AspenQt

Rose-Gaëlle Belinga
Outline

A. Greetings & Introduction
B. Background: Upsonde, dropsonde, sounding
C. Presentation:
   1. What is Aspen?
   2. Motivation
   3. Objectives
   4. Timeline
      a) Analysis
      b) Design & Documentation
      c) Code & Test
D. Demo
E. Q & A
What is Aspen?

The Atmospheric Sounding Processing Environment (ASPEN) is used for analysis and quality control (QC) of sounding data.

It has the following capabilities:

- Automatically apply quality control procedures to the sounding data
- Present data in tabular and graphical forms
- Automatically determine levels and code them in WMO message formats
- Transmit the WMO messages to other systems
- Save the raw and derived data products in various formats
Motivation

– The Microsoft Foundation Class (MFC) framework lacks many features needed for the development of Aspen

– Aspen currently runs on Windows machines only. AspenQt is compatible with Windows, Linux, and Mac platforms

– Aspen will become an open-source project in the near future, and most developers do not want to be constrained by Microsoft rules
Objectives

• Design the new AspenQt code graphical user interface

• Implement the cross-platform migration of the Aspen

• Develop a test suite and automated test procedures

• Create the project documentation
Part 1: Analysis

- Going through the current AspenMFC code and understanding the logic behind
- Learning how to use various software tools required for this project:
  - Red Hat Linux Enterprise (RHEL5)
  - Qt4
  - Eclipse
  - Python
  - SCons
  - Doxygen
  - Subversion
  - More...
Part 2: Design & Documentation

- UML diagram (use cases and sequence diagrams) designs using MagicDraw

- Graphical user interface designs using Qt4

- Online research by browsing through the Qt forums and manuals on the Safari Books Online website

- Code and design documentation using the reverse engineering features in Doxygen and MagicDraw

- Design patterns such as the MVC, Encapsulation, and Factory used to create a good code skeleton
Use Case: "Start Aspen application"

1. [double] clicks
2. calls
3. creates object and launches process
4. show()
5. GUI is displayed on the screen
### Launch Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Reported</th>
<th>Ignore</th>
<th>Override</th>
<th>Clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of Drop Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure (mb)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature (deg C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RH (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wind Speed (m/s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wind Direction (deg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude (deg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude (deg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altitude (m)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Launch Time**

- **Height Overrides**
  - HI? Surface?
  - Set Heights Missing?
  - Surface Altitude Unknown (Dropsonde over land)

- **Dropsonde Height Integration Results**
  - **Upward**
    - Launch Altitude (m)
  - **Downward**
    - Low Altitude (m)
#include <AspenMainWindow.h>

List of all members.

Public Member Functions

AspenMainWindow (QMainWindow *parent=0)
Constructor using a QMainWindow as main container.

virtual ~AspenMainWindow ()
Destructor.

Protected Slots

void open (void)
Function to open an existing sounding file using the built-in openFileDialog.

void openFile (QString fileName)

void openRecentFile (void)

void configuration (void)
Function to allow the user to modify the configuration file prior opening any sounding.
Diagram: Aspen RT Skeleton Page 1

MVC uses to describe models & views to help reduce architecture complexity. View renders code flexible and reuse.

1. User interacts with interface.
2. Controller handles the input event from the UI and a specific event handler.
3. Controller modifies model of user action, possibly resulting in a change of the model's state.
4. View uses model to generate.
5. UI needs for further interaction which initiates the cycle.

Date: 06/03/09
package Data

AspenConfigMgt

RegistryXML

AspenConfigSet

AspenAdvConfig

AspenQcConfig

AspenOtherOptions

AspenWidgetValidator
Part 3: Code & Test

• The behavior of the user interface was managed by controller classes written in C++

• This behind-the-scene section was the most vital and complex part of the project since it connected the actual Aspen logic to its ‘views’

• Class refactoring and widget validation were often used to accomplish this goal

• Frequent testing executed manually, usually by other interns
NCAR Experience

• Enjoyed this wonderful opportunity to visit Boulder, CO

• Experienced team work from different perspective

• Learned to communicate with engineers having various backgrounds using “non-technical” terms, especially during weekly meetings

• Observed and contributed to the life cycle of a software

• Was introduced to various software tools

• Learned to play Hacky Sack, Ultimate Frisbee with other interns while enjoying hiking and camping
Demo
Q&A