Boeing Team B CS397 Project Plan

Implementation of Table Based Volumetric Compensation

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

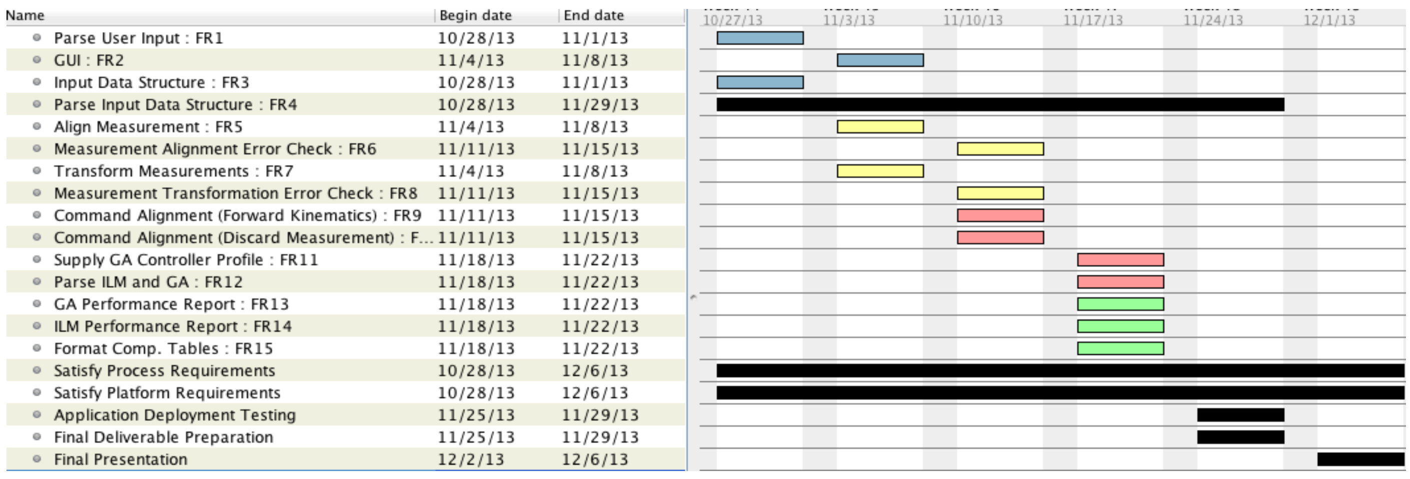
Introduction

The project requirements are decomposed into steps necessary to implement each requirement. Each task is assigned to a developer. The project size is measured in the estimated amount of time (in hours) required to finish each step/requirement. The requirements are listed in sequential order, indicating that one requirement listed before another must either be implemented or in progress. Some tasks may be started concurrently by team members, which is indicated in the Timeline Overview and Gantt Chart. Coding and testing will be done in two scenarios. As a group, the team will meet to work on an assigned requirement or assist with another team member's requirement. When group work isn't necessary, effort will be conducted individually. The duration field indicates a range of days in which a specific requirement must be implemented completely, but each team member will begin attempting to implement his set of requirements beginning October 28th.

Timeline Overview - Work Breakdown Structure

Requirement	**Req. ID**	**Developer**	**Time (hours)**	**Duration (Date)**
Parse User Input	FR1	Charles Ortman	5	10/28 - 11/1
GUI	FR2	Charles Ortman	5	11/4-11/8
Input Data Structure	FR3	Charles Ortman	5	10/28-11/1
Parse Input Data Structure	FR4	All	Ongoing	Ongoing
Align Measurements	FR5	Alex Bertels	4	11/4-11/8
Measurement Alignment Error Check	FR6	Alex Bertels	3	11/11-11/15
Transform Measurements	FR7	Alex Bertels	8	11/4-11/8
Measurement Transformation Error Check	FR8	Alex Bertels	2	11/11-11/15
Command Alignment (Forward Kinematics)	FR9	Kevin Zheng	5	11-11/11-15
Command Alignment (Discard misaligned)	FR10	Kevin Zheng	5	11-11/11-15
Supply GA Controller Profile	FR11	Kevin Zheng	1	11/18-11/22
Parse ILM and GA	FR12	Kevin Zheng	4	11/18-11/22
ILM Performance Report	FR13	Joseph Steurer	6	11/18-11/22
GA Performance Report	FR14	Joseph Steurer	7.5	11/18-11/22
Format Comp. Tables	FR15	Joseph Steurer	3.5	11/18-11/22
Add Manual	QR1	Joseph Steurer	6	TBD
Platform Requirements	PR1-PR6	All	Ongoing	Ongoing
Process Requirements	N/A	All	Ongoing	Ongoing

Gantt Chart



Name	Begin date	End date
Parse User Input : FR1	10/28/13	11/1/13
GUI : FR2	11/4/13	11/8/13
Input Data Structure : FR3	10/28/13	11/1/13
Parse Input Data Structure : FR4	10/28/13	11/29/13
Align Measurement : FR5	11/4/13	11/8/13
Measurement Alignment Error Check : FR6	11/11/13	11/15/13
Transform Measurements : FR7	11/4/13	11/8/13
Measurement Transformation Error Check : FR8	11/11/13	11/15/13
Command Alignment (Forward Kinematics) : FR9	11/11/13	11/15/13
Command Alignment (Discard Measurement) : F...	11/11/13	11/15/13
Supply GA Controller Profile : FR11	11/18/13	11/22/13
Parse ILM and GA : FR12	11/18/13	11/22/13
GA Performance Report : FR13	11/18/13	11/22/13
ILM Performance Report : FR14	11/18/13	11/22/13
Format Comp. Tables : FR15	11/18/13	11/22/13
Satisfy Process Requirements	10/28/13	12/6/13
Satisfy Platform Requirements	10/28/13	12/6/13
Application Deployment Testing	11/25/13	11/29/13
Final Deliverable Preparation	11/25/13	11/29/13
Final Presentation	12/2/13	12/6/13

Functional Requirements

FR1: Parse raw short tool and long tool CSV measurement, tool length, tool offset, and command position data into data structures.
[Developers: Charles Ortman — Time: 5 hours]

1. The most optimal data structures will be determined to store the tool measurement, length, offset, and command position data for processing. [Time: 0.5 hours]
2. A class will be created with the data structures determined in Step 1 as private fields with basic get/set functionality. [Time: 1.0 hours]
3. Functions will be written for the input storage class that can parse the raw CSV format of the input files and GUI class input to store in the private fields. [Time: 3.5 hours]

FR2: Provide GUI for selecting a machine configuration.
[Developers: Charles Ortman — Time: 5 hours]

1. A graphic outline will be drafted on paper to map out the intended look of the machine configuration GUI. [Time: 0.5 hours]
2. A class will be coded that utilizes the WinForms .Net API to have the functionality to implement the appearance of the rough draft from Step 1. [Time: 3.0 hours]
3. Multiple functions will be written to associate calls from the GUI (including button presses and text input) with get/set functions from the input storage class. [Time: 1.0 hours]
4. The appearance of the functioning GUI will analyzed and refined to present an interface that achieves a definitive feel of elegance, usability, and professionalism. [Time: 0.5 hours]

FR3: Indicate progress / current step of the application through the GUI.
[Developers: Charles Ortman — Time: 5 hours]

1. A graphic outline will be drafted on paper to map out the intended look of the process tracking GUI. [Time: 0.5 hours]
2. The class written for the machine configuration GUI will be updated with any necessary functionality from the WinForms .Net API to implement the additional graphical needs determined in Step 1. [Time: 3.0 hours]
3. Multiple functions will be written to accept calls from other classes to indicate the progress of other processes in program. These functions will then in turn call the added functionality from Step 2 to display this progress. [Time: 1.0 hours]
4. The functioning process tracking GUI will be refined to be consistent with the appearance of the machine configuration GUI. [Time: 0.5 hours]

FR4: Parse the input data structure to obtain inputs for each stage of the application.
[Developers: All — Time: Ongoing]

1. All modules of the application will access the same data structures for communication.

FR5: If the offset between long and short tool measurements is greater than two to three standard deviations, then attempt to align that offset or remove the measurement.
[Developers: Alex Bertels — Time: 4 hours]

1. Parse the long and short measurements data file. [Time: 0.5 hours]
2. Determine if the offset between the long and short tool measurements is greater than two to three standard deviations or greater than some maximum constant. [Time: 1.5 hours]
3. Attempt to align incorrect offsets or remove unmatched measurements using a brute force technique. [Time: 2 hours]

FR6: If an offset cannot be aligned or too many measurements were not aligned, then throw an error asking for more or new long and short tool measurement data.
[Developers: Alex Bertels — Time: 3 hours]

1. Decide if an offset cannot be aligned or too many measurements were not aligned. [Time: 1.5 hours]
2. If there is a problem, throw an error asking for more or new long and short tool measurement data. [Time: 1.5 hours]

FR7: Using the aligned measurements, parse the Calculate Transformation MATLAB DLL.
[Developers: Alex Bertels — Time: 8 hours]

1. Establish an environment to run MATLAB DLLs. [Time: 1.5 hours]
2. Convert the MATLAB script into a DLL capable of transforming measurements. [Time: 2.5 hours]
3. Using a C# program, pass input to, execute, and read output from the MATLAB DLL automatically. [Time: 4 hours]

FR8: If the transformation is deemed incorrect, assume that more or new long and short tool measurements are needed and throw an error stating so.
[Developers: Alex Bertels — Time: 2 hours]

1. Decide if the transformation was incorrect. [Time: 1.5 hours]
2. If there is a problem, throw an error asking for more or new long and short tool measurement data. [Time: 0.5 hours]

FR9: Using forward kinematics, align commands such that for a predicted measured position of the machine for a given command, the actual measurement is close to the predicted value.
[Developers: Kevin Zheng — Time: 5 hours]

1. Parse output from the transformed measurements data structure. [Time:.5 hours]
2. Load the machine forward kinematic from the data structure. [Time:.5 hours]
3. For each measurement in the aligned and transformed measurement, test that a predicted measurement value is reasonably close to a predicted value for that measurement. [Time: 4 hours]

FR10: If an estimated command position and actual command position do not match, discard the measurement. The error should not be greater than 10^{-2}.
[Developers: Kevin Zheng — Time: 5 hours]

1. Discard a command or measurement for all misaligned commands such that the actual and predicted values are close for every measurement in the data structure. [Time: 4 hours]
2. Continue attempting to align predicted values from commands to actual values from the transformed measurements tool. [Time: 1 hours]

FR11: Parse controller profile to the GA Table Selection.
[Developers: Kevin Zheng — Time: 1 hours]

1. Read in the controller profile, based on user input, to a data structure containing arguments to the GA. [Time: 1 hour]

FR12: Supply the aligned commands to the Implicit Loop Method and GA Table Selection MATLAB DLLs to obtain output for generating the tables and reports.
[Developers: Kevin Zheng — Time: 4 hours]

1. Open a handler for dynamically linked libraries and supply aligned data, tool lengths, relevant link lengths, and transformation from measurement frame to the ILM DLL. [Time: 1.5 hours]
2. Open a handler for dynamically linked libraries and supply aligned data, tool lengths, relevant link lengths, and transformation from measurement frame to the GA DLL. [Time: 1.5 hours]
3. Write all input and output to the ILM and GA to a data structure. [Time: 1 hours]

FR13: Build performance report from the output of the Implicit Loop Method DLL containing a goodness of fit value, calculation time, mean and maximum residual error, and plots of the identified table function.
[Developers: Joseph Steurer — Time: 6 hours]

1. Parse output of ILM DLL into a data structure. [Time 1.5 hours]
2. Title and format the report in Microsoft Word format using a C# library. [Time 1 hour]
3. Write the goodness of t value, calculation time, and mean and maximum residual error to a Microsoft Word document using a C# library. [Time 1.5 hours]
4. Plot the identified table functions using a C# library and output to the same document. [Time 2 hours]
5. Save the report to a directory. [Time 1 hour]

FR14: Generate a second performance report from the output of the GA Tables DLL showing the solutions to the genetic algorithm, graph of errors, performance statistics, calculation time, chi squared value, and max/mean residual error.
[Developers: Joseph Steurer — Time: 7.5 hours]

1. Parse output of GA Tables DLL into a data structure. [Time: 1.5 hours]
2. Title and format the report in Microsoft Word format using a C# library. [Time: .5 hours]
3. Write the performance statistics, calculation time, chi squared value and mean and maximum residual error to a Microsoft Word document using a C# library. [Time 1.5 hours]
4. Plot the graph of errors using a C# library and output to the same document. [Time: 2 hours]
5. Output the solutions to the genetic algorithm using a C# library to the same document. [Time: 2 hours]

FR15: Format the compensation tables populated by the GA algorithm based on a user specified controller type.
[Developers: Joseph Steurer — Time: 3.5 hours]

1. Open a file handler to begin parsing the raw tables generated from the GA. [Time .5 hours]
2. Select the controller based on user input. [Time .5 hours]
3. Format the tables according to a predefined controller specification. [Time 2 hours]
4. Save the formatted tables to a new file. [Time .5 hours]

Quality Requirements

QR1: Integrate a manual discussing how the application transforms its input into output.
[Developers: Joseph Steurer — Time: 6 hours]

1. Identify the steps required to transform user input to output. [Time 1 hour]
2. Summarize the steps required to transform user input to output. [Time 2 hours]
3. Add a menu option to the user UI for accessing the manual. [Time .5 hours]
4. Show the steps neatly in a new window such that the user can understand how the application behaves. [Time 2.5 hours]

Platform Requirements

PR1: Windows XP and Windows 7

1. Ensure the software runs on a Windows XP and Windows 7 platform.
2. Note missing dependencies such as libraries or .net framework installations.

PR2: C# / Microsoft Visual Studio

1. Obtain Visual Studio, write code using C# .net.

PR3: Mechanism for parsing output from the MATLAB DLL files.

1. Identify a C# mechanism for parsing DLLs.
2. Ensure that input to a DLL and output from a DLL can be written from and read into the applications memory.

PR4: Library for interaction with compiled MATLAB code.

1. Obtain a library from mathworks to facilitate communication between C# .net and compiled matlab code.
2. Add the library to the project.

PR5: Library to support interaction with Microsoft Office for automatic report generation.

1. Obtain a library for generating Microsoft Word documents using C#.
2. Add the library to the project.

PR6: Library to generate plots based on data obtained from MATLAB DLL.

1. Obtain a library for generating plots using C#.
2. Add the library to the project.

Process Requirements

The development group must provide weekly or bi-weekly updates to clients.

1. Set up meetings with Boeing contacts.
2. Establish agenda for team meetings.
3. Provide feedback, answer questions, explain progress and goals.

Coding will be managed in sprints, where the team meets to code as a group.

1. Team will meet in a classroom to work on project coding.
2. The team will work off of individual laptops.

Develop test cases for each step of the application to verify the correctness of output at the end of each processing step.

1. Consider possible ways to break a requirement.
2. Add test cases for these considerations.
3. Resolve errors and document results of test.

Version control must be employed throughout the development process.

1. Use the Git versioning system.
2. Team members will push and pull from the same repository.
3. Each team member has his own branch.
4. Team Lead will merge team member branches into master.