD</td>
<td>W AUDIO MSHLD</td>
</tr>
<tr>
<td>7</td>
<td>PANEL0</td>
<td>PANEL0</td>
<td>PANEL0</td>
<td>PANEL0</td>
</tr>
<tr>
<td>8</td>
<td>PANEL1</td>
<td>PANEL1</td>
<td>PANEL1</td>
<td>PANEL1</td>
</tr>
<tr>
<td>9</td>
<td>PANEL2</td>
<td>PANEL2</td>
<td>PANEL2</td>
<td>PANEL2</td>
</tr>
<tr>
<td>10</td>
<td>W SHLD</td>
<td>W SHLD</td>
<td>W SHLD</td>
<td>W SHLD</td>
</tr>
<tr>
<td>11</td>
<td>PANEL3</td>
<td>PANEL3</td>
<td>PANEL3</td>
<td>PANEL3</td>
</tr>
<tr>
<td>12</td>
<td>PANEL4</td>
<td>PANEL4</td>
<td>PANEL4</td>
<td>PANEL4</td>
</tr>
<tr>
<td>13</td>
<td>JUMP1</td>
<td>JUMP1</td>
<td>JUMP1</td>
<td>JUMP1</td>
</tr>
<tr>
<td>14</td>
<td>PANEL5</td>
<td>PANEL5</td>
<td>PANEL5</td>
<td>PANEL5</td>
</tr>
<tr>
<td>15</td>
<td>PANEL6</td>
<td>PANEL6</td>
<td>PANEL6</td>
<td>PANEL6</td>
</tr>
<tr>
<td>16</td>
<td>PANEL7</td>
<td>PANEL7</td>
<td>PANEL7</td>
<td>PANEL7</td>
</tr>
<tr>
<td>17</td>
<td>FBLUGND</td>
<td>FBLUGND</td>
<td>FBLUGND</td>
<td>FBLUGND</td>
</tr>
<tr>
<td>18</td>
<td>FBLUVIDEO</td>
<td>FBLUVIDEO</td>
<td>FBLUVIDEO</td>
<td>FBLUVIDEO</td>
</tr>
<tr>
<td>19</td>
<td>FGRNGND</td>
<td>FGRNGND</td>
<td>FGRNGND</td>
<td>FGRNGND</td>
</tr>
<tr>
<td>20</td>
<td>FGRNVIDOE</td>
<td>FGRNVIDOE</td>
<td>FGRNVIDOE</td>
<td>FGRNVIDOE</td>
</tr>
<tr>
<td>21</td>
<td>FREDGND</td>
<td>FREDGND</td>
<td>FREDGND</td>
<td>FREDGND</td>
</tr>
<tr>
<td>22</td>
<td>FREDVIDEO</td>
<td>FREDVIDEO</td>
<td>FREDVIDEO</td>
<td>FREDVIDEO</td>
</tr>
<tr>
<td>23</td>
<td>FHSYNC</td>
<td>FHSYNC</td>
<td>FHSYNC</td>
<td>FHSYNC</td>
</tr>
<tr>
<td>24</td>
<td>FVSYNC</td>
<td>FVSYNC</td>
<td>FVSYNC</td>
<td>FVSYNC</td>
</tr>
<tr>
<td>25</td>
<td>FMSDAT</td>
<td>FMSDAT</td>
<td>FMSDAT</td>
<td>FMSDAT</td>
</tr>
<tr>
<td>26</td>
<td>FMSCLK</td>
<td>FMSCLK</td>
<td>FMSCLK</td>
<td>FMSCLK</td>
</tr>
<tr>
<td>27</td>
<td>FKBBDAT</td>
<td>FKBBDAT</td>
<td>FKBBDAT</td>
<td>FKBBDAT</td>
</tr>
<tr>
<td>28</td>
<td>FKBCCLK</td>
<td>FKBCCLK</td>
<td>FKBCCLK</td>
<td>FKBCCLK</td>
</tr>
<tr>
<td>29</td>
<td>FPRND7</td>
<td>FPRND7</td>
<td>FPRND7</td>
<td>FPRND7</td>
</tr>
<tr>
<td>30</td>
<td>FAUDIOLEFT</td>
<td>FAUDIOLEFT</td>
<td>FAUDIOLEFT</td>
<td>FAUDIOLEFT</td>
</tr>
<tr>
<td>31</td>
<td>FPRND6</td>
<td>FPRND6</td>
<td>FPRND6</td>
<td>FPRND6</td>
</tr>
<tr>
<td>32</td>
<td>FAUDIORIGHT</td>
<td>FAUDIORIGHT</td>
<td>FAUDIORIGHT</td>
<td>FAUDIORIGHT</td>
</tr>
<tr>
<td>33</td>
<td>FPRND5</td>
<td>FPRND5</td>
<td>FPRND5</td>
<td>FPRND5</td>
</tr>
</tbody>
</table>
The input signals for the monochrome display and a head mounted display:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>6.4” monochrome display</th>
<th>head mounted display</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FPVTM</td>
<td>GREEN</td>
</tr>
<tr>
<td>3</td>
<td>FPHTM</td>
<td>BLUE</td>
</tr>
<tr>
<td>4</td>
<td>CAMERA DOWN</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FPDOTCLK</td>
<td>VIDEO POWER +7.2V</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>CAMERA POWER +7.2V</td>
</tr>
<tr>
<td>7</td>
<td>LCDVDD</td>
<td>AUDIO UP/LEFT</td>
</tr>
<tr>
<td>8</td>
<td>BLANK</td>
<td>AUDIO LEFT RETURN</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>MICROPHONE DOWN</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>MICROPHONE RETURN</td>
</tr>
<tr>
<td>11</td>
<td>LD0</td>
<td>OUTER SHIELD/BRAID</td>
</tr>
<tr>
<td>12</td>
<td>LD1</td>
<td>RED SHIELD</td>
</tr>
</tbody>
</table>
4.3.2.5 Batteries and Power Supply

With all of the systems being used by MIA there is a significant demand for power in the system. To resolve this we have found 7.2V 3000mAh Lithium-Ion batteries. These are as powerful as can be bought for reasonable cost and small size. The current retail cost was $99.00 at Circuit City in Monroeville, PA. One of advantages of this is that they are standard camcorder batteries and therefore can be fairly cheaply and easily replaced. It is believed that the power demands will only need 2-3 batteries to provide 4 hours worth of active use. The duration was chosen as 4 hours because it was felt that it would not be too much of a problem to have the inspector change batteries halfway through the inspection, i.e. at lunch. The minimizer functions of the hardware have been used, to have the machine go into sleep mode when it isn’t being used by the inspector. This would save battery power and allow for longer use with less battery power. However since the TI-AP was being used, there were restrictions by the form factor to use the batteries to provided by TI-AP. It is hoped that the form factor will be redesigned in the future to allow higher capacity batteries to be used.

4.3.3 Electronic Subsystem Evaluation Studies

When investigating touch screen display technologies, criteria such as size, weight, resolution, power consumption, type of display, such as color, monochrome, and input interface were critical. It was found that color screens are often heavier and consume more power than similar monochrome displays. As well, searches for displays with magnetic pen input interfaces proved to have little merit. This interface was preferred due to its greater ease of use. In that, touch screens are often difficult to use if you cannot lay your hand directly on
the screen; magnetic screens are only activated with a special stylus. The target size for the screen is approximately 9.6 inches, measured diagonally, which should provide ample screen real estate as well as a small enough package to work with and electrically power.

Criteria such as resolution, display, size, weight, and interface were important in comparing digital camera products. Most cameras proved to have an RS 232 serial interface and a display screen ranging from 1.6 inches to 2.5 inches. Cameras could store between 11 and 59 photos, depending on the resolution and size of the internal storage. Some cameras had removable storage devices.

Wireless networking product criteria included range and performance of the signal. Signal varied from 130 feet to 800 feet. Signal performance specifications varied from 30 kbs to 700 kbs, while noting that performance was consistently 500 kbs throughout old field studies performed by EDRC.

Storage was considered as a separate product because of the desire to provide a rugged and redundant solution to the bridge inspectors. Three types of storage were considered: solid-state flash technology, PC card hard disks, and 2.5” hard disks, common to many laptop computers. Reliability was inversely proportionate to the amount of storage a particular device offered; size was an important consideration because of the intense software that the device would need to support. Other criteria included size, weight, power consumption and operating temperature. Most of these criteria proved to be comparable among the different devices.

4.3.3.1 Computer Platform

Cardio spread out platform:

After finishing setting up the platform, which including all the software are installed in the hard drive, it seems the sound card cannot be set it up right. As a result, we cannot test the speech recognition. This problem still cannot be solved because Randy told us that windows 95 always has troubles to set this particular sound card (ESS 1888 chipsets) from previous other projects. However, all other software works smoothly under the development platform.

TIA-P:

After loading every software on to TIA-P, we could get the pictures transferred from digital camera to TIA-P; however, the transfer speed is not as great as with the regular PC. Speech recognition is not that great with TIA-P. It runs pretty slow and sometimes it has hard time to recognize the right word. Every time when we run the audio setup for the speech recognition, it gives us poor results on the microphone. Brian told us that one of reasons is that there is a lot of internal noises from TIA-P. At lease the software could recognize the command pretty well. And all other software and tools run smoothly on the TIA-P but not as fast as the PC because we use the cardio.

4.3.3.2 Wireless

The primary need of the software system for the wireless communications is the use of file transfer between the wearable unit and the van unit. For this purpose the tests which have been carried out have been done so using ftp to pass files of varying megabyte size from the van unit to the wearable and back the other way. Megabit transfers were achieved in the laboratory setting.

The next step is testing in the field, was completed with and without line of sight to the antenna. The recent acquisition of an ethernet hub is essential for this process. The hub connects the access point box to the ethernet card in the laptop (van unit). This creates the wireless connection between the access point (and thus the van unit) and the wearable unit. One of the problems still being addressed is the fact that the hub and the access point box both require AC connections for power. While the inspectors have assured us that AC power will be available in the new vans, we must look for alternatives.

In the field, 500 kbps transfer rates were normal within line of sight. This number ranged from up to 700 kbps at close ranges (< 200 feet), and dropped down to 30 kbps at longer
range (> 900 feet). When line of sight between the wearable and the antenna was lost (behind concrete walls, etc.), transfer rates dropped rapidly. In this situation the signal was often unusable.

4.3.3.3 Camera

Tests were conducted by taking similar images under the same conditions with both cameras at a bridge site. Different resolutions, zooms, and lighting conditions (including flash) were utilized. From these it was found that the resolution of the Epson conveyed much more of the necessary information, and the ease of use (even with gloves) was greater than that of the Minolta.

Back in the lab, it was found that the software used to download images was superior for the Epson. Download times were on the order of 11 seconds for the Epson and 18-20 seconds for the Minolta. From the data gathered it was found that the Epson PhotoPC 600 was the superior product and worth the difference in price. Currently work is being done to incorporate the software provided with the work being done by the SW group.
4.3.3.4 Microphone

Both the throat mounted microphone and earpiece microphone were tested using the speech recognition software and were found to be unacceptable. A standard noise canceling headset microphone with separate earpiece was decided upon, as it provided sufficient sound capture.

4.3.3.5 Display

The solution for the actual prototype is the screen that has been integrated into the TI-AP itself. This provides a 6.4” monochrome display that is sufficient for inputting data. In addition, its relatively small size means that power consumption will be less of an issue compared to a 10.4” screen. That size will also allow a design that is much less cumbersome to wear and use for a long-term basis. In fact, the bridge inspectors have hinted that a one-piece MIA unit would be preferred because it would allow hands-free operation using speech recognition. The detachable screen option is still available in the form of a 6.4” monochrome display that is currently being used for another project.

4.3.3.6 Batteries and Power Supply

Due to the differences in manufacturing for the different TIA-P units reliable battery life data is unavailable; however, we have been assured that using only the TIA-P the batteries should last for the length of an average inspection.
4.4 Mechanical Engineering and Industrial Design

4.4.1 Mechanical Design Concept
The final design has been that of a harness and container for the TI-AP platform on which MIA is based, as well as the peripheral hardware associated with the device. The unifying element was the safety vest the inspectors already wear. The primary mode of use is to cantilever the unit off of the chest or stomach, and a secondary storage on the back has been included for those inspectors desiring an alternate method for using MIA. The whole unit has been designed to be easy to use while providing the maximum possible flexibility.

4.4.2 Support Vest
The harness consists of strapping that has been attached to a typical safety vest worn by a bridge inspector. There is a belt strap that clips together in front and adjusts on both sides of the vest. Vertical straps extend from the front of the belt directly over the shoulders to the back. These straps have D-rings attached to them at regular intervals for the container for MIA to be clipped. A large pouch is sewn to the straps on the back of the vest to provide an alternate storage for MIA. The vest is shown in [Appendix C, Figure 1], the front buckle in [Appendix C, Figure 2], the adjustment straps in [Appendix C, Figure 3], and the back pocket is shown in [Appendix C, Figure 4].

4.4.3 MIA Container
In addition to the vest, we produced a separate container for MIA which could be placed in multiple locations on the vest. In the primary mode of use, it would be clipped to the D-rings on the front of the vest. In the alternate mode of use, it could be stored in the back pouch until it was needed, at which point it could be held in the hand.

The container consists of two rigid plates attached to a hinge, and a large pocket attached to one of the rigid plates. One place is solid, and is placed against the body in the primary mode of use. The hinges have several lock points, allowing the pocket to be pulled away from the body at several angles. The plate against the chest is used to cantilever and support the pocket, such that it can project from the body. The second rigid plate, which is attached to the pocket, has a hole cut in it such that the screen of the TI-AP unit can be viewed when the container is open. The TI-AP unit will be placed in the pocked and secured with an elastic strap. The camera will also be stored in the pocket.

The primary mode of use is to attach the rigid plate to the D-rings on the vest by use of clips, and open and close the container as necessary to use MIA. In addition, the entire container fits in the pouch on the back of the vest, as was shown in [Appendix C, Figure 4]. [Appendix C, Figure 5] will show the container on a table-top. Figure 6 will show MIA being clipped onto the D-rings, and [Appendix C, Figures 7, 8] will show the container open and closed respectively while it is in the primary mode of use. [Appendix C, Figure 9] will show the placement of devices within the pocket.
4.4.4 Additional Components

4.4.4.1 Computer Platform

The computer platform chosen is TIAP, and ICES design which already has its own housing and packaging. Due to convenience, we will be using this design unmodified.

4.4.4.2 Camera

The camera will be housed in the container next to the TIAP unit, and will be accessible through a separate flap. The serial connection to the TIAP will be mounted in the container and left in place throughout use of MIA.

4.4.4.3 Microphone

The microphone and speaker are part of a wire headset that fits easily and comfortably underneath a hard-hat. The point of attachment is a port in TIAP which is exposed through the container pocket, and the wire can be tucked beneath the vest so as to keep it out of the way.

4.4.5 Materials

The choice of vest was simple enough, as it is the safety vest currently employed by bridge inspectors. The straps, buckles, and slider chosen to add to the vest were all based on our experiences with backpacks and climbing gear. We saw our design as having many similarities with these sorts of devices, and wanted to have the same sorts of fabrics and mechanical parts as they did. The canvas for the pockets was chosen for similar reasons, and weatherproofing added to the canvas to mimic the synthetic fabrics used in backpacks. The foam padding was chosen simply for its ease of use in creating cushioning. The wood plates were used because of the ease of manipulating wood. Overall, we wanted to use materials that were easy to manipulate while maintaining the look and feel of other devices for carrying loads.

4.4.6 User Feedback

Our initial design for implementation was a belt based unit. However, once we had presented the idea, in addition to the initial enthusiasm we encountered some suggestions that perhaps it would be better for us to avoid a belt-based solution, as the tool belt was already a crowded area for inspectors. We quickly moved to the vest based solution, and the initial feedback indicated that this was the direction to move in. We do not have any detailed feedback as of yet, but the initial response has been positive.
4.4.7 Future Design Goals

While we are proud of what we have accomplished, we do feel that there are a few areas that we wish we had further explored. First, the attachment of the container to the D-rings is still a bit awkward, and if better connectors can be found, we believe this would help with using MIA immensely. Secondly, we feel that the search for materials to be used on MIA is not complete. The wood panels were a good choice for the prototype, but for the purposes of production it would be better to find another material. In addition, the use of fabrics such as Gore-Tex could prove beneficial to the design of the containers, and should be explored. Finally, the entire harness should be examined for ways to make the entire apparatus more secure and rugged.
4.5 Conclusions

MIA was designed under certain distinct design principles. The ability to gain effective operation in 5 minutes with minimal computer literacy was one of them. The other design philosophy was it should be able to complete and automate the bridge inspection process to a great deal, such that when an inspector departs from a site, he pushes a button and gets a complete report. Technological advances such as multimedia and so forth were incorporated to produce state of the art digital reports that can be stored, transferred updated and viewed with minimal effort. Thus Mia really demonstrated how technology, rapid prototyping principles, and general innovation can add value to process based activities. Bridge inspection is ideal, since it is a mapped out process, but still requires a lot of tacit knowledge.

From the design and mechanical standpoint the wearable design focused primarily on the need for an inspector to carry the MIA unit along with him to various parts of a bridge inspection site. At least one inspector gave us feedback that cranes are being used more and more for inspection. These provide a powered, mobile bucket for the inspectors to stand in while they are examining areas. It also has some desk-type space available for the inspector. For this case, the MIA would not need a separate vest or carrying harness. Although, the carrying compartment could still be folded out for a better viewing angle that available without it. Nevertheless, the inspectors say that there will always exist some portion of the inspection process which must be completed on foot. For this, we have the vest-harness that we have designed.

4.5.1 Summary of key design issues

The design was synthesized around the concept of ease of use and ease of learning. The designs that were chosen always tried to answer these two key design principles. Some of the key design decisions are summarized below.

First and foremost Visual Basic was chosen as the tool of choice for implementing the actual user interface. This choice was based on the fact that the underlying operating system was Windows95, and the tools that were being used had VB as the most flexible integrating environment. It was also decided to provide a three tier interface The top tier, or bracket of screen will contain a set of tabs that are functional dependent. These tabs, (as discussed and demonstrated by screen shots earlier) will be reflective of state of the MIA. There will be another tier of commonly called and needed functions as buttons on the lower part of the screen. The left of the screen will allow navigational capability through the state dictated by the tab. This three tier interface accounts for all user needs. Independent issues such as searching, albums and templates were explored in good details. Resolutions of these issues are mentioned in earlier sections of the report in detail.

The primary concerns of the mechanical engineering and industrial design group were to provide a way for a person to carry or tote the TAIP main component of the MIA. This carrying apparatus had to be unobtrusive, flexible in wearing positioning, weatherproof, and facilitate hands-free operation. Key parts of making the device easy to use are allowing it to be used with gloves, limiting the wires from place to place and keeping them from tangling, and placement of the stylus with the screen for easy access. The front mounted design makes locating the screen so that it is easily accessible with hands a challenge. Waterproofing is important for the device. Finally, the group believes that protection for the screen both in use and being stored is a key design point.
### 4.5.2 Requirement Feature Table

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Technology Options</th>
<th>Technology Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Must be able to annotate damage information with sketches</td>
<td>• Macromedia SmartSketch&lt;br&gt;• ART smART Scribble&lt;br&gt;• CIC QuickNotes&lt;br&gt;• DLR Draw</td>
<td>SmartSketch&lt;br&gt;Future Wave Software Inc.&lt;br&gt;8305 Vickers Street, Suite 200&lt;br&gt;San Diego, CA 92111-2111</td>
</tr>
<tr>
<td>• Must have access to previous inspection forms on site</td>
<td>• Microsoft Access 97&lt;br&gt;• Claris Filemaker Pro&lt;br&gt;• Microsoft FoxPro</td>
<td>Microsoft Access 97&lt;br&gt;<a href="http://www.microsoft.com/">http://www.microsoft.com/</a></td>
</tr>
<tr>
<td>• Ability to get a monitoring system, to graphically display deterioration of elements</td>
<td>• Microsoft Excel&lt;br&gt;• Microsoft Graph 8.0</td>
<td>Microsoft Graph 8.0&lt;br&gt;<a href="http://www.microsoft.com/">http://www.microsoft.com/</a></td>
</tr>
<tr>
<td>• Must be able to annotate damage information with photos</td>
<td>• Microsoft PictureIt&lt;br&gt;• Paint Shop Pro&lt;br&gt;• Digital Camera Suite</td>
<td>Paint Shop Pro&lt;br&gt;Jasc Software Inc.&lt;br&gt;11011 Smetana Road&lt;br&gt;Minnetonka, MN55343&lt;br&gt;<a href="http://www.jasc.com/">http://www.jasc.com/</a></td>
</tr>
<tr>
<td>• Must be able to annotate damage information with words</td>
<td>• Dragon Systems Naturally Speaking&lt;br&gt;• Dragon Dictate 3.0&lt;br&gt;• IBM Simply Speaking Gold&lt;br&gt;• IBM Via Voice Gold</td>
<td>Dragon Naturally Speaking Deluxe&lt;br&gt;Dragon Systems Inc.&lt;br&gt;320 Nevada Street&lt;br&gt;Newton, MA 02160&lt;br&gt;<a href="http://www.naturalspeech.com/">http://www.naturalspeech.com/</a></td>
</tr>
<tr>
<td>• Speech recognition is the first choice for annotation with words, but backup is needed if speech recognition is not reliable</td>
<td>• CIC Handwriting Recognition Software&lt;br&gt;• ART smARTwriter</td>
<td>CIC Handwriting Recognition Software&lt;br&gt;<a href="http://www.cic.com/">http://www.cic.com/</a></td>
</tr>
</tbody>
</table>
4.5.3 Lessons Learned

A major lesson learned was that user feedback is critical. We were fortunate that the users liked what we did, but if we had the opportunity to get feedback during the development, we could have perfected a lot of things. Another lesson learnt was that rapid prototyping involves more than just some screens.

From a cycle standpoint, it was clear that good HCI principles feed into good designs, feed into good implementation. The phases are highly dependent on each other, and a good job in a preceding phase eases the next one. Thus conceptual, detailed and implementation phases were extremely co-dependent, and that proves the model we used for the project.

From the software design point of view, it is extremely essential to determine well in advance, the availability of Application Programming Interfaces for the tools that are being selected. Another important factor that must be kept in mind, is the need to have alternate designs in hand, so that failure of one design at the time of implementation and integration does not have adverse effects on the prototype development as a whole.

From the mechanical and industrial design point of view, we appreciated the convenience and even necessity of mocking up a design before actually building it. We learned to gain a better spatial perspective of concepts by actually laying out all proposed physical component of the system before-hand. There were several specific lessons about our harness. For example, heavy parts of the design should be well fastened and well supported. Our harness holds up the MIA on the front with two discrete-positioning hinges which are rated for up to 50lbs. of weight. Another lesson learned was that belt designs for wearable computers are not good when the tool belt is full and the screen must be easily viewable.

4.6 Design Evaluation

The final product demonstration was attended by many bridge inspectors and valuable feedback was received from the software design point of view.

The bridge inspectors felt that on the whole, the software that had been developed would be extremely easy to use, although they did not have prior experience with the use of computers. However, they felt that it would be advantageous if all existing sketch templates could be viewed, instead of the current implementation where templates associated with a particular bridge alone could be viewed at any point of time. The modification involves a change to a database call, which can be easily implemented.

An additional requirement placed by the inspectors was the need to incorporate a maintenance form along with the inspection reports.
5. Project Management

5.1 Implementation Phase Results

Due to time constraints and the fact the TI-A0 never arrived in time we decided to use the TI-AP. This platform proved to be relatively easy to use but had some minor problems. One of them was simply a manufacturing problem that caused loose connections in the TI-AP. This problem should be eliminated in further production runs. The other problem was with the sound chip being inconstant in its performance with Dragon Naturally speaking. This problem was mainly due to Electro-Magnetic interference and the chip should be better shielded in future manufacturing runs. Overall the TI-AP proved to be a stable platform and ran quickly enough when the TI-AP was equipped with the 586 133MHz processor.

5.1.1 Task Dependency Chart

Mechanical and Industrial Design
Our primary dependency was the inspectors needs and desires. The inspectors are our clients and ultimately we altered all decisions to meet the statements they made during our meetings. Our hope is that the final solution we produced will alleviate their jobs of inspecting a bridge. The possibilities that we explored were narrowed down by the feedback that we received from the inspectors.

Our secondary dependency came from within the classroom. The main group MEI was dependent upon was the hardware group. In the beginning, of this phase our plans were to develop a housing for the color
screen. MEI was dependent on the specifications from the Hardware group. Once these specifications came in our task was changed to concentrate on the harness for the TIAP and the camera.

This part of our task of developing a harness for the MIA unit was relatively quick since it was given to us late in the phase. The task we completed was dependent on the movements and environment of the inspectors. We had to consider the gear the inspectors already wore and also add in adjustments for size differences. In addition we had to design around the weather conditions and the outdoors. Our goal was to create a rugged harness that was comfortable for the inspector to wear.

5.1.3 Summary of Log Book Hours

*Compilation of Log Book Data for Design/Implementation Phase*

*Total Person-Hours:*

Please note that the following data reflects the diligence of group members in posting accurate information to the web site. Entries were placed in best fit categories by key words.

<table>
<thead>
<tr>
<th>Group</th>
<th>H</th>
<th>S</th>
<th>M</th>
<th>E</th>
<th>I</th>
<th>H</th>
<th>C</th>
<th>I</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meetings</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>Integration</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Testing</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Present</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
5.2 Ranked Issue List

There are very few remaining issues left to be resolved but these are three that we can see being implemented in the future.

- Shield the sound chip to improve sound performance.
- Replace Monochrome screen with a Color Screen.
- Increase form factor size for battery pack to allow higher capacity batteries to be used.

5.3 Suggestions for Improving the Class

From the hardware point of view, the main suggestion that would help us provide a better system would be to have more options for the motherboard and to have them be available in house. The problem that we had with the TIA-0 was that we could never get an example in house to test it. It would also have been better to have multiple configurations of the system so that we may select configuration that best fits the needs of the system being designed. Overall this is the issue that would best help in creating a well designed system.
Appendix A
Software Tools Object Model

This appendix gives a detailed description of the object model of the various software tool designs, in pseudocode form.

- **Sketch Object Architecture**

  **Attributes:**
  - FileName
  - saveFileName
  - FileLocked [yes | no]

  **Methods:**
  - **initSketch()**
    ```
    Inputs: None
    Outputs: None
    Source: Sketch Object
    Function: Initializes all variables and buttons.
              Starts SmartSketch.
    ```
  - **setSketchButtons(Locked As Integer)**
    ```
    Inputs: None
    Outputs: None
    Source: Sketch Object
    Function: Enables the appropriate buttons on the interface based on the locked status of the sketch.
    ```
  - **closeComment()**
    ```
    Inputs: None
    Outputs: None
    Source: Sketch Object
    Function: Closes the comment box associated with the currently opened sketch.
    ```
  - **openComment()**
    ```
    Inputs: None
    Outputs: None
    Source: Sketch Object
    Function: Opens the comment box associated with the currently opened sketch.
    ```
• closeSketch()
  
  Inputs: None
  Outputs: None
  Source: Sketch Object
  Function: Closes the currently opened sketch.
            Gives the user the option of going on with the operation or canceling.
            Calls closeComment to close the associated comment box.
            Changes the focus back to the main MIA interface.

• doneSketch()
  
  Inputs: None
  Outputs: None
  Source: Sketch Object
  Function: Closes the currently opened sketch.
            Gives the user the option of closing and saving the sketch, closing and
            not saving the sketch, or canceling the operation.
            Calls closeComment to close the associated comment box.
            Changes the focus back to the main MIA interface.

• clearSketch()
  
  Inputs: None
  Outputs: None
  Source: Sketch Object
  Function: Clears the current sketch.
            Gives the user the option of going on with the operation, or canceling
            the request.

• copySketch()
  
  Inputs: None
  Outputs: None
  Source: Controller Object
  Function: Copies the currently selected item and puts it on the clipboard.

• cutSketch()
  
  Inputs: None
  Outputs: None
  Source: Controller Object
  Function: Cuts currently selected item from the sketch and puts it on clipboard.

• pasteSketch()
  
  Inputs: None
  Outputs: None
  Source: Controller Object
  Function: Pastes the item currently on the clipboard onto the sketch.

• printSketch()
  
  Inputs: None
  Outputs: None
• **undoSketch()**
  
  ```
  { 
    Inputs: None
    Outputs: None
    Source: Controller Object
    Function: Undo the previous action within the sketch tool.
  }
  ```

• **openSketchFile(FileName, saveFilename, FileLocked As Boolean)**
  
  ```
  { 
    Inputs: None
    Outputs: None
    Source: Controller Object
    Function: Opens the file given by filename and saves it to the filename 
    savefilename if it is not null. 
    If it is null, then the file is saved to filename. 
    Calls setSketchButtons to set the buttons according to the locked status.
  }
  ```

• **Photo Object Architecture**

  • **closeComment()**
    
    ```
    { 
      Inputs: None
      Outputs: None
      Function: Attaches the comment that is associated with the current photograph 
        to the database so that the picture and comment are stored at the same 
        place
    }
    ```

  • **openComment()**
    
    ```
    { 
      Inputs: None
      Outputs: None
      Function: Puts the text attached to the current opened photograph in the text 
        box in the photo tool. If it is a new photo then the text box is cleared.
    }
    ```

  • **Download()**
    
    ```
    { 
      Inputs: FileName
      Outputs: None
      Function: If the photo download software (Image Expert) is not open then it first 
        opens it up. Next it opens up the camera for viewing the photographs
    }
    ```

  • **openFile()**
    
    ```
    { 
      Inputs: FileName, Locked (True if it is from a previous report, False 
        otherwise)
      Outputs: None
    }
    ```
Function: If the photo editing software (Paint Shop Pro) is not open then it opens it up. Next it opens the file called for and also gives a call to openComment().

• cutPhoto()
  {
    Inputs: None
    Outputs: None
    Function: Cuts selected portion of the photograph (if any) and puts it in the clipboard
  }

• copyPhoto()
  {
    Inputs: None
    Outputs: None
    Function: Copy selected portion of the photograph (if any) and puts it in the clipboard
  }

• pastePhoto()
  {
    Inputs: None
    Outputs: None
    Function: Pastes the current clipboard contents into the picture
  }

• undoPhoto()
  {
    Inputs: None
    Outputs: None
    Function: Undo the last operation
  }

• printPhoto()
  {
    Inputs: None
    Outputs: None
    Functions: Prints the photo if a printer is connected
  }

• brightness()
  {
    Inputs: None
    Outputs: None
    Function: Brings up the brightness/hue control tool so that the user can change them if needed. It also gives a preview of the change done
  }

• pencil()
  {
    Inputs: None
    Outputs: None
    Function: Brings up the sketching pencil which allows the user to do
sketches on the photograph.

• **ruler()**

  
  Inputs: None  
  Outputs: None  
  Function: Puts a grid on the picture and also attaches a relative scale on the x and y axes.

• **select()**

  Inputs: None  
  Outputs: None  
  Function: Allows the user to select a particular portion of the picture.

• **zoom()**

  Inputs: None  
  Outputs: None  
  Function: After using this the user may click any part of the photograph and that portion will be zoomed in. To zoom out, the button corresponding to this function must be clicked again.

• **closedone()**

  Inputs: FileName  
  Outputs: None  
  Function: Depending on the situation this does various jobs. The decision to do a job is decided by the caption/name of the button in the interface that corresponds to this function.

  • If the button is named 'Download':
    It means that it is at a stage where photo download software is displaying all the pictures in the camera. If this function is initiated at this point it asks whether the inspector has selected a particular picture. If 'yes' then it starts downloading the picture from the camera and after that saves the picture in the filename that was supplied. It also changes the name of the button, it is associated, to 'Annotate'. If the user has not selected any picture then no action is taken. There is also a 'cancel' option which can be used when the user doesn’t want to download any photo. This will take it back to its main interface

  • If the button is named 'Annotate':
    The photo download root is closed down and a function call is given to openFile() to open the photo editing tool for annotating the photo.
• If the button is named 'done':
  This happens when the photo editing tool is open and the photo being displayed
  is from the current report. If this function is initiated at this point then it will ask the
  user whether the changes need to be saved. If 'yes' then the changes are saved or else
  they are not saved. In either case the photo editing tool is closed along with the
  photo.

• If the button is named 'close':
  This happens when the photo editing tool is open and the photo being displayed
  is from a previous report and hence 'locked'. If the function is initiated now, it closes
  the photo and the photo editor without saving any changes.

• Monitoring System Object Architecture

  Methods:
  • done_Click()  //effectively AddElement
    {
      Inputs: None
      Outputs: None
      Source: Monitoring System
      Function: Prompts for user input.
                Adds new monitored element to database.
                Updates the list of monitored elements.
                Calls Msdata_Update.
    }
  • Form_Activate()  //effectively AddElement_init
    {
      Inputs: None
      Outputs: None
      Source: Monitoring System
      Function: Initialize AddElement’s text boxes.
    }
  • MSdata_Update()
    {
      Inputs: None
      Outputs: None
      Source: Monitoring System
      Function: Queries database.
                Updates chart labels and data.
                Updates Table labels and data.
                Sets comment box.
    }
  • Form_Activate  //effectively AddValues_init
    {
      Inputs: None
      Outputs: None
      Source: Monitoring System
      Function: Initialize combo box for variable names.
    }
  • done_Click()  //effectively AddValues_close
    {
      Inputs: None
    }
Outputs: None
Source: Monitoring System
Function: Update database with user entry.
Clear table.
Call Msdata_updata()
}

**General Toolbar Functionality**

**Global Variables**

- **Public rsCopied As Recordset**
  - contains a recordset that describes the last bridge component that was copied

- **Public rsFBCopied As Recordset**
  - contains a recordset that describes the last floorbeams copied, if a superstructure component was copied

- **Public rsGBCopied As Recordset**
  - contains a recordset that describes the last girders/beams copied, if a superstructure component was copied

- **Public rsDICopied As Recordset**
  - contains a recordset that describes the last diaphragms copied, if a superstructure component was copied

- **Public rsPBCopied As Recordset**
  - contains a recordset that describes the last portals/bracing copied, if a superstructure component was copied

- **Public rsSTCopied As Recordset**
  - contains a recordset that describes the last stringers copied, if a superstructure component was copied

- **Public rsTMCopied As Recordset**
  - contains a recordset that describes the last truss members copied, if a superstructure component was copied

- **Public lastCopy As String**
  - what object was last copied, “element”, “sketch”, “photo”, “in sketch”, “in photo”, or “text”

- **Public copyTable As String**
  - if an element was copied, the database table from which it was copied

- **Public lastAction As String**
  - the last action completed, “cut” or “copy”

- **Public rsCCSearch As Recordset**
  - a recordset containing the results from the last condition codes search

- **Public CommentSearch()**
  - an expandable array containing the results from the last search by comments

- **Public Sub searchByConditionCode(criteria As String, value As Long)**
  - Input
    - criteria  //“<”, “>”, or “=” [any comparison operator is legal]
    - value //value to use when comparing
  - function
    - rsCCSearch is set to the recordset that is returned by the database and matches the criteria and value.
  - End Sub
• **Public Sub searchByComment(comment As String, elementID As String)**

    **Input**
    - `comment` //comment string to search database with
    - `elementID` //identification number of current element shown in working area

    **function**
    CommentSearch is set to an array of values that is returned by the database.
    CommentSearch(0, i) is the inspection date for the element.
    CommentSearch(1, i) is the element identification number.
    CommentSearch(2, i) is the descriptive name for the element.

    **End Sub**

• **Public Sub insertElement(oldElementID As String)**

    **Input**
    - `oldElementID` identification number of the current element in the working area

    **function**
    inserts another element of the same type as oldElementID into the database,
    if the element has child elements, calls `insertChildElements`

    **End Sub**

• **Public Sub insertChildElements(parentID As String, newParentID As String, table As String)**

    **Input**
    - `parentID` identification number of the current element in the working area
    - `newParentID` identification number of the element inserted into the database by `insertElement`
    - `table` the database table to search for child elements of parentID to insert as child elements of newParentID

    **function**
    for each element in table that is a child of parentID, inserts a new child element in the same table as a child of newParentID

    **End Sub**

• **Public Sub copyElement(inspectionID As Date, elementID As String)**

    **Input**
    - `inspectionID` the identification number of the inspection to copy from
    - `elementID` the identification number of the element to copy

    **function**
    sets lastCopy = “element”
    sets lastAction = “copy”
    sets rsCopied to the records returned by the database that match inspectionID and elementID
    if elementID is an element that has child elements, the rs*Copied recordsets are set to the child elements of elementID

    **End Sub**

• **Public Sub copyObject(FileName As String)**

    **Input**
    - `FileName` the filename of the multimedia object to be copied

    **function**
    sets lastAction = “copy”
    sets rsCopied to the record returned by the database that matches FileName

    **End Sub**
• **Public Sub pasteElement()**

  function  
  if lastAction = “copy”, inserts the contents of rsCopied into the database with the current inspectionID and bridgeID  
  calls *pasteChildren* if the last copied element has children  

  End Sub

• **Public Sub pasteChildren(parentID As String, rsChildrenCopied As Recordset)**

  input  
  parentID identification number of the parent element that was just pasted into the database  
  rsChildrenCopied recordset containing the copies of parentID’s children  

  function  
  inserts the contents of rsChildrenCopied into the database with the current inspectionID and bridgeID  

  End Sub

• **Public Sub pasteObject()**

  function  
  if lastAction = “copy” and (lastCopy = “sketch” or lastCopy = “photo”), inserts the contents of rsCopied into the database  

  End Sub

• **Public Sub cutObject(FileName As String)**

  input  
  FileName name of the multimedia file to be deleted  

  function  
  removes the file from the database and from MIA  
  sets lastAction = “cut”  

  End Sub

• **Public Sub cutElement(elementID As String)**

  input  
  elementID identification number of the element to be deleted  

  function  
  sets lastAction = “cut”  
  removes the element from the database  
  calls *cutChildren* if the element has child elements  

  End Sub

• **Public Sub cutChildren(parentID As String, table As String)**

  input  
  parentID identification number of the parent of this child element to be deleted  
  table database table from which to delete this child  

  function  
  removes the child elements of parentID from table  

  End Sub
### Database Functionality

#### Global Variables

- **Public dbMIA As Database**
  - the name of the database currently residing on MIA
- **Public Const MIAlocation As String = "D:\bridge\integrated\MIA.mdb"**
  - string describing the location of the database currently residing on MIA
- **Public Const DirectoryPath As String = "D:\bridge\"**
  - path of directory where bridge file structure is stored on MIA
- **Public gBridgeID As Integer**
  - identification number of the bridge currently being inspected
- **Public gInspectionID As Date**
  - date when the current inspection for gBridgeID was started

#### Public Sub openMIADatabase()

**function**
- sets dbMIA to the database located on MIA

#### Public Sub findBridgeIDs(district As String, route As String, segment As String, offset As String, rsBridge As Recordset)

**input**
- district: district number to search for
- route: route number to search for
- segment: segment number to search for
- offset: offset number to search for

**output**
- rsBridge: recordset that is returned by the database and contains records for all bridges that match the search strings

**function**
- queries that database for all bridges that match district, route, segment, and/or offset (any or all may be left blank)

#### Public Sub createNewBridgeID(district As String, route As String, segment As String, offset As String, bridgeID As Integer)

**input**
- district: district number of new bridge
- route: route number of new bridge
- segment: segment number of new bridge
- offset: offset number of new bridge

**output**
- bridgeID: identification number assigned to new bridge, -1 if creating the new bridge fails

**function**
- inserts a new bridge with district, route, segment, and offset numbers into the MIA database
- will fail if the combination of district, route, segment, and offset is non-unique

#### Public Sub createNewReport()

**function**
- creates a new bridge report for gBridgeID
- inserts a new inspection with gBridgeID and gInspectionID
- calls populateDB for each element table in the database
- calls populateConditionCodes
- calls createDirectoryStructures
• Public Sub populateDB(table As String, elementID As String)
  input  table  table to populate with a new element
         elementID  identification number to assign to the new element
  function  inserts a new element with elementID, gBridgeID, and gInspectionID into table
  End Sub

• Public Sub populateConditionCodes()
  function  for each record in dbMIA.conditionCodes, inserts a new record with the same conditionID, name, and associated element as conditionCodes and with bridgeID = gBridgeID and inspectionID = gInspectionID into dbMIA.conditionRatings
  End Sub

• Public Sub createFromExistingReport()
  function  copies the hierarchical structure of the most recent inspection of gBridgeID
            creates a new inspection for gBridgeID with gInspectionID
            calls copyFromLastReport to copy the structure of the element tables with the new inspection identification number
            calls populateConditionCodes
            calls createDirectoryStructures
  End Sub

• Public Sub copyFromLastReport(table As String, lastInspection As Date)
  input  table  database table to copy elements from the last inspection from
         lastInspection  the last inspection for gBridgeID
  function  copies all elements in table with gBridgeID and inspectionID = lastInspection and inserts them back into the table with inspectionID = gInspectionID
  End Sub

• Public Sub createDirectoryStructures(dirType As String)
  input  dirType  “new” if the bridge is a new bridge in the database, “existing” if the bridge is already in the database
  function  creates the directory structure needed by the database to find multimedia files on MIA
  End Sub

• Public Function authenticate(username As String, password As String)
  As String
  input  username  name of user, entered by user
         password  password entered by user
  function  if the username is not found in the database, returns a string indicating this
            if the password does not match the username in the database, returns a string indicating this
            if the password and username matches, returns “Authenticated.”
  End Function
• Public Sub getAllFilenames(bridgeID As Integer, filenames As Recordset)
input  bridgeID     bridge identification number to get
        multimedia files for
output  filenames recordset containing filenames returned by
        the database
function sets filenames equal to all of the multimedia files which are
        associated with bridgeID in the database
End Sub

• Public Function getFromDB(tag As String, Optional inspectionID As Date, Optional elementID As String) As String
input  tag           a tag which indicates the table name and
        field name of the item to retrieve from the
        database. The tag should be formatted as
        <tablename>!<fieldname>
        inspectionID [opt] identification number for this page, if it
        exists (it will not if the page is the bridge
        page)
        elementID [opt] element identification number for this page,
        if it exists (it will not if the page is the
        bridge or inspection page)
function returns the text contained in <tablename>|<fieldname> for
        gBridgeID, inspectionID, and elementID
End Function

• Public Sub saveToDB(value As Variant, tag As String, Optional elementID As String)
input  value        value to save into the database
        tag           tag which indicates the tablename and field
        name that will received the value in the
        database. The tag should be formatted as
        <tablename>!<fieldname>
        elementID [opt] identification number for the element
        contained on this page, if it exists (it will not for
        bridge or inspection pages)
function saves value into the database in <tablename>!<fieldname> for
        gBridgeID, glInspectionID, and elementID
End Sub

• Public Function getCCValue(tag As String, inspectionID As Date) As String
input  tag           tag which indicates which condition code
        value should be returned. The tag should be
        formatted as <conditionID>!<name>
        inspectionID identification number associated with the
        condition code value to be returned
function returns the value of the condition code for gBridgeID,
        inspectionID, conditionID, and name
End Function
• **Public Sub saveCCValue(tag As String, value As Long)**
  
  **input**
  
  tag tag which indicates where to save value in the 
  conditionRatings table. The tag should be formatted 
  as `<conditionID>!<name>`
  
  value value to store in the conditionRatings table.
  
  **function**
  
  saves value into the database for gBridgeID, gInspectionID, 
  conditionID, and name.
  
  End Sub

• **Public Sub getTextBoxes(tag As String, Optional inspection As Date, 
  Optional elementID As String)**
  
  **input**
  
  tag tag which indicates the structure of the page to query 
  the database for. The tag should be formatted as 
  `<name of textbox>-<number of 
  textboxes>!<condition code textbox name>-<number 
  of condition code text boxes>`
  
  **function**
  
  calls `getFromDB` to get values from the database for the textboxes. 
  Inserts the value returned from `getFromDB` into the textbox. 
  calls `getCCValue` to get values from the database for the condition 
  code textboxes. Inserts the value returned into the condition code 
  textbox.
  
  End Sub

• **Public Sub getCheckBoxes(inspection As Date, elementID As String)**
  
  **input**
  
  inspection identification number for inspection for this 
  page
  
  elementID identification number for the element 
  associated with this page
  
  **function**
  
  calls `decipherBinaryRep` for each checkbox or option button group 
  associated with this page
  
  End Sub

• **Public Sub decipherBinaryRep(rep As Integer, tag As String, repType As 
  String)**
  
  **input**
  
  rep the binary representation of the checked 
  options or boxes
  
  tag the name of the checkbox or option button 
  group associated with this representation
  
  repType “option” if this is an option button group, 
  “check” if this is a checkbox group
  
  **function**
  
  deciphers the binary representation into a “true” or “false” for each 
  box or button in the group 
  sets each box or button to the correct value
  
  End Sub

• **Public Function makeBinaryRep(repType As String, ck1 As Variant, ck2 
  As Variant, ck3 As Variant, ck4 As Variant) As Integer**
  
  **input**
  
  repType “option” if this is an option button group, 
  “check” if this is a checkbox group
  
  ck1 value contained in the first box or option in 
  the group
ck2 value contained in the second box or option in the group
ck3 value contained in the third box or option in the group
ck4 value contained in the fourth box or option in the group

function
returns the binary representation of the state (true or false) of each component in the group
the binary representation is made by adding a 1 if the first component is true, a 2 if the second component is true, a 4 if the third component is true, and an 8 if the fourth component is true

End Function

• Public Sub saveCheckBoxes(elementID As String)
  input elementID identification number of the element associated with this page
  function saves the binary representation of the state of the checkboxes and option button groups on this page to the database with gBridgeID, gInspectionID, and elementID

End Sub

• Multimedia Functionality

Global Variables
Public Const photoExtension As String = ".jpg" file extension used by the photo editing tool
Public Const sketchExtension As String = ".ssk" file extension used by the sketch tool

• Public Sub sortMultimedia(sortMethod As String, sorted As Recordset)
  input sortMethod "mmDate" to sort by date, "mmType" to sort by type
  output sorted recordset that contains a sorted list of the multimedia objects associated with gBridgeID

End Sub

• Public Sub newMultimedia(mmType As String, elementID As String, FileName As String)
  input mmType type of multimedia object to create
  elementID element identification to be associated with this object, "" if not associated with an element
  output FileName filename assigned to this object by the database
  function creates a new multimedia object in the database associated with gBridgeID, gInspectionID, and elementID (if not "") and returns the filename that it should be saved as

End Sub
• Public Sub viewMultimedia(FileName As String)
  input FileName filename of multimedia object to be viewed
  function determines if the object can be edited, based on the inspection date
  associated with it
calls the correct tool to view the object and indicates whether the
object is locked for editing
End Sub

• Public Sub createNewMS(Name As String, v1 As String, v2 As String, v3
  As String, v4 As String, v5 As String, u1 As String, u2 As String, u3 As
String, u4 As String, u5 As String, monitorID As Integer)
  input Name name of new monitoring system
  v1 through v5 variable names for the new monitoring
  system
  u1 through u5 unit descriptive names (ie, “in”, “cm”)
  associated with v1 through v5
  output monitorID identification number assigned to the
  monitoring system by the database
  function creates a new monitoring system associated with gBridgeID with
  Name, v1 through v5, and u1 through u5.
End Sub

• Public Sub addNewMSValues(monitorID As Integer, msDate As Date, v1
  As Double, v2 As Double, v3 As Double, v4 As Double, v5 As Double)
  input monitorID identification number of the monitoring
  system to add values to
  msDate date that these values are being added
  v1 through v5 values to be associated with value1 through
  value5 in the database
  function adds a new date and set of values to the monitoring system
  associated with monitorID
End Sub

• Public Sub getAllMS()
  function sets frmMain.monitoringSystems to all of the monitoring systems
  associated with gBridgeID
End Sub

• Public Sub getMSValues(monitorID As Integer, values As Recordset,
  variables As Recordset)
  input monitorID identification number of the monitoring
  system to retrieve values and variables for
  output values all of the value/date sets associated with this
  monitoring system
  variables the variable names and units names
  associated with this monitoring system
End Sub
• **Helper Functionality**

• **Public Function FilterField(rstTemp As Recordset, strField As String, strFilter as String) As Recordset**

  input
  - rstTemp recordset to filter
  - strField field to filter on
  - strFilter criteria to filter by

  function
  returns the filtered recordset

End Function

• **Public Function FilterDateField(rstTemp As Recordset, strField As String, strFilter As Date) As Recordset**

  input
  - rstTemp recordset to filter
  - strField field to filter on
  - strFilter criteria to filter by

  function
  returns the filtered recordset

End Function

• **Public Sub getDescriptiveName(elementID As String, descriptiveName As String)**

  input
  - elementID identification number of the element
  - descriptiveName descriptive name associated with elementID

  function
  looks up the descriptive name associated with elementID

End Sub

• **Public Sub Gettablename(elementID As String, tablename As String)**

  input
  - elementID identification number of the element
  - tablename tablename associated with elementID in database

  function
  looks up the tablename associated with elementID

End Sub

• **Handwriting Recognition Object Architecture**

  **Attributes:**
  - training file
  - text // The most recent text string interpreted.

  **Methods:**

  • **create(user)**
    Sent from controller object.
    Opens handwriting recognition software with training data and user references, which are maintained as training file.
    Done at startup and resident through all operations.
• **text write(ink)**
  Sent from annotation interface controller. Takes "ink" written in annotation box and converts it to text, which is sent back to the controller. Uses training file attribute to interpret the user's writing more correctly. The "ink" may contain characters that represent functions, such as copy and delete. These characters will be handled appropriately. The text is accumulated in the text attribute until the interface controller sends a message to the software that the writing is done. Then the text is returned.

• **train()**
  Opens handwriting recognition training window that allows the user to force the software to recognize handwriting with more accuracy. These changes are reflected in the training file.

• **Speech Interface Object Architecture**

  • **New Comment**
    ```
    {  
      Input: voice command "New Comment"
      Output: display blank edit window
      Function: Open edit window in active mode
    }
    ```

  • **Edit Comment**
    ```
    {  
      Input: voice command "Edit Comment"
      Output: display edit window with existing comments
      Function: select all text from text box
               Copy to clipboard
               Open new edit window
               Paste clipboard contents onto edit window
    }
    ```

  • **Complete**
    ```
    {  
      Input: voice command "Complete"
      Output: display spoken comments in text box
      Function: select all text from edit window
               Copy to clipboard
               Switch to main interface window
               Paste clipboard contents at current location of cursor
    }
    ```

**Attributes:**
- Caption name on button or tab

• **ButtonClick (CaptionName)**
  ```
  {  
    Input: voice command CaptionName name on the command button
    Output: activate corresponding button
    Function: Send corresponding control character sequence
  }
  ```
### Bridge

<table>
<thead>
<tr>
<th>district route segment offset</th>
<th>Main Structure Type</th>
<th>BMS Main Structure Approach Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>02 2085 0010 0603 tied-arch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BMS Approach Type Number**

over

**Location**

pittsburgh

**Overall Comment**

**Directions**

Tuesday, March 25, 1980 Page 1 of 1

**Inspection**

**Inspection Type**

Main Span(s) only

**Weather Conditions**

partly cloudy

Temperature (degrees F) 26

**Overall Comment**

for span 1-12 only

Tuesday, March 25, 1980 Page 1 of 1

**Traffic Safety**

**Bridge Railing Transition Approach Guiderail Drainage**

8 8 8 8

**Name**

traffic safety

**Bridge Railing**
good condition
Transition
good condition
Approach Guiderail
good condition
Approach Rail Ends
good condition

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition Description</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach Roadway</td>
<td>numerous transverse cracks in concrete</td>
<td>5</td>
</tr>
<tr>
<td>Pavement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shoulders</td>
<td>numerous transverse and diagonal cracks in concrete and near approach</td>
<td></td>
</tr>
<tr>
<td>Drainage</td>
<td>good condition</td>
<td></td>
</tr>
<tr>
<td>Bump At Bridge</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Approach Slab</td>
<td>four transverse crack 1/16” - heavy long. Wearing ADJ. EXP.</td>
<td></td>
</tr>
<tr>
<td>Relief Joint</td>
<td>near APPR. 3” wide transverse crack</td>
<td></td>
</tr>
</tbody>
</table>

Deck

Name Condition Rating
deck1 7

Top
good condition
Underside
good condition
Deck Drainage
light vegetation and debris in drains
Median Barrier
good condition
Curb-Parapets
minor full height hairline crack and minor scrapes from collision damage
Railings
side walk railing good condition
Sidewalks
good condition
Drains/Scuppers
light vegetation and debris in drains

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wearing Surface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wearing surface1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Deck Wearing Surface
concrete wearing surface in good condition

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superstructure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superstructure sapn 1-12</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Superstructure
good condition

Drainage System (Below Deck)
scupper pipes are in good condition

Bearings

bearings at expansion delamination across the structure show R1 to R2 with rust flaking

Tuesday, March 25, 1980 Page 1 of 1

Girders/Beams

Name
girder1

Girders/Beams

mainline box girder are generally in good condition. Walk through inspection revealed areas of spotty R2 caused by water

see page exterior shows light to moderate rusting mostly at FLB. Connections. Girder approaches are generally in good condition

w/ spotty R1 to R2 located at expansion DAMS

Monday, March 24, 1980 Page 1 of 1

Floorbeams

Name
florrbeam1

Comment

several floorbeams show R1 to R2 with rust flaking on the top and bottom flanges. Connection inspection revealed cracking in the

toe of the welds well as the web base metal of the floorbeams located at the top of the beam in the flange- web weld termination

area. Cracks vary from 1/2" in length to 5.75" in length.

Monday, March 24, 1980 Page 1 of 1

Stringers

name
stinger1

comment
stringers are generally in good condition although the stinger ends under the expansion joint area of the structure show R1 to R2 with flaking rust on the web ends as well as the upper and lower flange ends

Monday, March 24, 1980 Page 1 of 1

Diaphragms

Name
diaphragms

Diaphrams
generally are in good condition with spotty R1 to R2 at expansion dam area

Tuesday, March 25, 1980 Page 1 of 1

Truss Members

Name
upper cable connection

Truss Members

random checking of upper connection shows no visible cable defects. Connection assembly plates show very light and spotty R1 at the joints.

Tuesday, March 25, 1980 Page 1 of 1

Portals And Bracing

Name
bracing

Portals/Bracing
general inspection of the bracing between the upper chords shows them to be in good condition at this time.

Tuesday, March 25, 1980 Page 1 of 1

Paint Condition

Name Condition Rating
Paint Condition 77
New Paint If New Paint
0 Spot
Interior Beam/Girder
good condition
Fascias
good condition
Splash Zone: Truss/Girder
good condition
Truss
good condition
Bearings
good condition
Other
none

Tuesday, March 25, 1980 Page 1 of 1

Abutments
Name
abutment1
Wings
good condition
Bridge Seats
good condition
Backwall
near- in good condition far is not required
Stem
full height hairline cracks randomly spaced across face of near stem. Far not required.
cheekwalls
good condition

Embank-Slope-Wall
good condition

Footing
not visible

Piles
not visible

Scour
none detected

Undermine
none detected

Settlement
none detected

Wall Drainage
good condition

Tuesday, March 25, 1980 Page 1 of 1

Piers And Bents

Name
Pier 1

Bridge Seats
good condition

Cap
light coverage rust stains and vertical hairline cracking randomly spaced underside

Cheekwalls
good condition

Columns/Stems
of cap at u.s. corner of pier 10 is cracked delaminated and spalled. Spalls are presented at the column bases of pier 1. Downstream

column of Pier 4 is scaling.

Footings
<table>
<thead>
<tr>
<th>Channel</th>
<th>Name</th>
<th>Condition</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Channel 7</td>
<td>good alignment</td>
<td></td>
</tr>
</tbody>
</table>

**Scour**
none detected

**Undermine**
none detected

**Settlement**
none detected

**Collision Damage**
none

**Channels**

**Name Condition Rating**

**Channel 7**

**Channel Alignment**
good alignment

**Scour**
none

**Embank Erosion**
none

**Chan/Emb Protection**
concrete walls are stable but have heavy cracking, apalls, and large area of delamination

**Debris**
none

**Vegetation**
none

**Highwater Mark**
no high water mark visible

Streambed Material

none

Tuesday, March 25, 1980 Page 1 of 1
Appendix C

Mechanical and Industrial Design

Figure 1: The Vest
Figure 2: The Buckle

Figure 3: Adjustment Strap
Figure 4: Rear Pouch

Figure 5: The Container
Figure 6: Clipping MIA to the Vest

Figure 7: MIA at Rest
Figure 8 MIA open for use

Figure 9: Contents of Container
Appendix D  
MIA User Guide

Getting Started

To run 'Mobile Inspection Assistant' (MIA), it is necessary to use an IBM PC compatible computer running under Windows95 or Windows NT. To run the program, press on 'Start' button on the left corner of the screen and click on 'Run'. The user will be asked for the appropriate location of the file (MiaIntegrated.exe). Press on 'Browse', locate the file 'MiaIntegrated.exe' and click OK. The user can also run the program from DOS prompt.

In order to be able to issue command via speech, the user has to open 'Dragon NaturallySpeaking Deluxe' by pressing on 'Start', choosing 'Programs' then selecting 'Dragon NaturallySpeaking Deluxe'. The user should see Dragon NaturallySpeaking and Mia icons on the task bar, which are located at the bottom of the screen.

LOGIN

The first thing that the user is going to see is the Authentication (LOGIN) window asking for user name and password.

SELECT BRIDGE FILE

Once the user name and password have been authenticated, the 'Select Bridge File' window is displayed. In this window, the user will see four text entry boxes that represent the bridge identification number and two list boxes which display the bridge files that are available. The office list box on the right, displays the bridge files that are in the office (server) and the MIA list box on the left displays the bridge files that are currently on MIA. The user has two options: either to load an existing bridge file from the office archive or to create a new bridge file.

To load the existing bridge file, the user may choose the file from the office list box on the right side of the select bridge file window or the user can begin typing in the text entry boxes that are located on the top of the window. The user may enter all or part of the identification number. If the user presses search, only matching files are displayed on the right list box. The user may click on the file number in the office list box at any time to select it. If there is no possible match in the database, the pop up window showing "No Matches" will be displayed. To move the selected bridge file to MIA, the user must press the check out button. The selected bridge file will then be removed from the office list box and displayed on the MIA list box.

To create the new bridge file, the user must enter a new bridge identification number and then click "NEW". The bridge identification number of a new bridge file will be displayed on MIA list box.

To display the bridge file, highlight the bridge identification number from MIA list box and click DISPLAY button to move to the main interface to begin the inspection process.
MAIN INTERFACE

In the main interface, the user should see a series of TABS located at the top of the window, a navigation box located at the left side of the window, and a TOOLBAR button located at the bottom left corner. The working area is the remaining area in the window, which will display the form that the user is currently working on.

There are six tabs on the top of the window. These are current report, previous report, bridge info, album, sketch template, and monitoring system tabs.

'Current Report tab' contains information relevant to the current inspection. 'Previous report tab' contains information of past inspections, sorted in chronological order. 'Bridge info tab' contains items and information relevant to the bridge as a whole and this information is time independent, such as bridge identification number and location. 'Album tab' contains the collection of photos and sketches associated with the bridge. 'Sketch Template tab' contains the sketch templates of the common bridge elements. These sketches are time independent and are used as a template for a new sketch of the same element. 'Monitoring System tab' contains the information of monitored elements and their behavior (crack, rotation, or translation etc.).

When the user clicks on one of these tabs, the selected tab is highlighted, indicating the part of the bridge file that the user is working on. The items displayed on the navigation box is dependent on the highlighted tab. The navigation box will be updated each time the user presses on any of these tabs on the tab bar. For example, if the Current Report tab is highlighted, the items in the Navigation box are the bridge elements to be inspected. When the user clicks on Album tab the items in the Navigation box are updated to list the names of the associated photos and sketches.

The working area contains the form corresponding to the item currently highlighted in the Navigation box. For example, if the user clicks on superstructure in the Navigation box under Current Report tab, the form for superstructure (name, condition code, and comment) will be shown in the working area.

The toolbar button will bring up a collection of buttons that will be used while performing inspection process. These buttons are copy, cut, paste, file, print, quit, backup and undo. The functionality of each button will be discussed later on.

After the user clicks on DISPLAY button from the 'select bridge file' window, the main interface is displayed. The 'Bridge info' tab is automatically highlighted to enable the user to update any general information about the bridge. The items in the navigation box include history of the bridge, photographs, maps, etc. Currently on this version, only general information of the bridge such as its id number and location are available. The user can update any information under the 'Bridge Info' tab, except for the bridge identification number.

Viewing Information from the past

To view the information of any particular item, click on the tab that contains this item, then click on the item in the navigation box. For example, if the user wants information on the superstructure from previous inspection, then the user has to click on 'Previous Report tab'. The items in Navigation box will be updated to list inspection dates. Select the appropriate inspection date, to display the items constituting that report. Now click on superstructure item in the Navigation box and the information of the superstructure is displayed in the working area. To navigate back to the earlier inspection report, simply click on 'previous report', which is the first item in the navigation box. The items in the navigation box will be updated to enumerate inspection dates.
The user can also view a sketch and photo from 'Current Report', 'Previous report' and 'Album' tab. The user can view the photo in 'Current Report' or 'Previous Report' tab by selecting a photo or a sketch from a drop down list box, followed by a click on the View Sketch or View Photo button. The photo editing or sketch software will be launched to display the selected item with its associated comment. In order to switch back to the main interface, the user clicks on the 'Done' button.

The user can also view the photo or sketch from 'Album' tab by clicking on the 'view' button in the working area after clicking on the photo or sketch in the Navigation Box. The sketch or photo can be viewed in the same way as viewing under previous and current report tabs.

**Working on Current Report**

When the user clicks on the 'Current Report' tab, the items in the navigation box are updated to list the bridge elements that have to be inspected. The user is allowed to change any information, including inserting photo(s) and sketch(es) while working on the current report.

**Making new or changing comment in the text entry box**

The user can modify information contained in the text box by simply clicking on the text box using the stylus or using speech command (consult speech interface user guide). The user can make a new comment or edit the existing comment using virtual keyboard, handwriting recognition, or speech recognition (consult the respective sections of the user guide).

Alternatively, the user is provided with the facility to copy an entire comment from a previous report to the current report with a single command.

The systematic sequence of operations that need to be performed, to execute any of the features provided on MIA, are enumerated below.

*To copy the entire form from the previous report to current report*

1. Click on 'previous report tab' and the item that needs to be copied.
2. Click on 'Toolbar' button to bring up a collection of toolbar utilities.
3. Click on 'copy' button to display a dialog box asking the user to indicate what the user wants to copy.
4. Select 'this page' and click 'OK'
5. Click on 'Current Report tab'
6. Click 'Paste'
7. The item will be automatically inserted to the list in the Navigation box.

*To cut an entire form from the current report*

The user will notice that there are currently 2 items in the list which have the same name, owing to the above action.

1. Click on the item in which there is no information.
2. Click 'Cut' button which will display a cut window.
3. Select 'Cut this page' and click 'OK' to remove the unwanted item.

*To copy part of the comment from previous report to current report*

1. Click on 'previous report tab' and the item that has the comment to be copied.
2. Use a stylus to swipe the comment to be copied. The selected text will be highlighted.
3. Click 'Toolbar'
4. Click 'Copy'. The selected comment will be automatically pasted in appropriate location in 'Current Report'.
To make a new sketch

If the user want to make a new sketch, starting with a blank page

1. Click on 'Current report' tab and click on the item that the sketch will be associated with.
2. Click on the 'toolbar'
3. Click 'insert'
4. Select 'Sketch'
5. Click 'OK'
6. Draw a sketch on the sketch window. Comments may be added to the comment box using either of the three modes of annotation (refer annotation)
7. Click 'Done' to switch back to the main interface.

To add a new sketch from an existing template

1. Click on 'Sketch Template' tab
2. Select the appropriate template from the Navigation Box
3. Click 'Toolbar'
4. Click on 'Copy' and the user should see the Copy window.
5. Choose 'Selected Sketch'
6. Click 'OK' to go back to the main interface
7. Click on 'Current report' tab
8. Click 'Paste' and the sketch will be automatically inserted into the sketch combo box
9. Select the sketch that was newly inserted (the name should be the same as the template that was copied)
10. Click 'View Sketch'
11. Click on appropriate icon in the Sketch software (SmartSketch) and draw on it, to make necessary changes and annotate damage information.
12. Click 'Done', when complete, to switch back to the main interface.

To make a new sketch by using the sketch from previous report or album, the user follows the same steps as making a new sketch from a template but instead of copying a sketch from a template the user now copies the sketch from the previous report or album.

To insert a new Photo and edit it

1. Take the photo
2. Click on 'Current Report' tab
3. Click on the item to which the photo will be attached.
4. Click 'Toolbar'
5. Click 'Insert' and the insert window is displayed.
6. Select 'Photograph'
7. Click 'OK'
8. The photos that have been taken by the user will appear in the photo window. Select the photo that the user wants to insert and click 'download' in the comment form.
9. In order to edit the photo, the user has to click on 'Annotate' button and the photo-editing (Paint Shop Pro) tool will be displayed.
10. Click on appropriate buttons to activate tool used for annotation such as pencil, brightness etc.
11. Add comments if required in the text box provided, using one of the three possible modes of annotation.
12. Click 'Done' when complete, to switch back to the main interface.
To insert a new bridge element to the report

In some cases, the user may want to insert an element into the bridge report. For example, the user may want to report the condition of two different girders, located on span 1 and span 2. The program allows the user to do this by following the simple steps as follows:

1. Click on 'Current Report' tab
2. Click on the element for which the user wants a duplicate (e.g., click on girder if girder needs to be inserted)
3. Click 'Toolbar'
4. Click 'Insert' which will bring up the insert window.
5. Select 'Bridge Component'
6. Click 'OK'

The element will be inserted into the Navigation box.

To rename the element

There are two ways to rename the element's name, e.g., directly changing it from the Navigation Box or changing it from the name text entry in the form.

To change directly from the Navigation Box, follow the following steps:

1. Select the item to be renamed
2. Click twice on the item
3. The cursor will appear on the item
4. Change the name of the item, by writing with a stylus or using a virtual keyboard.

To change the name from the text entry box

1. Select the item to be renamed
2. Click on the 'name' text entry box and rename it, using any of the annotation modes.

In this case the name will not be updated in the navigation box until the user clicks on the Navigation box.

To remove an item (element)

1. Click on the item to be removed
2. Click 'Toolbar'
3. Click 'Cut' to bring up the cut window
4. Select 'Cut this page'
5. Click 'OK'

Working on Monitoring System

To create a new monitoring system

1. Click 'Monitoring System' tab
2. Click 'Add Element' and the Add new element window is displayed
3. Enter the name in the text entry box 'Monitor element Name'
4. Input the value in the 'Variable' and 'Units' text entry box that the user wants to measure.
5. Click 'Done'
6. Click 'Add Values'
7. From a drop down list box, select the item to which a new value has to be added. Click 'Add'
8. Repeat the above step to add as many new entries as required.
9. Click Done, when all measured values have been entered.
10. The Graph will be displayed and the table displayed below the graph.

To update the monitoring system

1. Click 'Monitoring System’ tab
2. Click on an item that needs to be updated, from the box on the left
3. Click 'Add Values’
4. Select the item that the user want to add the measured value from the Pull-down menu and click 'Add’
5. After all values are added, click 'Done’
6. The Graph will be displayed and the table displayed below the graph

Changing to Another Bridge File

1. Click 'ToolBar’
2. Click 'File’ which will display the 'Select Bridge File' window
3. Select the bridge id from the list on the left.
4. Click 'Display’

Monitoring System

The Layout

Upon clicking on the Monitoring System tab, the user is brought to the Monitoring System. Here the format is a little altered from. There are two buttons in addition to the "ToolBar” button located on the screen--"Add Element” and "Add Values”. Still on the left of the screen, the tab-specific menu is a short list of monitored elements. Directly above the list is a label signifying which monitored element is currently being displayed. Directly below the list is the "Add Element” button which is used to add a new monitoring system for an element. Below this button is the familiar "ToolBar” button. To the right of the "Add Element” and "ToolBar” buttons is a table for displaying the variables, dates, and all data associated with the monitored element for the last four inspections. The center of the screen is occupied by a line graph for displaying the data found in the table. The right side of the screen has a comment box for the inclusion of any additional comments to the monitoring system. Finally, directly beneath the comment box and above the table is the "Add Values” button. This button allows the user to enter values for the current date to the monitoring system.

Navigation

Selecting an element in the tab-specific menu at the right of the screen will make that monitored element active. That is to say, the label above the tab-specific menu, the data in the table and graph, and the text in the comment box will all change to the information stored for that monitored element. In addition to reviewing the information, the user can take three actions to change the data: Adding Comments, Adding Values, and Adding New Elements.

Adding Comments

To add or update a comment, the user takes the same familiar action he or she would in editing a form in the Current Report tab. The user selects the comment box with the pen or says, "Go to comment," and begins editing in the normal fashion.
Adding Values

To add or replace values in the currently selected monitored element for the current date, the user selects the "Add Values" button located below the comment box. This action will bring up the Add Values form. There are two fields located on this form in addition to two buttons, "Add" and "Done". The first field is a drop-down list of all variables monitored for the element. To access the list, the user selects the arrow located on the right side of the field. The second field is the value field. Here the user would enter a value for a particular variable. If "Add Values" is inadvertently selected, the "Done" button may be pressed immediately and no values will be added. The process of adding values is accomplished in the following way. A variable is selected from the first field. A value is entered in the second field. The "Add" button is selected. This process is repeated for each variable. Once all data has been added, the "Done" button is selected. This final action will cause the Add Values form to disappear and the table and graph to be automatically updated with the new data. If the user wishes to alter any or all of the values for the current date, the whole process described above must be repeated.

Adding New Elements

To add a new element to the list of monitored elements, the user selects the "Add Element" button located below the list of monitored elements. Upon the selection of this button, the Add Element form will appear. This form has several fields the user must enter information into. The top-most field is for the name of the new monitored element. The next ten fields are for five variable-unit combinations. The first pair is already entered as "Temp" and "F". The user may choose to change this pair or leave it as is. Then the user would continue to add variable names and their units to the list. Unused variables may be left blank. Once the user is finished, the "Done" button is selected. If the user wishes to cancel the adding of a new element, the "Cancel" button is selected.

SKETCHES

The sketch UI has 3 main buttons in addition to the buttons (icons) given on the sketch tool itself, SmartSketch. The lowest button (Toolbar) is to show and hide the buttons common to MIA at all times (cut, copy, paste, undo, etc.) The other two buttons and icons are explained below:

A. Clear:

The clear button is only active when the open sketch is unlocked and can therefore be edited. The clear button will clear the entire sketch. This action can be undone by pressing the Toolbar button and then the Undo button. A dialog box will appear asking the user if they are sure they wish to clear the sketch. The user can respond by either hitting the 'Yes' or 'Cancel' button.

B. Close/Done:

The Close/Done button’s caption (either Close or Done) depends upon the status of the opened sketch. If the sketch is unlocked, then the caption on the button is 'Done', otherwise, the caption is 'Close'. If the user is finished editing or viewing the sketch, they may close the tool by clicking on the Close/Done button. This will bring up a dialog box asking they if they are sure they wish to quit out of the sketch tool. If the file is locked (Close), then the user will have the choice to either close the sketch (given by a 'Yes' response), or cancel the close request (given by a 'Cancel' response). If the file is unlocked (Done), then the user will have the choice to one, close the sketch and save any changes (given by a 'Yes' response), two, close the sketch and do not save any changes (given by a 'No' response), or cancel the done request (given by a 'Cancel' response).
C. Arrow (icon):

The arrow icon on the left-hand side of the sketch interface allows the user to select a part of the sketch by clicking on the area to be selected after clicking on the arrow icon.

D. Pencil (icon):

The pencil icon allows the user to draw on the white workarea. In addition, icons appear on the bottom left-hand side of the sketch tool which allow the user to select pre-defined shapes, the color of the line, the thickness of the line, and the type of line (solid, dashed...).

E. Eraser (icon):

The eraser icon allows the user to erase any part of the sketch. In addition, a pull-down menu appears which allows the user select the size and shape of the eraser.

F. Magnifying Glass (icon):

The magnifying icon allows the user to zoom in and out of the sketch. Two icons will appear below the magnifying icon, which allow the user to select between zooming in versus zooming out. The magnifying glass with the plus sign allows the user to zoom in while the one with the minus sign will allow the user to zoom out.

G. Letter A (icon):

The letter A icon, allows the user to add text to the sketch. In addition, pull-down menus appear in the lower left-hand corner which allow the user to change the font size and type.

H. Lasso (icon):

The lasso icon allows the user to select an arrow of the sketch by drawing any shape around the area to be selected. Unlike the arrow icon, which allows the user to only select lines of the sketch, the lasso tool allows the user to select any shape or amount of the sketch.

I. Magnet (icon):

The magnet icon appears when the user is in select mode (arrow icon). This will turn on and off the snap to grid option.

J. To Show the Grid:

To show a grid under the sketch, the user must goto the Format pull-down menu at the top of the tool. In the menu, the user must then select the Document option. Then a window will appear, in which they can then select 'Show Grid' and click the OK button and the grid will appear.
PHOTOS

The photo UI has seven buttons and a text box. The lowest button is for the toolbar to show and hide buttons like cut, copy, paste, etc. The other six buttons are specific to the photo controller. They are:

A. Ruler:

This button makes the grid visible on the picture and also brings up a ruler at the x and y-axis. This will be useful to get a relative idea of the distances in the photograph.

B. Pencil:

This allows the user to sketch on the photograph.

C. Zoom/Normal:

This button allows the user to zoom into a particular area. To zoom at any particular place of the photograph you have to click on this button first and then click at the location you want to zoom. Successive zoom can also be done by continuing to click on area to be zoomed. When the button is clicked, its name changes to 'normal'. So to revert back to the normal size you have to click on it again.

D. Select:

This allows you to select a particular area to cut or copy. It is a free shape tool and hence the user can select arbitrary shaped areas.

E. Brightness:

This brings up the brightness control, which also shows a preview of any changes that the user wants to make.

F. Download/annotate/close/done:

This button functions differently according to the stage and type of operation being done. This is explained below. For reference to this button we will call it the doneclose button.

There are two ways the Photo controller may be called:

1. To download a photograph:

In this case the UI comes up with the doneclose button showing the caption 'Download'. The Ruler, Pencil, Zoom, Select and Brightness buttons are disabled at this time. The toolbar also has the appropriate buttons disabled at this time. The download software (Image Expert) is also brought up at this point and the pictures in the camera are displayed in thumbnail form. The user has to select a picture by clicking on its thumbnail and click on 'Download' (the doneclose button). On clicking it a warning comes up to verify whether the user has selected a picture. The options are yes, no or cancel. The user has to click 'cancel' in order to abort the download sequence and go back to main controller (in case he/she decides not to download any photograph at this point. If 'no' is pressed then no work is done and the user is returned to its previous state. If 'yes' is pressed then the controller starts downloading the photo. The progress is shown on a window. The doneclose button caption at this point changes to 'annotate'.

97
Once the downloading has finished the user has to click ‘annotate’ (doneclose) button and is given an option to decide whether to go ahead with editing the downloaded photograph or go back to main form without further editing. If editing option is chosen the controller opens the photo editing tool with the present filename (unlocked).

2. To edit a photograph:

There may be two cases - when the file is locked (photograph is from a previous report) or unlocked (photograph from present report). If the file is locked then the Pencil button is disabled and the other buttons are enabled. The cut and paste buttons of the toolbar are also disabled. The caption for the doneclose button is 'close'. Thus the user can only view this photograph and copy from it. For an unlocked file all the buttons are enabled and the doneclose button is named 'done'. The user has a freedom to add sketches to the photograph, cut a portion of it, paste some thing on it, change its brightness, etc. When the user is done with the editing the photograph and wants to return to the main form he/she clicks the 'done' (doneclose) button. He/she is given an option at this stage to choose whether to save the changes he made or save it as it was previously. Depending on the selection the file is either saved or not saved and the user is returned to the main form.

Annotation

There are three modes by which the user can annotate damage information.

- speech commands and dictation
- handwriting the comment using a stylus
- clicking on characters using a virtual keyboard and stylus

Speech commands and dictation

Speech Recognition allows hands free operation at the field. The user can annotate damage information an also navigate through the user interface using voice commands. The voice commands mimic that are possible on the user interface.

The speech recognition software is user dependent. In order to use speech recognition for annotation, the user must go through a 30 minute training program that is prompted by the software at the time of creating a user file. The user is required to read through a passage, so that the speech recognition engine can store acoustic information, to facilitate efficient recognition. Further, in order to ensure that the phrases listed below as commands are not recognized as text, the user must validate his user file by updating the Global.dvc file to include the commands listed below, in the folder created for him by the software.

The voice commands that can be used are listed below. It must be kept in mind that the user should pause briefly before and after saying a command so that it will be recognized as a command rather than as text.

- **Start** : starts MIA  
  *Opens the authorization screen*

- **Form** : switches back to the previous window  
  *Switches back to the main interface from the NaturallySpeaking window*

- **Annotate** : add a new comment  
  *Opens a blank NaturallySpeaking window*

- **New Comment** : add a new comment (same as annotate)  
  *Opens a blank NaturallySpeaking window*
• **Edit Comment** : edit an existing comment
  
  *Opens a NaturallySpeaking window, with the existing comment*

• **Clear Screen** : clears the text on the NaturallySpeaking window
  
  *Opens a blank NaturallySpeaking window*

• **Cancel Screen** : clears the text on the NaturallySpeaking window
  
  *Switches back to the main interface*

• **Complete** : saves the comment on to the database
  
  *Switches back to the form, with the comment pasted at the specified cursor location*

• **<TabList> Tab** : highlights the corresponding tab
  
  *TabList values: Current Report, Previous Report, Bridge Info, Sketch Template, Album, Monitoring System*

• **next** : executes the tab key on the keyboard
  
  *changes focus to the next element on the tab order*

• **previous** : executes the (shift + tab) key on the keyboard
  
  *changes focus to the previous element in the tab order*

  *(the next and previous commands are used for navigating within the form)*

• **Click <CommandList>** : click on the corresponding command button
  
  *CommandList values: Toolbar, Show Toolbar, Hide Toolbar, Display, Connect, Search, New, Check In, Check Out, Done, Cancel, Clear, Add, View, View Sketch, View Photo, View Results from Previous Search*

• **Choose <ToolList>** : click on the corresponding tool bar button
  
  *ToolList: File, Copy, Cut, Paste, Insert, Undo, Backup, Print, Quit*

• **Insert <InsertList>** : clicks on the corresponding option in the Insert dialog box
  
  *InsertList: Bridge Component, Photograph, Sketch*

• **Copy <CopyList>** : clicks on the corresponding option in the Copy dialog box
  
  *CopyList: This Page, Select Sketch, Select Photograph*

• **Index** : changes focus to the tree view
  
  *Highlights the corresponding element name on the tree hierarchy*

• **Cursor** : locates the cursor on the first text box in the form currently displayed in the working area

• **Move <Direction>** : moves the cursor by one step in the corresponding direction
  
  *Direction: Up, Down, Left, Right*

• **Page Up** : moves the vertical scroll bar up
  
  *Changes focus to the vertical scroll bar in the working area, and moves it up*

• **Page Down** : moves the vertical scroll bar down
  
  *Changes focus to the vertical scroll bar in the working area, and moves it down*

• **Photo** : positions cursor on the photo list tab

• **Sketch** : positions cursor on the sketch list tab
• List box : displays drop down list box associated with the current cursor location

**Commands used for annotation in the Edit (NaturallySpeaking) Window only :**

- **Scratch that** : deletes the last word or phrase
- **Paste that** : pastes the current contents of clipboard
- **Select <word/phrase>** : highlights the chosen words or phrase
- **Spell that** : allows oral spelling of words
- **New Line** : moves cursor to a new line
- **New Paragraph** : moves cursor to a new paragraph
- **Move to end of document** : moves cursor to the end of document

**Handwriting**

The handwriting recognition software is running in the background at any point of time. In order to use the handwriting recognition software, the user merely has to click on the text box and start writing, using a stylus. Some of the features provided by the software are enumerated below.

- **Editing commands**
  Delete is indicated by two diagonal lines: up and to the right, then down and to the left, retracing the same line. Making a capital D with a circle drawn around it also deletes. To erase a word, select the word by dragging the pen over it and make either of the delete gestures. To erase a word, select the word by dragging the pen over it and make either of the delete gestures.

- **The following steps need to be followed to improve recognition**
  - pay attention to the order and direction of strokes
  - separate characters
  - distinguish between uppercase and lowercase by varying the size of the characters
  - write large slashes, parentheses and brackets
  - close loops
  - avoid excessive retracing
  - avoid connected pairs of letters

For further help, refer to hwriter.hlp file available online.

**Virtual Keyboard**

In order to use the virtual keyboard, click on the virtual keyboard icon displayed at the left corner of the text box. This would display the virtual keyboard. The information is entered by clicking on the characters on the keyboard using a stylus.

**INSTALLATION INSTRUCTIONS (database and controller only)**

**TO INSTALL A NEW SYSTEM**

```bash
mkdir d:\bridge
mkdir d:\bridge\integrated

contents of d:\bridge
temp.ssk
```
contents of d:\bridge\integrated
MIA.mdb
MIAIntegrated.exe

All multimedia files will be saved into d:\bridge by the program.

**TO INSTALL A DEMO**

mkdir d:\bridge
mkdir d:\bridge\integrated

contents of d:\bridge
bridge1\
bridge2\
temp.ssk

contents of d:\bridge\integrated
MIA.mdb
MIABackup.mdb
MIAIntegrated.exe

contents of d:\bridge\bridge1
photo\
sketch\
template\

contents of d:\bridge\bridge2
ins1995262\
ins1997302\
photo\
sketch\
template\

contents of d:\bridge\bridge2\ins1995262\photo
AB00-1.jpg

contents of d:\bridge\bridge2\ins1997302\sketch
AB00-1.ssk

To reset the demo, copy MIABackup.mdb and save it as MIA.mdb (this will erase all of the entries made in the database). Remove all of the multimedia objects created when you ran the demonstration such that the file structure contains only those files listed above.