Computer Science
&ENGINEERING

State-Based Plot Coloring

Revised Specification and Design

Haoxuan Lin, Myeongwan Beom, Zachary Young

Team #36
Instructors: Dr. Sergiu Dascalu,
Prof. Devrin Lee
External Advisor: Mr. Eric Gilchrist
February 23th, 2018

Table of Contents

1. Abstract

Our project is a State-based Plot Coloring which is designed to help users monitor their condition of machines. It allows users to see their condition of machines into four different states which are a rolling state, a startup state, a running state, and a shutdown state with different colors. It makes this project to be unique because it can offer users convenience way of observation for the condition of their machines in four different states. Each state represents a specific machine state, so it can possibly help people to have less failure of their machine since the user might be able to take an action on the problems after observing the graph. In conclusion, the objective of the project is intended to be designed to let the users be able to observe what states their machines are in and check whether their machines have problems. The project is implemented in C# and Oxyplot for the ease of development and better performance for the User interface.

2. Recent Project Changes

There is no recent change since Project Assignment 1 in CS 426.

3. Updated Specification

- **Summary of Changes**

Below are the updated requirements and use cases. Sections highlighted in yellow represent changes made, sections in light blue represent new additions, and sections in gray represent removed sections. For functional requirements the waterfall plot was changed to a cascade plot because the team needs to use it instead of waterfall for the project. A requirement added was allowing the user to select a machine to only see its plot because the sponsor requested that as a requirement. For use cases waterfall was changed to cascade. Three new use cases were added to satisfy new requirements and to add better user experience. Two use cases and NFR01 were removed because instead of showing the data being plotted one point at a time the plot will just be completely displayed.

- **Functional Requirements**

Requirement ID	Requirement Priority	Requirement Description
FR01	1	The system shall read in data from sample machine data.
FR02	1	The system shall plot the data using a Bode Plot

FR03	1	The system shall plot the data using a Cascade Plot
FR04	1	The system shall plot the data using a Trend Plot
FR05	1	The system shall color the different plots based on the state of the machine.
FR06	1	The system shall allow the value ranges for each the states to be defined by the default values for the machine.
FR07	1	The system shall allow the value ranges for each the states to be defined by the user.
FR08	1	The system shall handle edge cases where the machine is in two or more states at the same time.
FR09	1	The system shall allow the user to select a machine to view in the plot when multiple machines are present.
FR10	1	The system shall allow the user to switch between the different plots.
FR11	2	The system should allow the user to display multiple plots at once.
FR12	2	The system should allow the user to define their own states that they want to monitor.
FR13	2	The system should allow the user to choose their own colors for each state.
FR14	2	The system should allow the user to save a plot of data including relevant user entered states, values, and colors.
FR15	1	The system might read in real machine data.

- **Non-Functional requirements**

Requirement ID	Requirement Description
NFR01	The plots shall be displayed in real time as data is read in.
NFR02	The system shall be implemented using C#.
NFR03	The system shall run on Windows7/8/10 operating systems.
NFR04	The user interface shall be easy to understand and use.
NFR05	The user shall be able to select their colors using a traditional color wheel.
NFR06	The system shall be usable on a web based platform.
NFR07	The plots shall be plotted in such a way that is easy to understand and distinguish between the states.

- **Use Case Diagram**

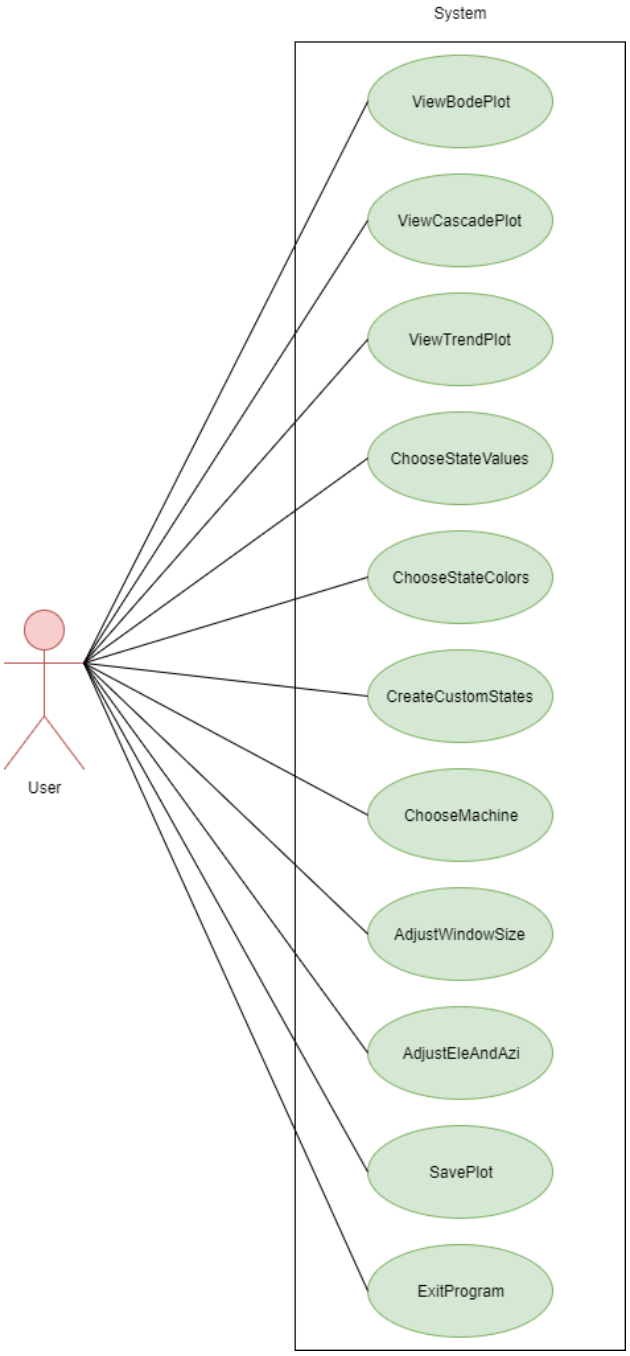


Figure 1: Use case diagram of State Based Plot Coloring system.

- **Use Cases**

Use Case ID	Use Case Name	Use Case Description
UC01	ViewBodePlot	The user selects the Bode Plot to view which results in the plot being displayed on screen.
UC02	ViewCascadePlot	The user selects the Cascade Plot to view which results in the plot being displayed on screen.
UC03	ViewTrendPlot	The user selects the Trend Plot to view which results in the plot being displayed on screen.
UC04	ChooseStateValues	The user enters their own values for each state that is either a default state or a state defined by the user. These values are then used when plotting the data.
UC05	ChooseStateColors	The user chooses the colors to represent each default or user defined state which are then used when plotting the data.
UC06	CreateCustomStates	The user can select to use the default states for the machine or define their own custom states that will be saved for future use.
UC07	ChooseMachine	The user selects a machine in one of the three plots to select a specific machine to view rather than viewing all machines.
UC08	AdjustWindowSize	The user adjusts the size of the window to make the window larger or smaller to fit their needs.
UC09	AdjustElevationAndAzimuth	The user adjusts the elevation and azimuth of the cascade plot using two sliders.

UC10	SavePlot	The user can save the current plot to a file for future use after data has been plotted.
UC11	ExitProgram	The user can exit the program and all plots that are not saved will be lost giving the user an option to save before exiting.
UC12	StartPlottingData	The user can choose when to start plotting the data. This begins reading data from the sample machine data and plots it. If FR14 gets implemented then the data will be read from a machine.
UC13	StopPlottingData	The user can stop plotting the data after data has been plotted and then decide what to do from there such as viewing specific plots or saving the plot. This also gets triggered if there is no more data to process.

- **Detailed Use Cases**

Use case: ChooseStateValues
ID: UC04
Brief description: The user enters their own values for each state that is either a default state or a state defined by the user. These values are then used when plotting the data.
Primary actors: User.
Secondary actors: None.
Preconditions: 1. The states must be created by either using the default states or creating custom states.

Main flow:
1. The use case starts when the user clicks on the button to choose the state values.
2. The user enters a low value for each state.
3. The user enters a high value for each state.

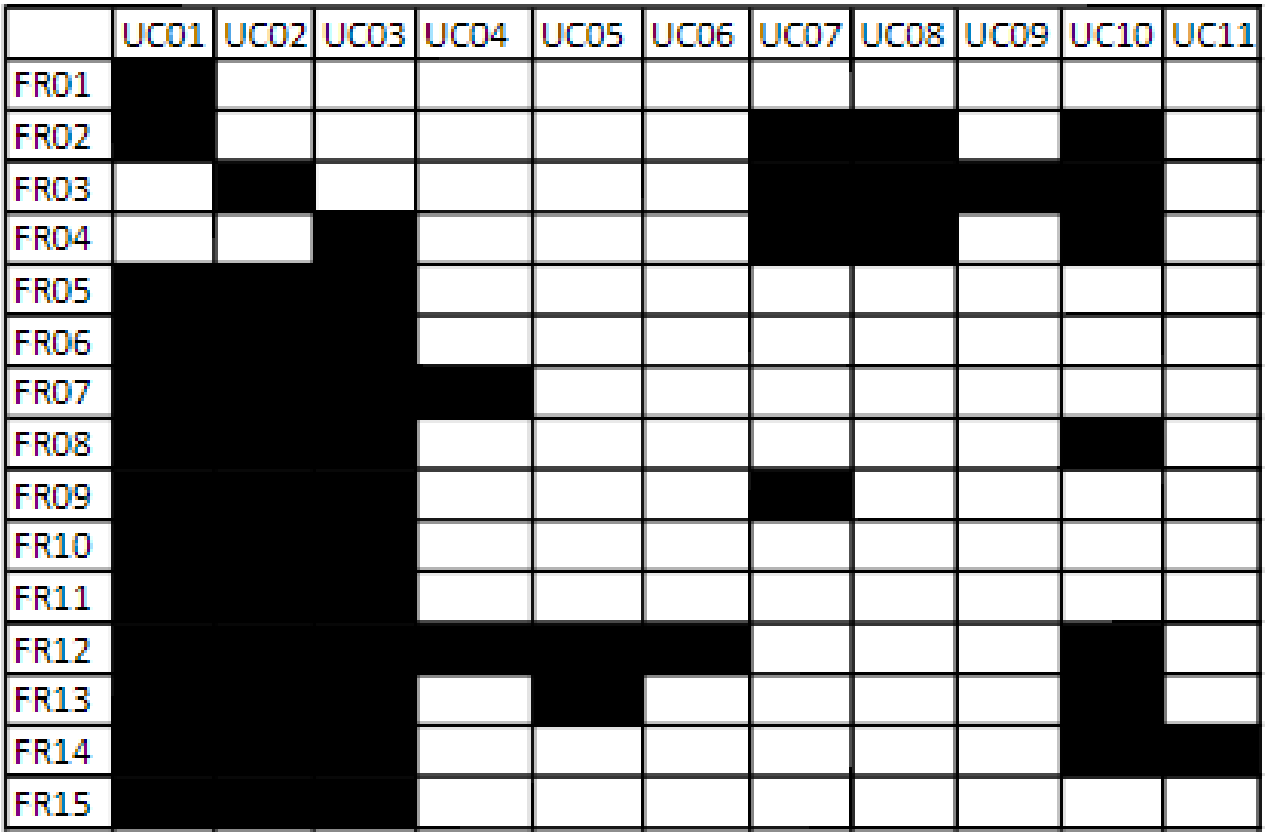
Postconditions:
1. The state values are set and ready to be used for plotting.

Alternative flows:
None.

Use case: AdjustElevationAndAzimuth
ID: UC09
Brief description: The user adjusts the elevation and azimuth of the cascade plot using two sliders.
Primary actors: User.
Secondary actors: None.
Preconditions: 1. A cascade plot is plotted. 2. The user has the cascade plot open on their screen.

Main flow: 1. The use case starts when the user plots the cascade plot. 2. The user can slide the elevation slider up or down to change the elevation. 3. The user can slide the azimuth slider up or down to change the azimuth.
Postconditions: 1. The azimuth and or elevation have changed in the cascade plot.
Alternative flows: None.

- **Requirements Traceability Matrix**



4. Updated Design

4.1 Summary of changes

The teams Context model remained the same with no changes. The class diagram had a complete overhaul as new requirement were made known. Below the new class diagram can be seen with descriptions of some of the core attributes and methods to each of the classes.

4.2 High-level and medium-level design

- **Context Model**

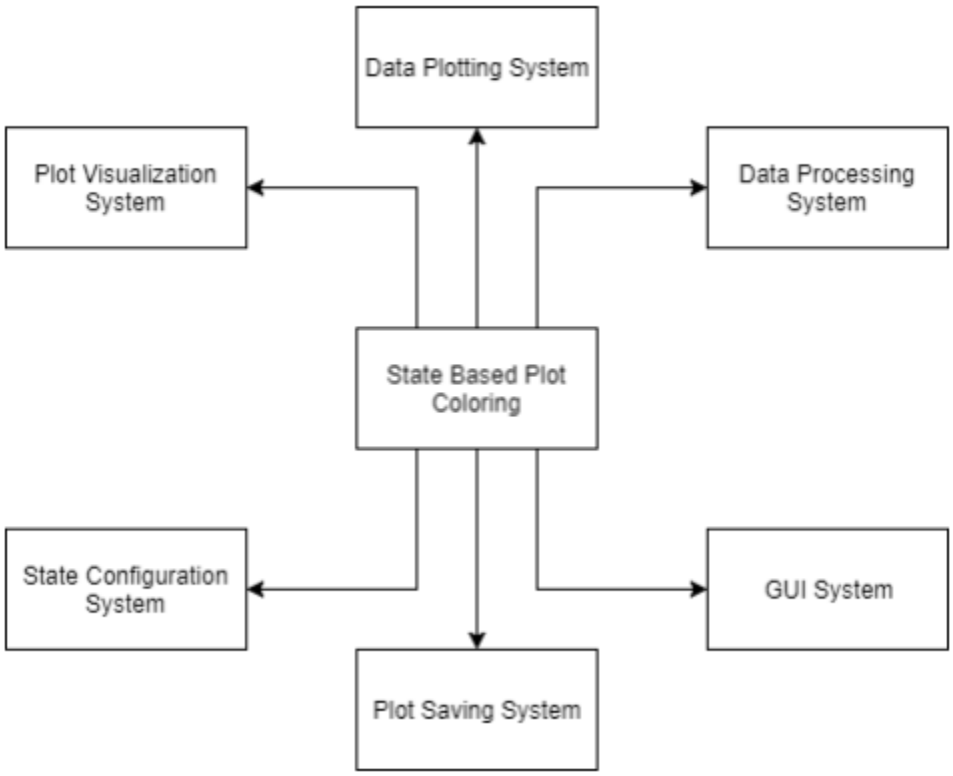


Figure 2: The context model of the State Based Plot Coloring system.

Figure 2 is a context model of the State Based Plot Coloring system. The model depicts many of the subsystems of the system and provides a high level design of the system. The plot visualization subsystem allows for each of the three plots to be visible on screen in the correct format. The data processing subsystem will handle reading in the data and converting the data into values that can be used by the Plot Visualization subsystem. The state configuration subsystem will configure the states to be used for the plots by either machine given or user given values and states. The plot saving subsystem handles saving the current plot and the states and colors associated with the plot. The data plotting subsystem handles plotting the data points to each plot. The GUI system handles the User Interface that the users will be interacting with.

- **Class Diagram**

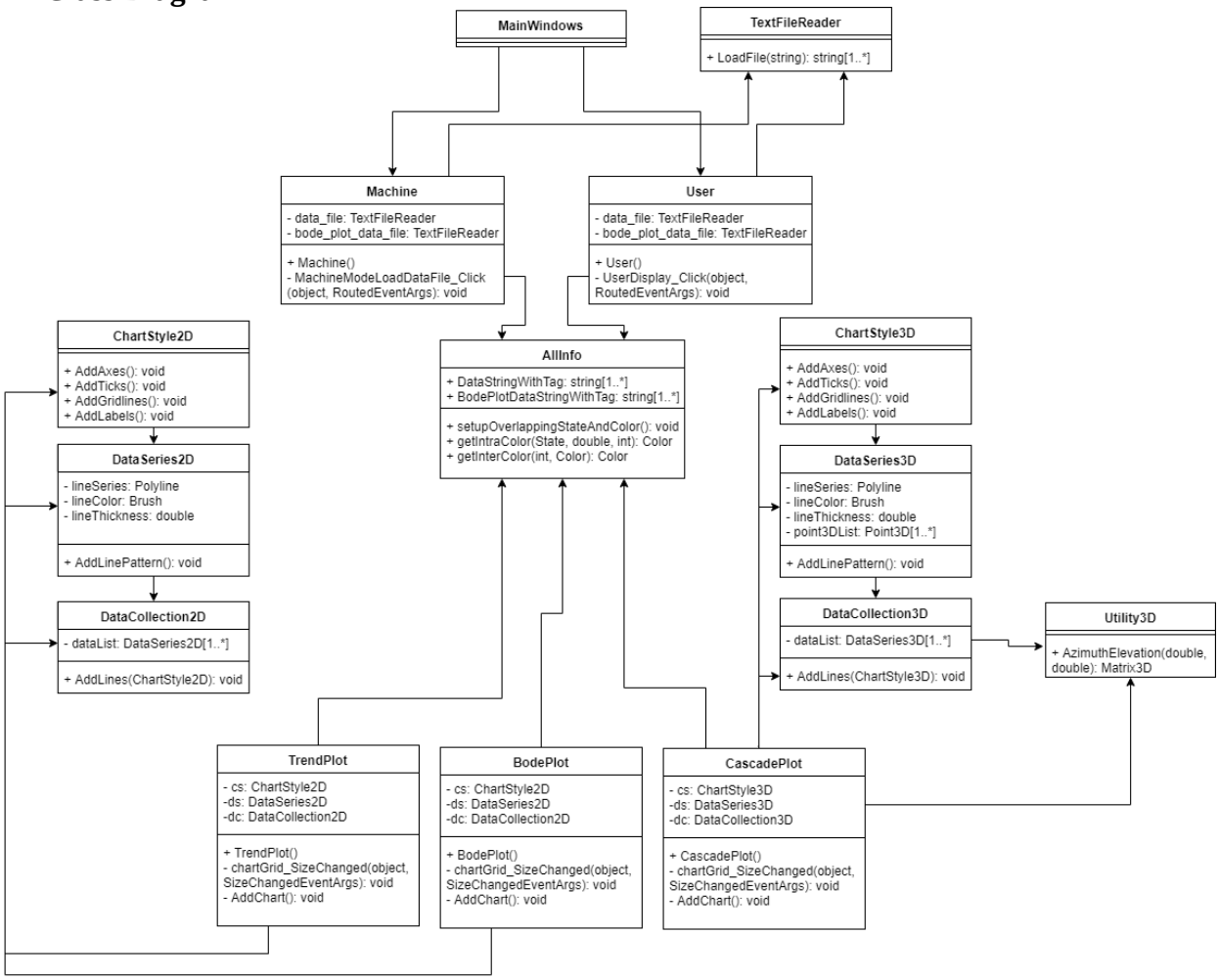


Figure 3: The class diagram of the State Base Plot Coloring System.

- **Class Descriptions**

Class: MainWindows	Initializes the main window of the program.

Class: TextFileReader	Used to read data from the file and store it in an array.
LoadFile()	Function to read in the data from the file, store it in an array, and return it.

Class: Machine	Initializes the window for the machine defined states and handles interactions with the window.

data_file	TextFileReader variable for reading in data from the file.
bode_plot_data_file	TextFileReader variable for reading in bode plot data from the file.
Machine()	Constructor to initialize the class.
MachineModeLoadDataFile_Click()	Loads the data file when the button is clicked.

Class: User	Initializes the window for the user defined states and handles interactions with the window.
data_file	TextFileReader variable for reading in data from the file.
bode_plot_data_file	TextFileReader variable for reading in bode plot data from the file.
User()	Constructor to initialize the class.
UserDisplay_Click()	Loads the data file when the button is clicked.

Class: AllInfo	Share necessary information between the trend plot, the cascade plot, and the bode plot to maintain the consistency.
DataStringWithTag	The string of data and the associated machine.
BodePlotDataStringWithTag	Tne bode plot data string and the associated machine.
setupOverlappingStateAndColor()	Set up overlapping states and colors, which will be called after user clicks "display" button.
getIntraColor()	Return the color of the polyline based on the current rpm and the state that a machine was in. It uses the tag to tell which machine does the data come from. It can be used to handle the overlap between states.
getInterColor()	Return a new color based on machine's tag and polyline's current color. It is used to handle the overlap between machines.

Class: ChartStyle2D	Handles the styling of the chart.
AddAxes()	Adds axes to the chart.
AddTicks()	Adds tick marks to the chart.

AddGridLines	Adds gridlines to the chart.
AddLabels	Adds labels to the chart.

Class: ChartStyle3D	Handles the styling of the chart.
AddAxes()	Adds axes to the chart.
AddTicks()	Adds tick marks to the chart.
AddGridLines	Adds gridlines to the chart.
AddLabels	Adds labels to the chart.

Class: DataSeries2D	Stores a list of polylines.
lineSeries	Variable for a polyline.
lineColor	Variable for the color of the line.
lineThickness	Variable for the thickness of the lines.
AddLinePattern()	Adds a line pattern with the given variables.

Class: DataSeries3D	Stores a list of 3D points.
lineSeries	Variable for a polyline.
lineColor	Variable for the color of the line.
lineThickness	Variable for the thickness of the lines.
point3DList	A list of 3D points to plot.
AddLinePattern()	Adds a line pattern with the given variables.

Class: DataCollection2D	Store a list of data series. For each data series, the AddLines function draws 2D polylines on a 2D screen.
dataList	The data series list.
AddLines()	Add lines to the list.

Class: DataCollection3D	Store a list of data series. For each data series, the AddLines

	function draws 3D polylines on a 2D screen.
dataList	The data series list.
AddLines()	Add lines to the list.

Class: TrendPlot	
cs	ChartStyle variable to interact with ChartStyle Class.
ds	DataSeries variable to interact with DataSeries Class.
dc	DataCollection variable to interact with DataCollection Class.
TrendPlot()	Constructor to initialize the class.
chartGrid_SizeChanged()	Update the plot when the window's size changes.
AddChart()	Add a trend chart.

Class: BodePlot	
cs	ChartStyle variable to interact with ChartStyle Class.
ds	DataSeries variable to interact with DataSeries Class.
dc	DataCollection variable to interact with DataCollection Class.
BodePlot()	Constructor to initialize the class.
chartGrid_SizeChanged()	Update the plot when the window's size changes.
AddChart()	Add a bode chart.

Class: CascadePlot	
cs	ChartStyle variable to interact with ChartStyle Class.
ds	DataSeries variable to interact with DataSeries Class.
dc	DataCollection variable to interact with DataCollection Class.
CascadePlot()	Constructor to initialize the class.
chartGrid_SizeChanged()	Update the plot when the window's size changes.
AddChart()	Add a cascade chart.

Utility3D	Define the Azimuth and Elevation view matrix to a 3D parallel projection on a 2D screen.
AzimuthElevation()	Modify the Azimuth and Elevation Matrix.

- **Data Structures**
  - The ChartStyle, DataSeries, and DataCollection classes are used as data structures for the plotting classes.
  - The TextFileReader class is used as a data structure for holding the read in data from a file.
  - The DataSeries class uses Polyline and Brush data structures that contain information used to draw the plots.

5. User Interface

Figure 1 shows the machine mode of State-Based Plot Coloring. It asks the user to load the state file, the data file, and the bode plot data file. After loading all files, the user can display the Trend Plot, the Cascade Plot, and the Bode Plot by clicking the "Display" button.

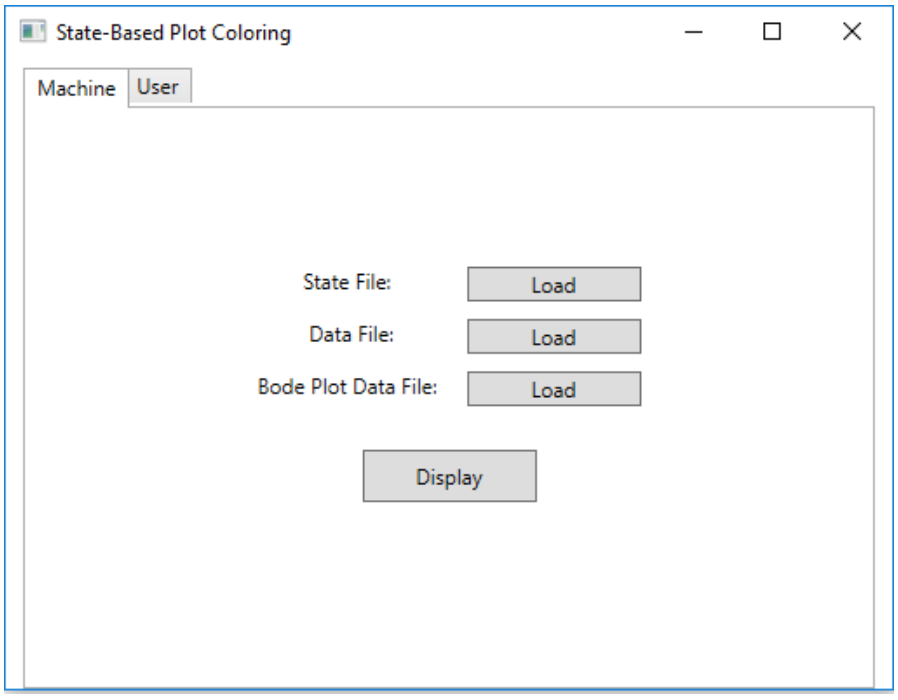


Figure 1: Machine Mode

Figure 2 shows an error message that is generated by State-Based Plot Coloring. In the machine mode, if the user clicks "Display" button without providing the state file, the program asks he or

she to load the state file. If the user clicks “OK” on this error message, the user can continue to load the state file.

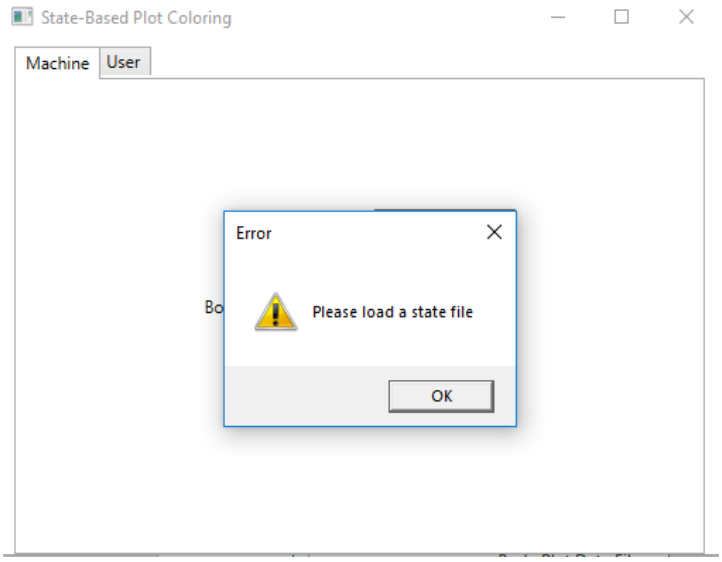


Figure 2: Error Message

Figure 3 shows the load file dialog of State-Based Plot Coloring. If the user clicks “Load” in the menu, the load file dialog will be displayed. The user could choose where he or she wants to load his or her file. If the user clicks “Open” in the load file dialog, the file will be loaded. He or she could close the load file dialog by clicking “Cancel” in the load file dialog.

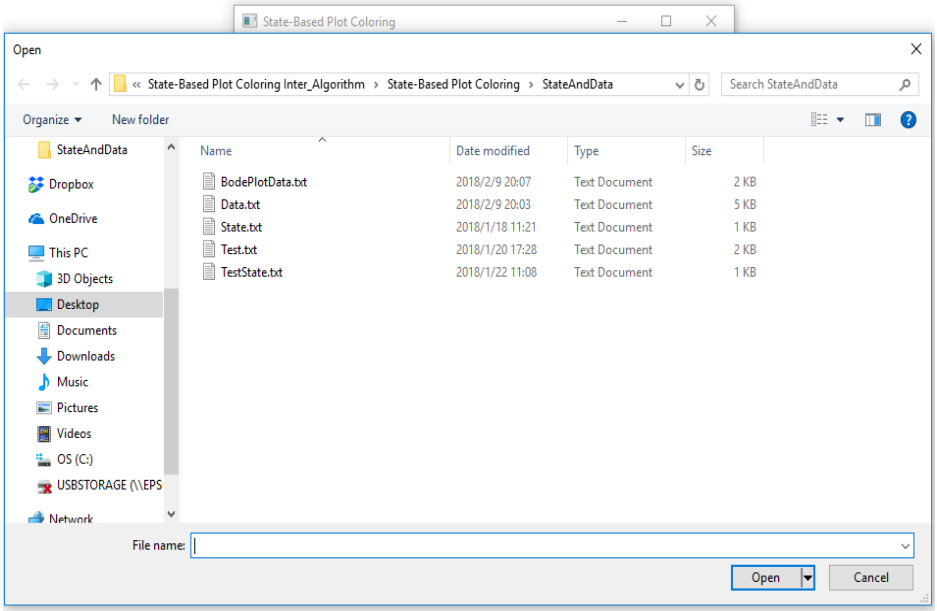


Figure 3: Load File Dialog

Figure 4 shows the user mode of State-Based Plot Coloring. It asks the user to load the data file

and the bode plot data file. The user could define and set the color of each state.

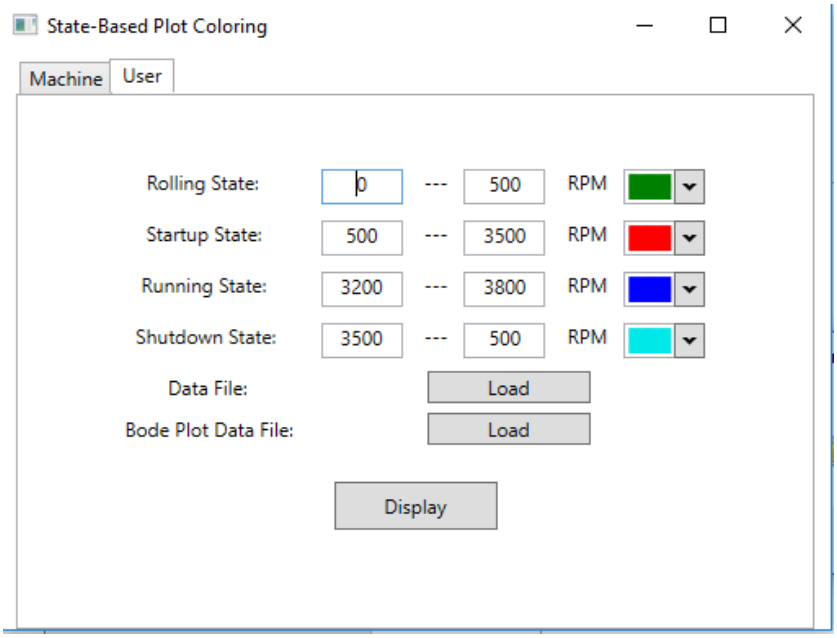


Figure 4: User Mode

Figure 5 shows a warning message that is generated by State-Based Plot Coloring. In the user mode of State-Based Plot Coloring, after the user clicks the "Display" button, if the startup state lower bound is greater than the startup state upper bound, the program will ask the user to check what he or she enters.

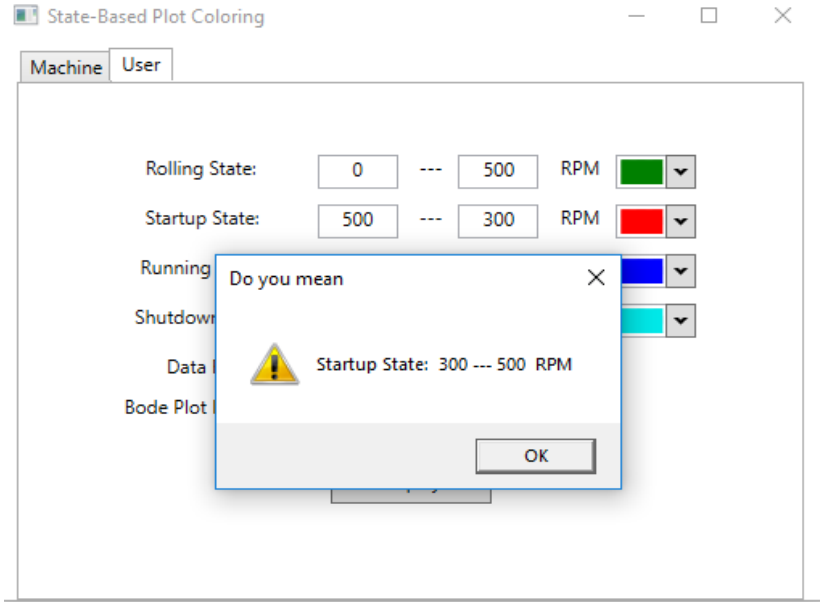


Figure 5: Warning Message

Figure 6 shows the standard color picker of State-Based Plot Coloring. It allows the user to select

the pre-defined colors for each of his or her states.

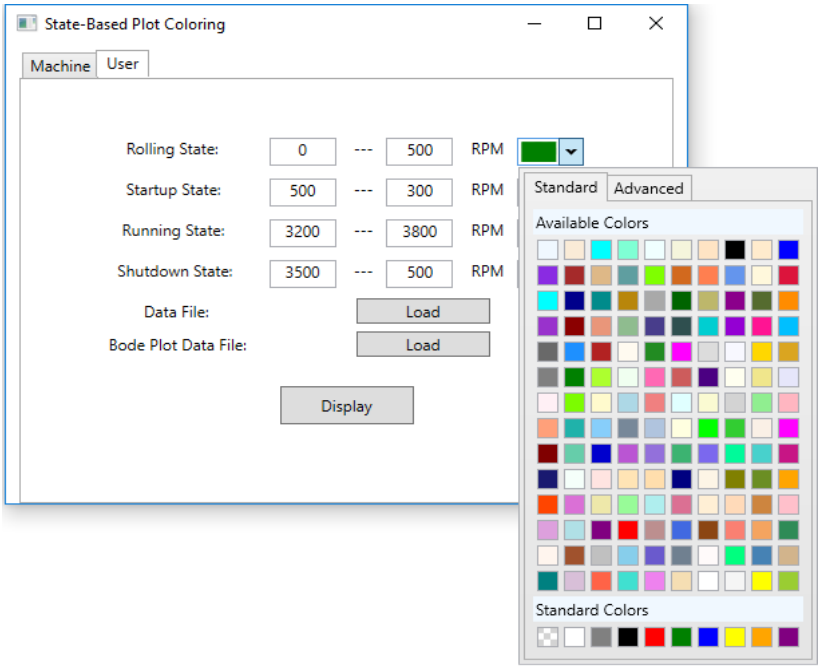


Figure 6: Standard Color Picker

Figure 7 shows the advanced color picker of State-Based Plot Coloring. It allows the user to set the colors to each of their states by choosing the ARGB values he or she wants.

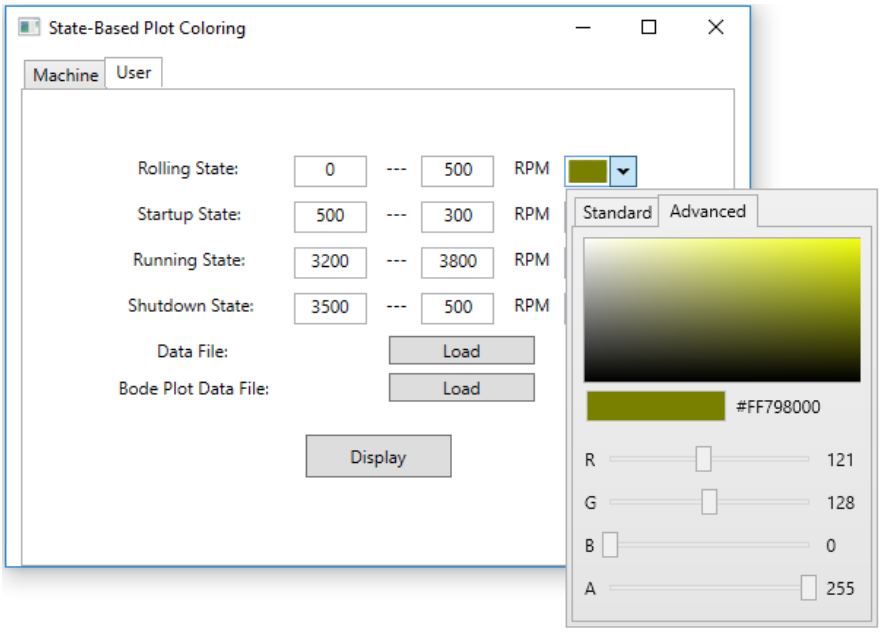


Figure 7: Advanced Color Picker

Figure 8 shows the output of State-Based Plot Coloring. If the user enters all necessary information, the software will display data that the user provides in the Trend Plot, the Cascade Plot, and the

Bode Plot.

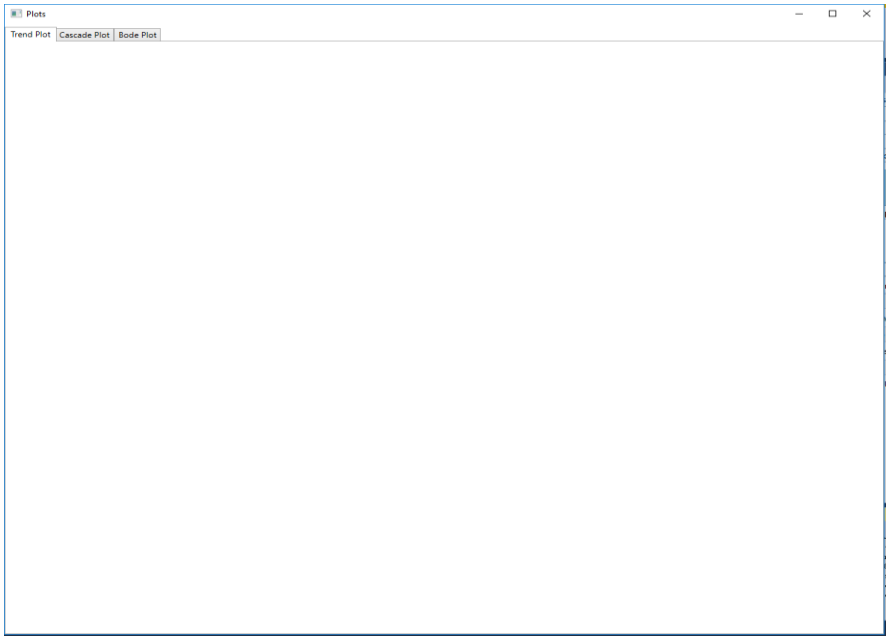


Figure 8: Plots

Figure 9 shows User Interface of the Cascade Plot of State-Based Plot Coloring. It allows the user to adjust the elevation and azimuth of the Cascade Plot.

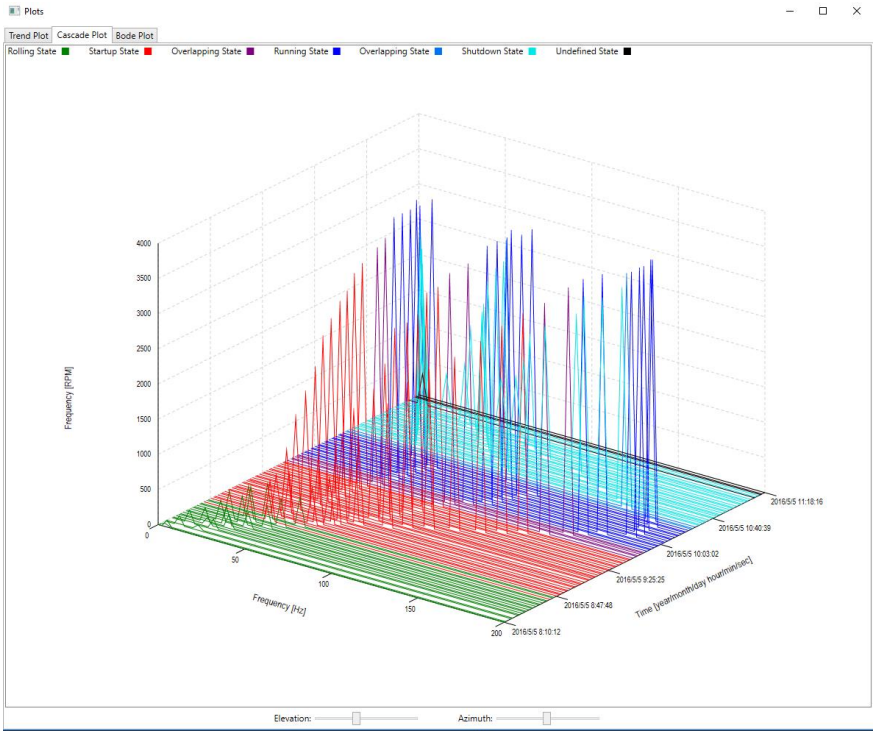


Figure 9: Cascade Plot

6. Glossary of Terms

1. **API (Application Programming Interface):** A set of subroutine definitions, protocols, and tools for developing software.

2. **Bode Plot:** A standard format for plotting frequency response of LTI systems. The plot can be used to interpret how the input affects the output in both magnitude and phase over frequency.

3. **C#**: A strong typing, imperative, and object-oriented programming language developed by Microsoft.

4. **Cascade Plot**: A three-dimensional plot in which multiple curves of data are displayed simultaneously. The Cascade Plot is used to observe frequency or order changes versus rotational speed. A cascade plot consists of a series of spectra acquired at consecutive speeds.

5. **Hit Test**: The process of determining whether a user-controlled cursor intersects a given graphical object drawn on the screen.

6. **Color wheel**: A visual representation of colors arranged according to their chromatic relationship.

7. **Condition Monitoring system**: A system which monitors a parameter of condition in machinery while in operation to identify a significant change which is indicative of a developing fault.

8. **Data Binding in WPF**: A mechanism in WPF that provides an easy way to display and interact with data.

9. **Image processing**: A method to convert an image into digital form and perform some operations on it to extract some useful information from it.

10. **Machine diagnostic system**: A system that identifies the nature and cause for a certain phenomenon. It is used in many disciplines with variation in logic and analytics. It determines the cause and effect of the phenomena, and it is generally used in system engineering and computer science.

11. **Model-View-View Model(MVVP) pattern**: A user interface based design pattern. It provides a clean separation of concerns between the user interface controls and their logic.

12. **OxyPlot**: An open source plotting library for .NET, which is a software framework developed by Microsoft. It is useful for the 2D plot.

13. **Overlap between states**: An overlap between two states. For example, the startup state can be defined as 500-3500 RPM and the running state can be defined as 3200-3800 RPM. We can say there is an overlap between the startup state and running state.

14. **Rolling state**: The state of machines when it is rotated by other means until it reaches its operating speed. It is done to avoid stress development.

15. **Running state:** State of the machine when it is powered up and performing its work.

16. **Startup state**: The state of the machine when it is started for its operation.

17. **Shutdown state:** The state of the machine when it has no memory state and is not performing any tasks.

18. **Trend Plot**: A two-dimensional plot in which describe the trend. It displays information in sequence over time and reveals a general pattern of change.

19. **User Interface**: A visual part of a computer application or operating system with which a person can interact.

20. **Windows Presentation Foundation (WPF):** A presentation system for building Windows client applications. It is released as part of .NET Framework 3.0 in 2006.

7. Engineering Standards and/or Technologies.

1) Engineering Standards

a. Accessibility Standards

Accessibility Standards is to consider designing a product in a way that every people are easier to use, including people with disabilities. We consider the Accessibility Standards and implement our functionalities of visualization in a way that people can choose a color as they want, so it can help people better observe, including people with color blindness.

b. USB

USB is an industry standard which was developed for connection, communication, and power supply between a computer and other devices which have USB ports. We use the USB as temporary backup storage for our project, so when we have data loss, we can restore the data loss. Also, we use USB as a portable device, so we can bring our own works and show the progress of the project in any lab without our own devices.

2) Engineering Technologies

a. C #

C # is a programming language which supports structured programming, lexical variable scope, and recursion. We use C # programming language to develop software system for our project. Basically, we use C# to implement data structure like classes for strong data and functionalities of our project.

b. Oxyplot

Oxyplot is a cross-platform library for .NET. It supports different axis and series for a plot, and it can export the plot to file formats such as PNG, PDF, and SVG. We use we use Oxyplot to display data in a 2D plot with x-axis and y-axis or 3D plot with the x-axis, y-axis, and z-axis.

c. Visual Studio

Visual Studio is an integrated development environment (IDE) form Microsoft. It is used to develop a computer program, website, web apps, and web services. We use visual studio as IDE for our project, and it is used to utilize the Oxyplot since Visual Studio supports .NET and Oxy plot is libraries for the .NET.

d. Microsoft Window

Microsoft Window is a group of several graphical operating system. We use Window operating systems to work with the Visual Studio since it is not skill and labor intensive for a developer.

8. References

8.a Articles

Alexander Ypma. Learning methods for machine vibration analysis. 12 November 2001. <http://rduin.nl/papers/thesis_01_ypma.pdf>

This research describes the methods that can be used for machine vibration analysis and machine health monitoring. It provides readers with a lot of examples to help them understand the learning approach to machinery health monitoring. It helps us to understand why our project is important for monitoring the machine health.

Chockalingam Aravine Vaithilingam, Gilbert Thio, Rajparthiban Rajkumar. Health monitoring of induction motor for vibration analysis. <http://www.jee.ro/covers/art.php?issue=WU1264295972W4b5ba024a5c9d>

This article explains why condition monitoring system is important. It helps readers understand

the meaning of the data. The condition monitoring system, which is described in this article, plots data in different ways to help users analyze these data. It is important for us to know what the users' need is so that we can plot the data in the way that users can easily analyze these data.

Chen Lu, Yang Wang, Minvydas Raguiskis, Yujie Cheng. Fault Diagnosis for Rotating Machinery: A Method based on Image Processing. 6 October 2016. <http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0164111&type=printable>

This article describes the method based on image processing. Readers can use it to realize automatic feature extraction and fault diagnosis in two-dimensional space. This article goes into details about how to use the diagnosis method based on image recognition to provide a method for condition monitoring and fault diagnosis of rotating machinery.

8.b Problem Domain Book

<https://mehmetakifsonmez.files.wordpress.com/2013/12/wpf-4-unleashed.pdf>

This book demonstrates how to create sophisticated user interface mechanisms, such as Visual Studio–like collapsible/dockable panes. It provides a tour of different controls built into WPF. In addition, it covers the features, such as 2D graphics and 3D graphics in WPF.

8.c Websites

http://www.wpftutorial.net/Home.html

This website describes the Windows Presentation Foundation (WPF). It teaches readers how to use data binding and how to draw a 2D and 3D graph by using C# and the WPF. It also provides readers with books and instruction videos about WPF

http://docs.oxyplot.org/en/latest/index.html

This website describes the OxyPlot, which is a cross-platform plotting library for .NET. It is useful for the 2D plot. It teaches readers how to install and use it. The website lists all API for this library. We may use this library to display data in both the Trend plot and Bode plot since they are considered a 2D plot.

https://www.codeproject.com/Articles/42174/High-performance-WPF-D-Chart

This website describes the high-performance WPF 3D Chart. It teaches readers how to set the camera, light, and 3D model. It also teaches them how to use auto zoom and color the 3D model. This website is useful for us to create the Cascade plot, which is considered a 3D plot.

9. Contributions of Team Members

Haoxuan Lin spent 3 hours contributing to the following:

- Cover Page
- Recent Project Changes
- User Interface Design
- Glossary of Terms
- List of References
- Contributions of Team Members

Myeongwan Beom spent 3 hours contributing to the following:

- Table of Contents
- Abstract
- Engineering Standards and/or Technologies

Zachary Young spent 4 hours contributing to the following:

- Specification
- Design