Design Presentation

NIRSS
NASA Icing Remote Sensing System

A system for detecting and measuring icing conditions aloft

Patrick Carmichael
Matthew Hulse
Stephan Zednik
CU: Computer Science
Undergraduate Senior Project

CSCI 4308/4318
Fall 2005 to Spring 2006

Mandatory class for completion of BS Degree in Computer Science
Presentation Checkpoints

- Design Overview
- User Interface
- System Design
- NIRSS Prototype
- Summary & End
NIRSS Design Overview

- Design Overview
  - Background
  - The Problem
  - The Solution
- User Interface
- System Design
- NIRSS Prototype
- Summary & End
Background: Icing Detection

- Icing termed high priority towards increasing aircraft safety
- *Avoid and Exit* strategy
  - remote measurement of an icing environment
  - Need software to visualize icing hazards for strategy
- NIRSS developed for detecting hazards
Problem: NIRSS has limitations

- User interface constraints
  - Local only display
  - LabVIEW installation required
- Platform constraints
  - Difficult to upgrade
  - Barrier to adoption
Solution: Upgrade NIRSS

- Improvements to NIRSS
- NIRSS Upgrade Strategy
- NIRSS Major Requirements
  - Environmental
  - Functional
- Conceptual overview of NIRSS
- NIRSS System Diagram
NIRSS Improvements

- Display enhancements
  - Network accessible
  - Viewed through browser
- Platform enhancements
  - Logic re-written in standard programming language
  - Improved data and memory management
NIRSS Upgrade Strategy

- ANSI C fusion core
- Java applet for display
- Apache Tomcat as display server
- Fusion core configuration file
- Improved situation awareness
Fusion System Environment Requirements

- **Software environment**
  - ANSI C
    - GCC 4.0
  - Sun Java 1.4.2
  - Apache Tomcat 4.28
  - Linux OS
    - kernel 2.6.9+ (stable)

- **Hardware environment**
  - x86-based workstation
    - 2GHz+ CPU
    - 1GB+ RAM
    - Outgoing Ethernet connection
Remote Display Environment

Requirements

- **Software environment**
  - JavaScript enabled web browser
    - IE 5.0+
    - Firefox 1.0+
    - Safari 2.0+
  - Sun Java 1.4.2

- **Hardware environment**
  - 800MHz+ CPU
  - Network access
    - 256k+ broadband
  - Color monitor
  - Keyboard + Mouse
Fusion Core Requirements

● Written in ANSI C
● Real-time operation
● Administration by configuration file
● Produces secondary data
  ● cloud profile
  ● super cooled liquid water profile
  ● hazard severity profile
● Flag data ingest stalls
● Sanity checks on data
Display Requirements

- Two main displays
  - Radar reflectivity profile with overlay
    - Cloud base (Radiometer)
    - Cloud base (Ceilometer)
    - Temperature altitudes
  - Hazard severity
Display Requirements

- Secondary displays
  - Radiometer temperature profile
  - Integrated Liquid Water (ILW) plot
  - Cloud profile
  - Super cooled liquid water (SLW) profile
  - Instrument status
  - Rain flag
NIRSS Conceptual Overview

- Cloud profile
- Integrated Liquid Water (ILW) distributed across cloud layers
- Icing hazard determination
- Log file/file buffer
- Apache Tomcat Server
- Java applet
NIRSS System Diagram
Presentation Checkpoints

- Design Overview
- User Interface
- System Design
- NIRSS Prototype
- Summary & End
NIRSS User Interface

- Design Overview
- User Interface
  - NIRSS Display
  - Configuration File
- System Design
- NIRSS Prototype
- Summary & End
NIRSS: Display

Diagram showing the interaction between Data, Fusion Processor, Configuration File, Administrator, Display System, and User.
NIRSS Display Invocation

- Point your browser to the Fusion Core Server: http://<ip_address>
- Select the date from the drop-down or enter the date manually
- Click load.
Display Components
Primary Display: Reflectivity
Display Components
Primary Display: Hazard Index
Display Components Secondary Displays
Display Components Secondary Displays Cont.
Display Components Secondary Displays Cont.
Display Components Interface
The NIRSS Display
NIRSS: Fusion Processor
NIRSS Invocation (Fusion Core)

- Pre-compiled
- Configuration file existence
- Start Tomcat (*apachectl start*)

**Invocation:**

```
% nirss config.txt
% nirss <config_file>
```
Tomcat Administration

- Tomcat Server Manager
  - Set which servlets are being hosted
  - Pause server
- Web.xml file
  - Servlet configuration and location
Configuration File Options

- Designed to act like a standard *NIX configuration file
- Contains
  - File paths
  - Initial display interval
  - Other Fusion Core constants
- Tag must be hard-coded into Fusion Core system to add new variable to configuration file
### Configuration File

```
#<tag> <value>

#SET PATH TO LOG FILE DIRECTORY
logpath "/data/nirss/log"

#SET PATH TO RADAR FILE
radarpath "/data/nirss/data/radar.dat"

#SET MINIMAL NORMALIZED REFLECTIVITY
#THAT CORRESPONDS TO CLOUD PRESENCE
cloud_reflectivity_bound 1.0

#SET DISPLAY INTERVAL (SECONDS)
display_interval 300
```

- ASCII format
- Variables associated with tags
- Fusion Core aware of all possible tags
- Tags and variables space delimited
- Variables containing spaces in quotes
- Comments denoted by ‘#’ at start of line
Presentation Checkpoints

- Design Overview
- User Interface
- System Design
- NIRSS Prototype
- Summary & End
NIRSS System Design

- Design Overview
- User Interface
- System Design
  - Hardware
  - Current Design
  - Proposed Design
  - Classes/Components
- NIRSS Prototype
- Summary & End
Hardware Instrumentation

- Radiometrics TP/WVP-3000 Radiometer
- Technology Service Corporation X-Band Radar
- Vaisala Lidar Ceilometer
Current Design

- Written entirely in labVIEW
- All data analysis logic present
  - ILW distributed via fuzzy algorithm
  - Hazard severity calculated
  - All layers treated independently
- real-time execution
- Localhost limited display
- Not memory efficient
Proposed Design

- Divide NIRSS into two systems
  - Fusion Core
    - Performs all data analysis
    - Written in ANSI C
  - Display System
    - Java Applet
      - Browser based display page
      - Allows for remote display
    - Apache Tomcat
      - Serves data to Java applets
Proposed Design Architecture
Data Ingest Architecture
File Reader Components

- All file reader functions have the same general layout and interface
- C structures passed by reference
- Check for data ingest stalls

```c
void config_filereader(&config_data config,
                        &control_struct config_control)
```
Radar Data Struct

```c
struct radar_data {
    double month;
    double day;
    double year;
    double hour;
    double minute;
    double second;

    double size;

    vector<float> range;
    vector<float> reflectivity;
    vector<float> normalized_reflectivity;
};
```
Control Data Struct

```c
struct control_struct
{
    string filepath;
    bool eof;
    bool stalled;
    bool data;
    int offset;
    int previous_offset;
    time_t timestamp;
    time_t previous_timestamp;
};
```
Fusion Processor Architecture
Fusion Processor Architecture

- **Data Regrid**
  - Cast data into sets with one minute time deltas
  - Drop excessive data
  - Use last if input slow
  - Interpolate data onto a range with constant delta
- **Quality Control**
  - Check for anomalies in the data stream
  - Replace bad data with default values (where applicable)
  - Compute normalized Reflectivity profile
Derived Data Processor

- Cloud profile
- Cloud base (Radiometer)
- Temperature Altitudes
  - H₂O freezing point
  - Minimal temperature where SLW possible
- Cloud layers
  - Number
  - Depth
  - Altitudes
  - Temperature at bases/tops
  - ILW
Derived Data Methods

Compute Cloud Profile

```c
void get_cloud_profile(&struct_secondary secondary,
                        &struct_radar radar,
                        &struct_config config)
```

Compute Cloud Layers

```c
void get_cloud_layers(&struct_secondary secondary,
                       &struct_radar radar,
                       &struct_radiometer radiometer,
                       &struct ceilometer ceilometer,
                       &struct_config config)
```
Cloud Logic Processor

- Super-cooled liquid water concentration profile
  - Weighted sum of four distribution methods
    - Uniform
    - Wedge
    - Temperature
    - Inverse Reflectivity

- Hazard severity profile
  - Severity determined by SLW concentration
Cloud Logic Methods

Create Icing Hazard Profile

```c
void compute_icing_hazard(&struct_secondary secondary,
                         &struct_config config)
```

g/m$^3$ liquid water (LWC):

<table>
<thead>
<tr>
<th>LWC</th>
<th>Severity</th>
<th>Hazard Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWC &lt; 0.01</td>
<td>0</td>
<td>no icing</td>
</tr>
<tr>
<td>0.01 ≤ LWC &lt; 0.017</td>
<td>1</td>
<td>trace</td>
</tr>
<tr>
<td>0.017 ≤ LWC &lt; 0.03</td>
<td>2</td>
<td>trace-light</td>
</tr>
<tr>
<td>0.03 ≤ LWC &lt; 0.066</td>
<td>3</td>
<td>light</td>
</tr>
<tr>
<td>0.066 ≤ LWC &lt; 0.12</td>
<td>4</td>
<td>light-moderate</td>
</tr>
<tr>
<td>0.12 ≤ LWC &lt; 0.2</td>
<td>5</td>
<td>moderate</td>
</tr>
<tr>
<td>0.2 ≤ LWC &lt; 0.37</td>
<td>6</td>
<td>moderate-heavy</td>
</tr>
<tr>
<td>LWC ≥ 0.37</td>
<td>7</td>
<td>heavy</td>
</tr>
</tbody>
</table>
Display Architecture
Apache Tomcat

- JSP Servlet Container.
- Based off of Apache Web Server.
- Works with static and dynamic content.
- User preferences stored on server
  - Accessed via browser cookie validation
Display Classes

- DrawingMapApplet
- DrawingLinesApplet
- DataDelivery
- AppletContainer
DrawingMapApplet

- Primary Pixel Map Display
- Passed Parameters:
  - Offset
  - Date
  - Type
- 6 Included in the Main Display
DrawingLineApplet

- Primary Line Display
- Passed Parameters:
  - Offset
  - Date
- 2 Included in the Main Display
DataDelivery

- Delivers Data to the Display Applets
- Passed Parameters:
  - Date (URL)
- Opens File and Sends Data to a Web Browser/Object
AppletContainer

- Dynamic JSP Webpage
- Uses a Static HTML Base
- Interprets Commands
  - Offset
  - Date
- Starting Location for the Display Subsystem
Presentation Checkpoints

- Design Overview
- User Interface
- System Design
- NIRSS Prototype
- Summary & End
Prototype
Summary

- **Design Overview**
  - Background
  - The Problem
  - The Solution
- **User Interface**
  - NIRSS Display
  - Configuration File
- **Software Design**
  - Hardware
  - Current Design
  - Proposed Design
  - Classes/Components
- **Prototype**
NIRSS Upgrade Design

- Operates in real-time
- ANSI C Fusion Core
  - Improved memory management
  - Cleaner implementation
  - Enhanced configuration options
- Java applet for display
  - Easy-to-interpret data
  - Network accessible