# Project Plan
# CSV Software

Authors: Heather Sills, Liam Carney
Creation date: 26/3/18
Last Revised: 18/4/18
Version: 1.5

# Table Of Contents

# Introduction

### Purpose of Plan

Gait Research Project Plan will provide a clear definition as well as outline the roles and responsibilities for the delivery of the proposed project. The plan will stipulate the project's goals and objectives accordingly and provide the clear parameters of scope. The plan identifies the potential constraints of the project and where available will outline proposed solutions. The plan has been developed in consultation with all stakeholders and will serve as an agreement between the following parties: The Clients, Project Supervisor, Project Team, and end users associated with and/or affected by the project.

### Project Description

To produce software designed to take raw input of participant weight, speed, and centre of mass distribution and ultimately process the information to produce a structured data set, and respective output for 3-dimensional and 2-dimensional graphing.

### Project Overview

There is currently a solution in place that requires the use of two separate programs. The first program is used to collect the data identifying the position coordinates of each marker, this data is then exported into a second program to apply a regression based formula to construct the centre of gravity position from the imported position coordinates from the markers. The second program exports ASCII files for use by the end user. The Client is seeking a more efficient method of taking the input data to produce an output file.

This project will increase the productivity and workflow of the gait research project and allow more flexibility while conducting research. It will also remove the dependency on Matthew to process all of the files and data for Jelena.

The software produced will be written using the language java, with the CSV / TSV file model in mind and to be compatible with Excel and MATLAB formats. It will be capable of importing raw data processing through the various formats aforementioned above and outputting to a database.

GUI and menu design must be relatively easy to navigate and use with minimal user input from importation to exportation. Options and settings must be present, and available to the user with the ability to see help and about options.

# Background Information/Available Alternatives

## User Stories

### User Story 1: Data format manipulation

Priority: High

Accept a file in .csv which contains multiple runs. For an example of such a file see the attached data_in_example.csv. Group the runs of the same people together. Have the option of relabelling the names. Note that the speed isn't included in the file, but a sequence number on each file name. The user should be able to enter a base speed, and this will be the first speed, and subsequent sequence numbers will correspond to subsequent speeds. Note that the frame number is given, but in order to get the time, we need the time step size, this should be entered by the user during the import process.

For a particular run some data could be missing. This will manifest itself with missing x, y, and z values. This could happen because the sensors are occluded from the camera momentarily. If this only happens for a few frames at the start and/or end, then we simply drop these frames. However, if this happens somewhere in the middle of the run then we drop the whole run, and alert the user as to the missing data.

With this data the software should now write to a new, or append to an existing, csv "database". For an example of such a file see the attached data_out_example.csv. Note that the columns correspond to:

label, speed, time, x, y, z

If appending to an existing database, check for conflicting labels. If conflicts exist, ask the user whether they want to include the new data and overwrite the old data. If they choose to proceed then all previous entries corresponding to the conflicting label should be removed from the csv "database".

### User Story 2: Data summary

Priority: High

There are a variety of summary statistics we wish to calculate for a single run, e.g. the "spring" constant. We want to be able to select data from the csv "database", see User Story 1, and get an output of these summary statistics. Another feature is that we want to select a particular label and create a table of their summary statistics at the different speeds they ran at. These tables should be in a form where we easily can copy and paste one or another row/column. Moreover, we want to create tables of these summary statistics for different labels but a single speed. For an initial list of summary statistics see the below MATLAB script, the calculation of the desired summary statistics are shown in red. We could add more summary statistics in the future, so your codebase has to be flexible in this regard.

### User Story 3: Centre of mass regression

Priority: Very low

Write some software that does the centre of mass regression from the position of the recorded sensors, see papers Kath sent.

### Solution Alternatives

#### Alternative 1

Using java we create a cross platform system to be used in conjunction with Kath's software and create the various outputs for Lena's graphs and MATLAB. Issues pertaining to this solution include, limited to no access to the original software input and output, limited programing functionality with the SDK and no guarantee that the solution is technically feasible.

#### Alternative 2

Using excel we write VBA extensions to be installed with excel that automatically import the raw form of data, convert it to tsv, implement graphing solutions inside of excel, and output from tsv to MATLAB code. Extensions would have a higher chance of deprecation since excel versions especially vba extensions tend not to be backwards compatible with previous versions because of large internal changes to the api.

# Objectives

## Project Objectives

- To create a more efficient solution to the current solution
- To produce real time output for the end user
- Ensure output is mathematically correct throughout all stages of development
- Ensure codebase is modular for later improvement and functionality
- Ensure code is well documented
- Ensure class structure is well defined

Ensure cross-platform support

# Scope

**Scope Definition**

Upgrade the current software solution to the gait research project. Develop a software product that can accept a file in .csv containing multiple data sets. The software will have the capability of grouping the data together in accordance with set requirements as outlined in the *User Stories*. The software will provide a GUI for simple data entry for the speed variable. Following data input the software will write new, or append to a csv database the data: label, speed, time, x, y, z. Some capability for checking data integrity will be built into the system based on checking for conflicting labels. A user prompt will provide acceptance of the changes.

An output of modifiable summary statistics will be produced by the software, in accordance with the *User Stories.*

**Items Beyond Scope**

- Any development or documentation produced past the 3 April 2018
- The construction of an independent database and archiving system
- Any Matlab development besides the integration of the script given by Matthew Cooper
- Integration of the software directly from the raw ascii inputs give by Kath, or the calculation of the adjusted gait regression.
- The final image storage of any graphs produced through the software.

**Project Budget**

| Item Name | Item Cost | Total Needed | Cost Total |
|---|---|---|---|
| Developer Hours | $0 /hr | 400 | (Appx. 400), $0 |
| Software Licensing | $0 | 1 | $0 |
| Physical Hardware | $0 | 1 | $0 |
| Asset Licensing | $0 | 15 | $0 |
| | | **Sub Total** | $0 |

**Initial Project Risk Assessment**

| Risk | Risk Level<br><br>L/M/H | Likelihood of Event | Mitigation Strategy |
|---|---|---|---|
| **Project Size** | | | |
| Person Hours | **M:** Over 400 | **Certainty** | Extensive project management approach, Communications plan |
| Project Schedule | **M:** Over 3 months | **Certainty** | Planned Project timeline with frequent meetings, daily discussions and progress reports |
| Team Size at Peak | **H:** Two members | **Certainty** | Planned Project timeline with frequent meetings, daily discussions and progress reports |
| Number of Interfaces to Existing programs | **H:** Two | **Certainty** | Structure a plan for the integration, early testing will help better grasp the time required |
| **Project Definition** | | | |
| Project Scope Creep | **L:** Scope defined and set | **Unlikely** | Scope defined in project plan, set by Project Team, (Project Manager) to prevent scope creep |
| Project Deliverables unclear | **L:** Well defined | **Unlikely** | Included in project plan, Deliverables well defined |

| | | | |
|---|---|---|---|
| Cost Estimates Unrealistic | **L**: Cost well within budget, unlike to ever exceed | **Unlikely** | Included in project plan, no risk of wage runoffs, no need for additional software or hardware so very little chance budget or cost will ever grow. |
| Timeline Estimates Unrealistic | **M:** Timeline assumes completion and testing within time frame, although time could exceed | **Somewhat likely** | Timeline reviewed by project manager, timeline and scope creep to be reviewed weekly to prevent overtime |
| Number of Team Members Unknowledgeable of Business | **L**: Team well versed in client requirements and user stories | **Unlikely** | Information gaps to be filled when needed or when they arise by project manager. |
| **Project Leadership** | | | |
| Absence of Commitment Level/Attitude of Management | **L:** Understands value & supports project | **Unlikely** | Frequently seek feedback to ensure continued support |
| Absence of Commitment Level/Attitude of Users | **L:** Understands value & supports project | **Unlikely** | Frequently seek feedback to ensure continued support |
| Absence of Commitment Level/Attitude of Software engineers | **L:** Most understand value & support project | **Unlikely** | Frequently seek feedback to ensure continued support |
| **Project Staffing** | | | |

| | | | |
|---|---|---|---|
| Project Team Availability | **M:** Time frame and work schedules make weekly catch ups somewhat difficult | **Somewhat likely** | Continuous communication through other avenues will prevent team members from falling behind and video recordings help messages and workflow to be passed over time periods |
| Physical Location of Team prevents effective management | **M:** Team is separated over distance | **Likely** | Use of Emails, Skype and shared repository. Following Communications Plan |
| Weak User Participation on Project Team | **L:** Users are part-time team members | **Unlikely** | User Group Participants coordinated by full time employee |
| **Project Management** | | | |
| Procurement<br><br>Methodology Used foreign to team | **L:** Procurement Methodology familiar to team | **Unlikely** | N/A |
| Change Management Procedures undefined | **L:** Well-defined | **Unlikely** | N/A |
| Quality Management Procedures unclear | **L:** Well-defined and accepted | **Unlikely** | N/A |

**Milestones**

**Phase I: Interview Clients and record user stories**

Interview the clients and list down what their stories are and any relevant information that will be included within scope. Throughout this process, when necessary prioritisation of key elements of the project will occur and elements of low priority will be listed as secondary objectives or removed from scope.

**Phase II: Present Project Plan**

Project plan will be present to the clients, and pending amendment will proceed accordingly to schedule if decided so. Amendments can occur at any stage throughout the project lifecycle, if all parties are in agreement within doing so.

**Phase III: Initial Programing and testing**

Programing will start and initial testing and continuous client interaction for feedback. This phase's primary goal is to get preliminary functionality underway and present it to the clients. From there extended functionality and Graphical user interfaces will be implemented and tested with the clients.

**Phase IV: Software Testing**

Testing of the software with moderate functionality with the clients, making sure that the primary goals of the software are near or completed, with secondary goals to be reviewed and pursued if time and budget permits.

**Phase V: Review Software**

Review of the software, refactoring of classes and final restructuring of the project and design. Review of the graphical user interface and the user input and output processes.

**Phase VI: Conduct Training**

In this phase clients will be trained in the semi-final build of the software. They will be walked through general use of the software and talked through its extended features. By this stage all key features and goals of the software will be complete and implemented into the software. Help documents will be included in this final build as well as gui-based information and manuals.

**Phase VII: Present final software build**

Final build of the software is to be presented to the clients with source code, code docs and repository history. All manuals, tutorial videos and information to be compiled into a "wiki" for future references.

# Assumptions

### Project Assumptions

The following assumptions were made in preparing the Project Plan:

- The software will not require any form of network as the database will be local and stored on a user by user basis.
- Since many of the users that will be using this software will be cross platform, a cross platform language will be needed.
- The users for this software will be mostly academics but will also require basic user prevention and warning dialogs.
- Database does not need to be in a structured query language or document driven structure.
- Software will need to be MIT licensed to allow further development at a later date.
- Software will be running of the latest version of the software language chosen.
- Export will be to microsoft office excel 365 version and to the latest matlab version.

# Constraints

### Project Constraints

The following represent known project constraints:

- Project time is limited to only 87 days
- Limited amount of programmers to compete code
- Limited amount of perspectives
- Physical location and distance from all parties involved requires remote meetings and workspace
- Limited and non-shared development time means that co-programming and debugging time will be longer.

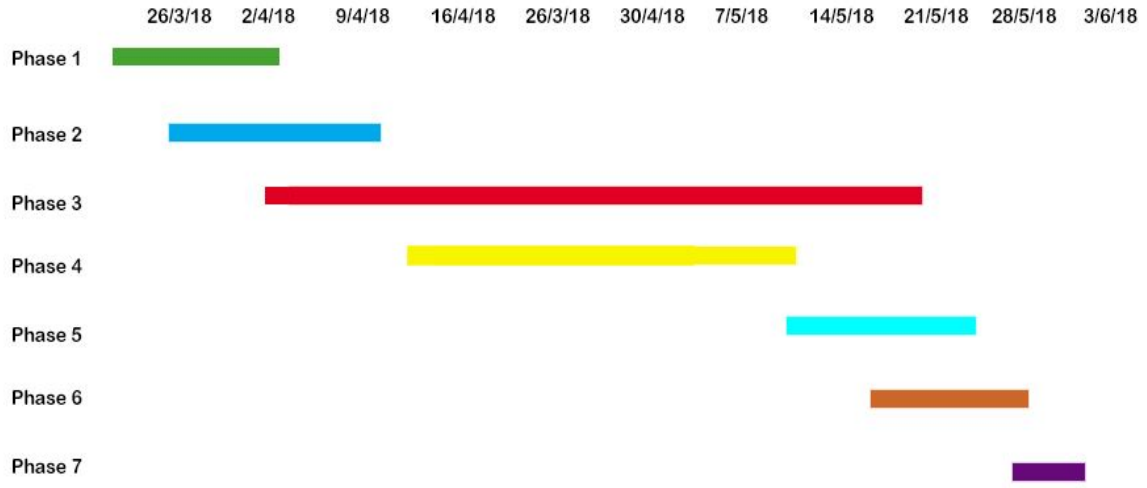
# Feasibility Assessment

The proposed software meets the Client requirements and can be produced within the proposed timeframe provided risks are appropriately mitigated in accordance with this plan.The project will be completed by June 2018. The project will not exceed the budget and the project code will be available and licenced for future development and revision.

# Project Management Approach

## Project Timeline

| | | |
|---|---|---|
| Phase I | Completed by | 2 April 2018 |
| Phase II | Completed by | 9 April 2018 |
| Phase III | Completed by | 21 May 2018 |
| Phase IV | Completed by | 14 May 2018 |
| Phase V | Completed by | 28 May 2018 |
| Phase VI | Completed by | 3 June 208 |




## Project Roles and Responsibilities

| | | |
|---|---|---|
| Team Member | Liam Carney | - Programing, Documentation, Meetings |
| Team Member | Heather Sills | - Programing, Documentation, Meetings |
| Client | Matthew Cooper | |
| Client | Jelena smaltz | |
| Other Stakeholders | Kath Shorter | |

## Communications Plan

Weekly skype meetings will be facilitated to enable discussions about the project with the clients and to discuss and potential changes or additions to the software within scoping boundaries. These meetings will be documented throughout the process.

# ATTACHMENTS

## M_lab_script

```
clear all;

clf;

% get the data
[t, x, y, z] = getcom('lena', '9 km/hr');

% smooth data
x = fft_filter(x, 0.03);
y = fft_filter(y, 0.3);
z = fft_filter(z, 0.03);
zA = fft_filter(z, 0.3);

% calculate derivative
dt = t(2) - t(1);
%zt = diff(z)/dt;
zt=(z(3:end)-z(1:end-2))./(2*dt); %associate with t(2) to t(end-1)
%ztt = diff(zt)/dt;
ztt=(z(3:end)-2*z(2:end-1)+z(1:end-2))./(dt^2); %t(2) to t(end-1)
% plot data
plot3(z(2:(end - 1)), zt, ztt, 'b');
xlabel('z');
ylabel('z_t');
zlabel('z_{tt}');

% compute plane of best fit
% X = z(1:(end - 2));
% Y = zt(1:(end-1));
% Z = ztt;
X = z(2:(end - 1));
Y = zt;
Z = ztt;
%XA = zA(1:(end - 2));

% construct least squares error function, and minimize
[n, c] = planefit(X, Y, Z);
c=c/n(3)
n=n/n(3)

z0 = (min(X) + max(X))/2
stiffness = n(1)
viscosity = n(2)
%quality = sqrt(stiffness)/viscosity
%forceOverMass = c - stiffness*z0

%Determine points where zt is 'close' to zero
zt0i=find(Y>-1e-2 & Y<1e-2 & Z>0);
xconditions=X(zt0i);
av_xc=mean(xconditions);
zt0i2=find(Y>-1e-2 & Y<1e-2 & Z<0);
xconditions2=X(zt0i2);
av_xc2=mean(xconditions2);

A=0.5*abs(av_xc-av_xc2) % average amplitude
dz=z(2:end-1)-z(1:end-2);
E=sum(dz.*ztt)  %energy lost in 10 secs
N=5*sqrt(stiffness)/3.14; %number of cycles in 10 secs
dE=3.14*E/(5*sqrt(stiffness)) %energy lost in one cycle
%velocitx=sqrt(-dE/5*3600/1000)
Quality=-5*stiffness*sqrt(stiffness)*A^2/(E)
Frequency=sqrt(stiffness)/(2*pi); %stride frequency
```