Boeing Team B CS397 Project Plan

Implementation of Table Based Volumetric Compensation

Alex Bertels, Charles Ortman, Joseph "Gus" Steurer, Kevin Zheng

Introduction

This document lists the classes implemented in the VECTool application and provides a description of what each class does. The variables and functions in each class diagram are implemented using C# .net. As the team progresses through the Coding and Testing phase, we expect that the document will evolve throughout our implementation as we encounter new challenges presented during development.

Classes and Descriptions

VECState
+rawLongShortTool: Dictionary<integer, List<double> > +MALongShortTool: Dictionary<integer, List<double> > +CALongShortTool: Dictionary<integer, List<double> > -machineConfiguration: const integer -controllerProfile: const integer -toolLength: const double -toolOffset: const double +currentStep: integer
+ToolLength(): double +ToolOffset(): double +ControllerProfile(): integer +MachineConfiguration: integer

VECState represents data that the VEC application will utilize throughout its run cycle. Each phase reads and writes to this class. The GUI/user input portion of the software will populate **rawLongShortTool, machineConfiguration, controllerProfile, toolLength, and toolOffset**. The accessors listed allow us to make some variables constant. During measurement alignment, **MALongShortTool** will be populated, followed by command alignment where **CALongShortTool** will be populated. The **VECState** data will then be used to parse the GA and ILM DLLs. Finally**, VECState** is parsed during the automatic report generation. This class satisfies FR4.

VECGUI :: Form
+ vecState: VECState + WinFormObjects: Form
- VECGUI(): void - runErrorCalc(): void + updateProgress(float): void - completeScreen(): void + getVECState() : VECState + setVECState(): VECState

VECGUI is the class that instantiates and maintains the program GUI throughout its operation. It contains a pointer to the **VECState** input data structure and GUI objects**. VECState** can be called and set through its respective get/set methods. The **VECGUI** constructor will instantiate all components of the GUI, including the association between GUI objects and their associated fields in **VECState**. The **runErrorCalc** method will be called when the user selects the 'Run' button on the GUI to begin the process of calculating the measurement error. As the later steps of the program progress, they will call back to the VECGUI class through the **updateProgress** method to update the state of the progress bar. When the program has finished, the **completeScreen** method will be called to instantiate the 'Finished' screen of the GUI. This class satisfies FR1, FR2, FR3 and QR1.

ToolMeasurementHandler
- state: VECState - transformationScript: MATLABDLL
+ ToolMeasurementHandler(VECState) + alignMeasurements(): VECState - alignmentErrorCheck(): boolean + transformMeasurements(): VECState - transformErrorCheck(): boolean

The **ToolMeasurementHandler** class is responsible for tool measurement alignment and transformation as well as all error handling for these steps. Prior to executing the **alignMeasurements()** function, the tool data must be parsed and stored in the **rawLongShortTool** variable in the corresponding **VECState** provided during construction of the **ToolMeasurementHandler.** The **VECState** will be stored in the *state* variable. During the execution of **alignMeasurements(),** the raw measurements will be scanned for any misalignments and then values will be removed or adjusted. Once the modifications are finished, the **alignmentErrorCheck()** is completed. If there is a problem, an error will be thrown back to the user interface asking for either new or more data. If the error check returns true and the data has been aligned, then the **transformMeasurements***()* function can be executed and will use the known MATLAB script (**transformationScript**) to transform the corrected tool measurements. The transformed data will be stored in the **MALongShortTool** variable that belongs to the *state* variable. If the **transformErrorCheck()** detects an issue after the MATLAB script is called, then an error will be thrown to the user interface asking for either more or new tool measurements. After the execution of **transformMeasurements(),** the commands may then be transformed. This class satisfies FR5, FR6, FR7, and FR8

ToolCommandHandler
- state: VECState - GAScript: MATLABDLL - ILMScript: MATLABDLL
+ CommandMeasurement(VECState) + alignCommands(): VECState - commandErrorCheck(): boolean

The **ToolCommandHandler** class is used to check the values of the command measurements before passing them onto the Genetic Algorithms and Implicit Loop Method script. The inputs that this class requires are the long and short tool transformations done by the **ToolMeasurementHandler** and the original long and short tool measurements along with the controller profile. All of these inputs are retrieved from the **VECState**. **AlignCommands()** will then be run to check for errors within the new command measurements by comparing them to the original measurements. If there are errors, a few of the outlier data points will be deleted to check if the error can be minimized; otherwise, functionality will be provided to either return to the previous state, prompt user for new/more data and/or exit the program prematurely. If there are no errors, then the class will proceed to pass the command measurements to the **GAScript** and **ILMScript**, and also the controller profile to the **GAScript**. Furthermore, this class will serve as a staging point for more expansion should the need arises for more calculations are this stage. This class satisfies FR9, FR10, FR11, and FR12.

ReportGenerator
- state: VECState
+ FormatCompensationTables(ControllerType) +GenerateILMReport +GenerateGAReport

The report generator translates the raw output of the ILM and GA DLLs into reports and compensation tables. The ILM and GA DLL will write output to disk and each of **ReportGenerator's** functions will open a file handler to read this data. **FormatCompensationTables** will read the unformatted compensation tables, format them according to a controller type, and write the formatted tables to a user specified directory. **GenerateILMReport** and **GenerateGAReport** will read the raw output of the ILM and GA DLL and write to an automatically generated Microsoft Word Document. The performance report will contain a goodness of fit value, calculation time, mean and maximum residual error, and plots of the identified table function. The GA report will contain the solutions to the genetic algorithm, graph of errors, performance statistics, calculation time, chi squared value, and max/mean residual error. Plots will be automatically generated and included in the report. The report will be written to a user specified directory. This class satisfies FR13, FR14, and FR15.

UML Diagram

The following UML diagram describes the relationships between classes. There is a one to one association between all classes, indicating only one instance of each class during runtime. The classes associated with VECGUI access VECState through the instance that VECGUI creates. Variables and functions are omitted from the diagram to make it more readable.