IOWA STATE UNIVERSITY

Software Engineering

Engine Data Analysis Tool

sdmay20-06

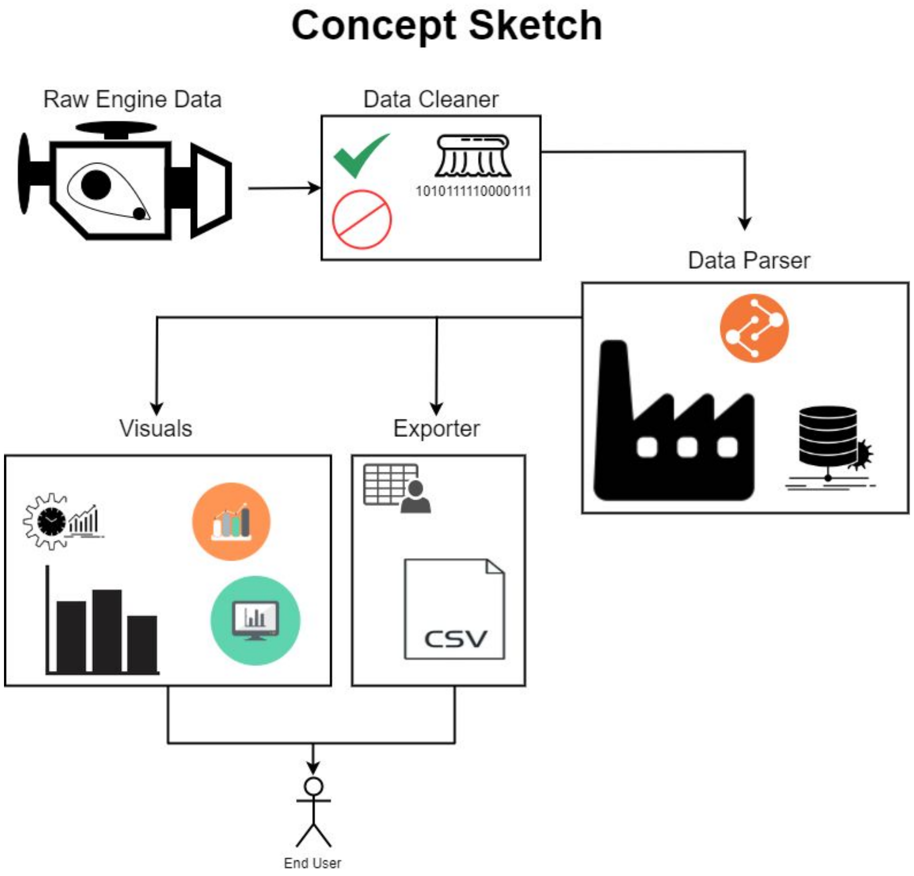
Zachary Frisvold, Will Sartin, Thomas Haddy, John Powen, Ryan Radomski

Team Site: http://sdmay20-06.sd.ece.iastate.edu/

Problem Statement

- To create a program to parse a .csv log file from the center cluster of a C-130 Airplane and show the data in a user-friendly format.

Conceptual Sketch



Functional Requirements

- Windows compatible executable file
- No internet connection necessary
- Capability to create user defined graphs, charts, and tables
- Convert the ASCII data into ARINC-429
- Parse the input file into selectable data fields
- Report anomalies and provide quick access to visualizations
- Export parsed data to either .csv or .xls
- Allow playback of trend data once imported

Non-Functional Requirements

- Parsing needs to run quickly, with a worst case of 10 minutes
- Accept any .csv and .xls files as input
- Recognize if the .csv input file is valid and provide a notification on failure
- Provide data field names for predefined data as per import data
- Capability for user defined data field names
- Allow filtering what's included in the data set export
- Playback sequences the data points at 1 Hz update rate
- Allow a graphical depiction of the EIDS dials during playback

Technical Considerations

- Runtime
- Ported to Windows
- No visible faults
- High test coverage

What Makes This Project Unique?

- Existing Collins Product:
  - Takes hours to parse data files
  - Not used because of runtime

Risks and Mitigation

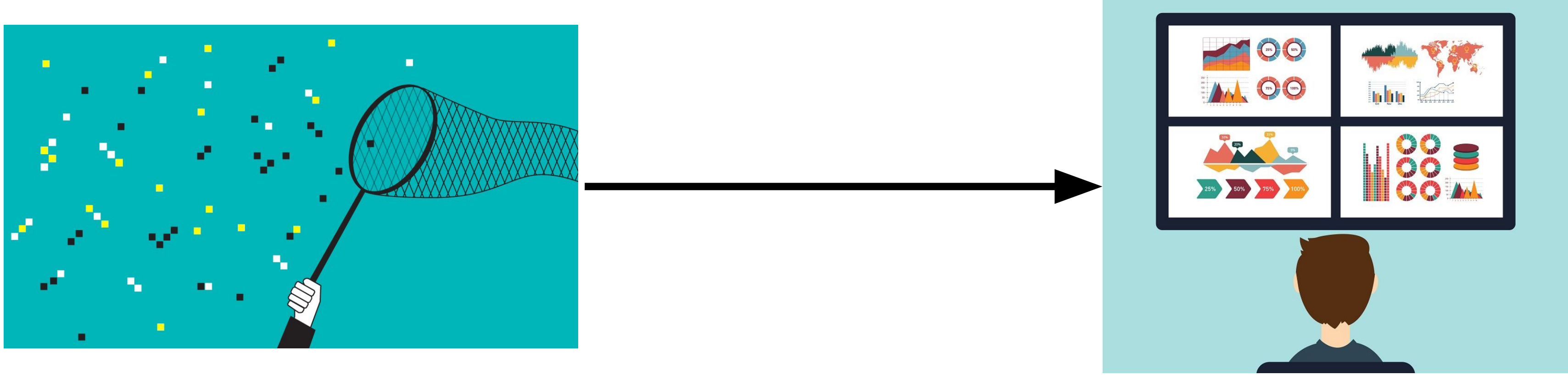
- Feature Creep
  - Timeboxing
  - Prioritization of Requirements
- Wishful Thinking
  - Ninety-Ninety Rule
  - Overestimate, monitoring
- Heroics
  - Accountability

Cost Estimate

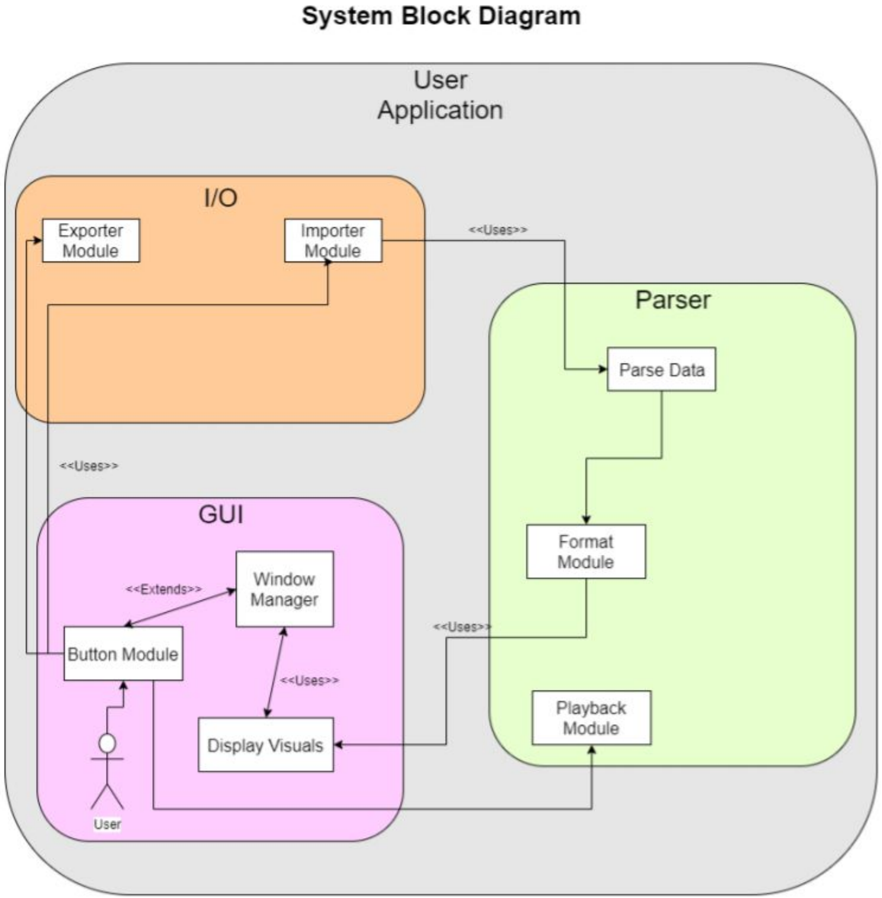
- Create data parser in 1 semester
- Create front end component in 1 semester
  - Agile development practices – will keep adding features through entire semester

Project Milestones

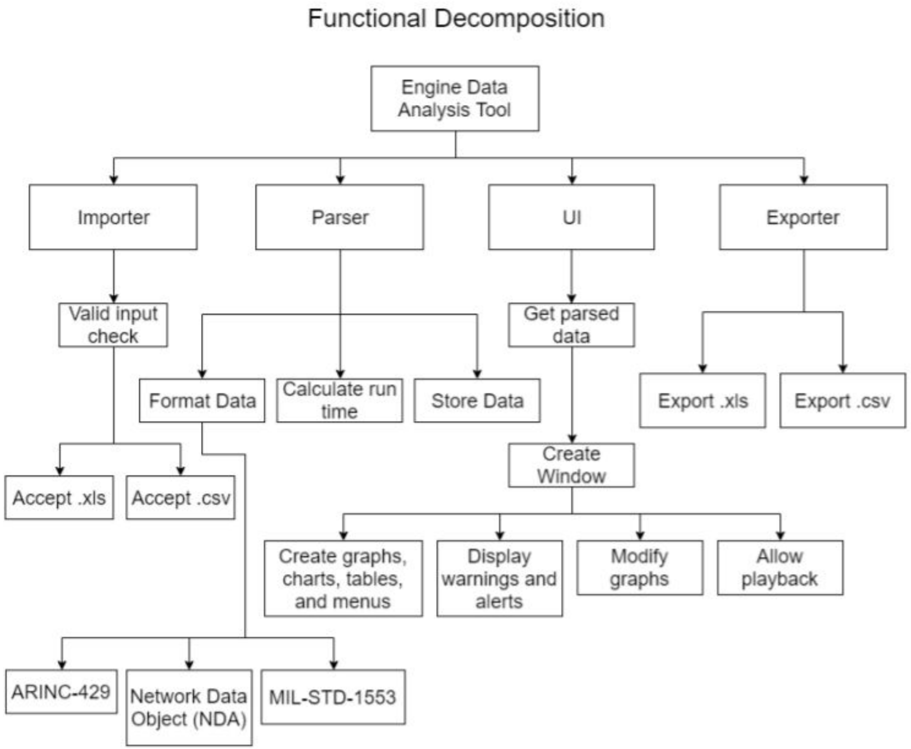
- Parse data into data structure
- Show data in user friendly manner
- Allow for user configurable visuals
- Stretch goals



Design

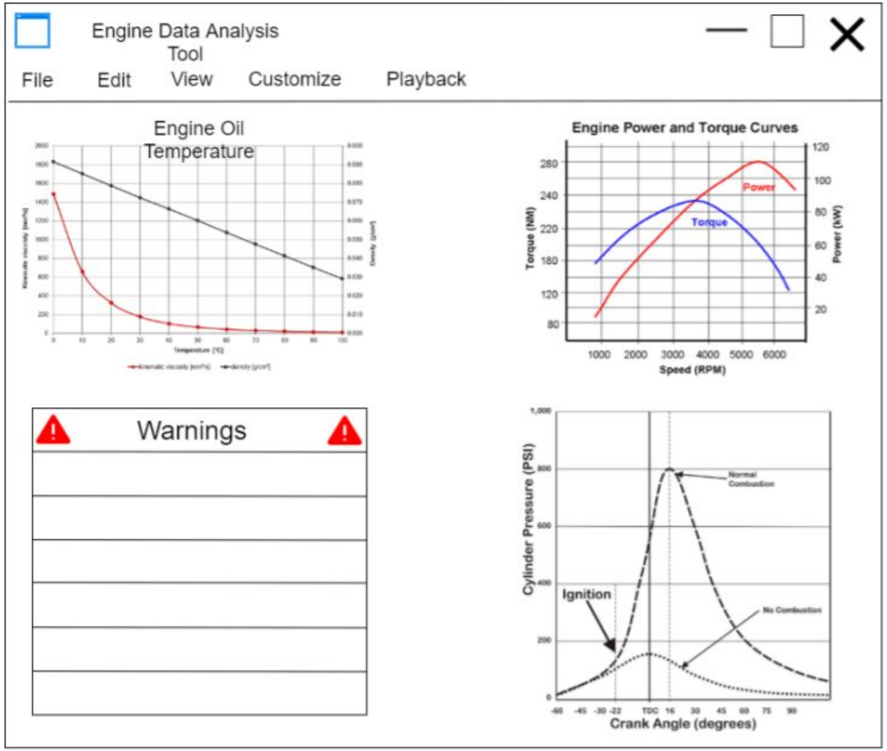

System Block Diagram
User Application
I/O
Exporter Module
Importer Module
<<Uses>>
Parser
Parse Data
Format Module
Playback Module
<<Uses>>
GUI
Window Manager
<<Extends>>
Button Module
<<Uses>>
Display Visuals
<<Uses>>
User

Functional Decomposition



Design

UI Example

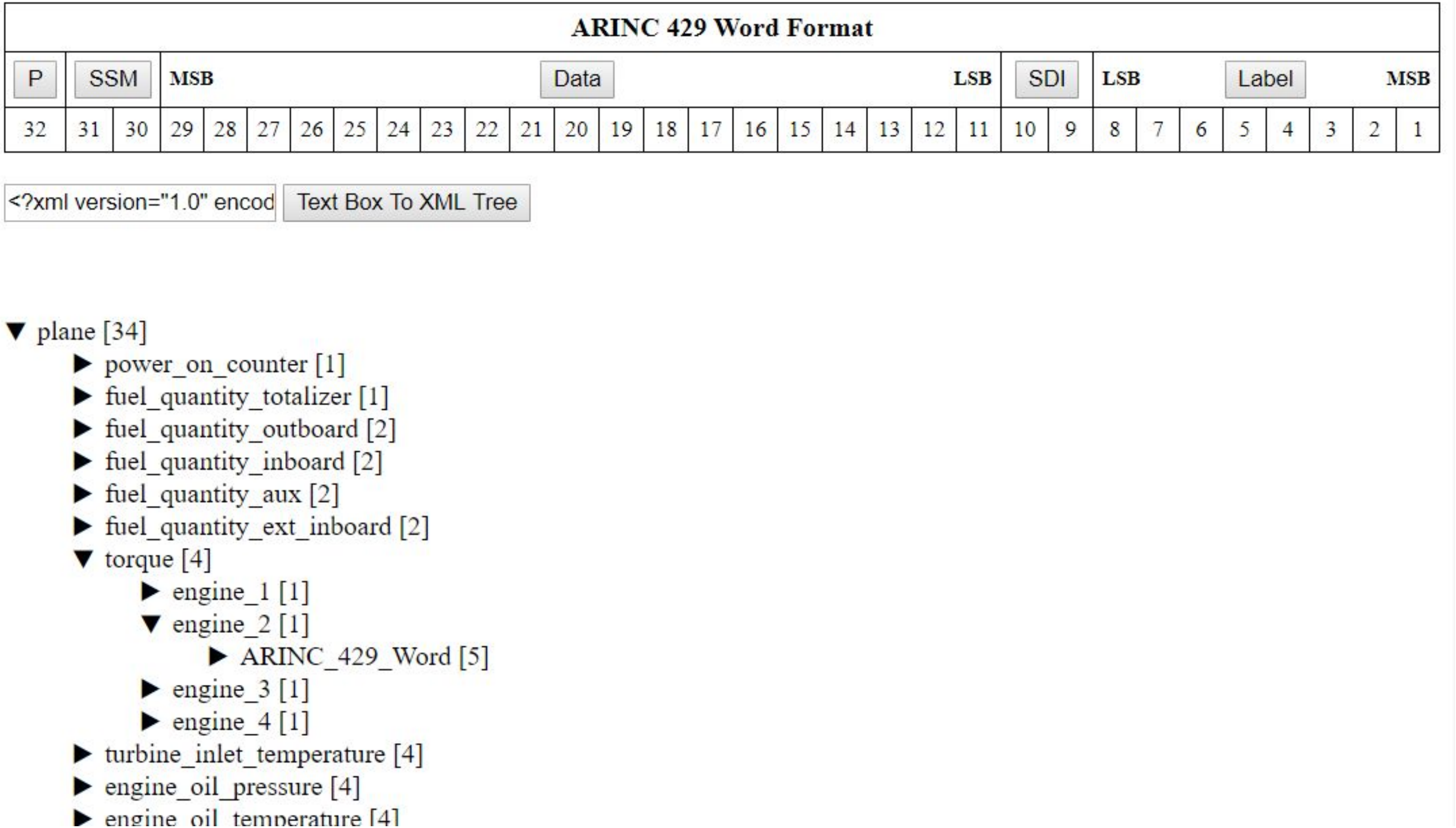


Technology Platforms Used

- C++ - Parsing Engine data in timely fashion
- C# - Creating the GUI and charting logic
- WPF - Platform for Windows Desktop apps

Test Plan

- Automated Test Generation for GUI
  - Selenium/Cucumber
- Fuzzy Test



ARINC 429 Word Format

P	SSM		MSB								Data								LSB	SDI		LSB			Label				MSB
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

<?xml version="1.0" encod

▼ plane [34]
- ▶ power_on_counter [1]
- ▶ fuel_quantity_totalizer [1]
- ▶ fuel_quantity_outboard [2]
- ▶ fuel_quantity_inboard [2]
- ▶ fuel_quantity_aux [2]
- ▶ fuel_quantity_ext_inboard [2]
- ▼ torque [4]
  - ▶ engine_1 [1]
  - ▼ engine_2 [1]
    - ▶ ARINC_429_Word [5]
  - ▶ engine_3 [1]
  - ▶ engine_4 [1]
- ▶ turbine_inlet_temperature [4]
- ▶ engine_oil_pressure [4]
- ▶ engine_oil_temperature [4]

Current Status

- Engine Parser prototype complete
- Technology stack for desktop selected
- Requirements documents done
- Design document complete

Team Contributions

Zak: Communication Lead

Will: Meeting Facilitator

Thomas: UI/UX Architect

Ryan: Quality Assurance engineer

John: Scrum master

Plan for Next Semester

- Call parser from C#
- GUI
- Base graphing tooling
- Custom graph creation tool