

Step File Converter Design Document

Alan Halpin

Started 30/11/2020



Supervisor: Joseph Kehoe
Student Number: C00229361

Table of Contents

Abstract	2
Introduction	2
System Architecture Stack Diagram	3
Class Diagram	4
Domain Model	5
Sequence Diagram	5
STEP Data Structure	7
Use Cases	8
Pseudo-Code Algorithms	10
Extract Features - Searching Algorithm	10
Generate Permutation	11
Project Plan	12
Plagiarism Declaration	13
Declaration	13

Abstract

The purpose of this document is to provide details of the system design of the Step File Converter. The system architecture and functional design will be outlined and each function described in detail using pseudocode where necessary. This tool is a proof of concept.

Introduction

The STEP File Converter is a tool for CNC engineers, designed to automate a lengthy and costly process. Nothing like this application currently exists. The value is derived from the process being automated. This process involves performing vibration analysis on different permutations of an object until a satisfactory sequence of faces is found. Vibration analysis is performed to ensure the piece worked on remains stable throughout the cutting process. The proposed tool will extract geometrical data from a STEP file and catalogue the different faces of the object described. High level features will be identified from these faces. One by one these features will be subtracted from a solid block shape. Vibration analysis is performed between each step, if the piece does not pass the requirements then a different face will be used. This will eventually create a list of faces to cut in order. Ex. A,B,F,G,C might be an order in which to carve first. Although vibration analysis already exists within various CAD applications, the process described only exists in a manual implementation. This tool will use the Solid Works API to outsource the calculations and identify the optimal cutting order within the application. An engineer will pass a STEP file to the program and the program will start working on the results. Depending on the parts complexity this process may take some time. Typically this process is done manually and can take up to three weeks to complete, this tool hopes to automate that process and reduce that time down to one day.

System Architecture Stack Diagram

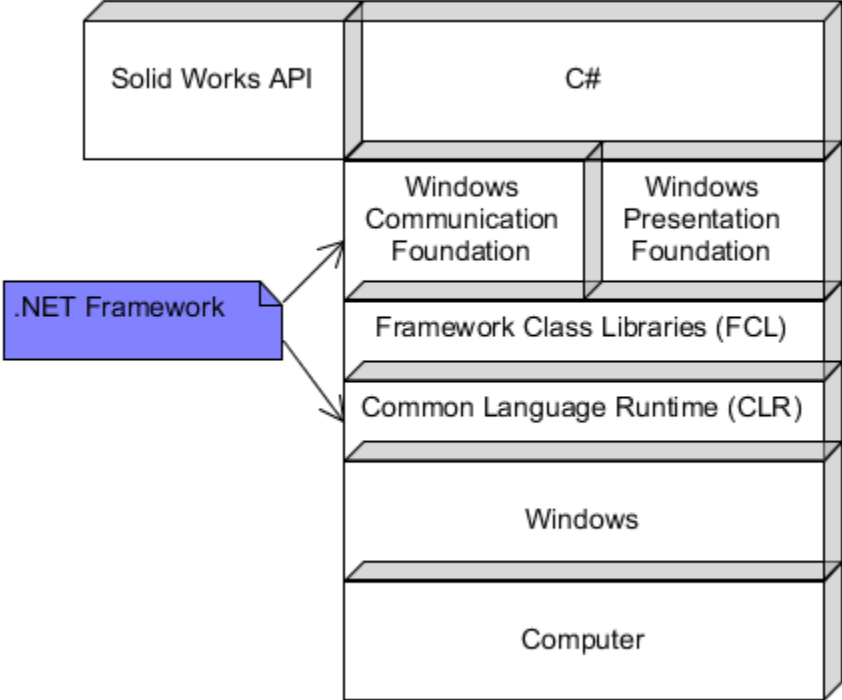


Figure 1: System Architecture Stack

Class Diagram

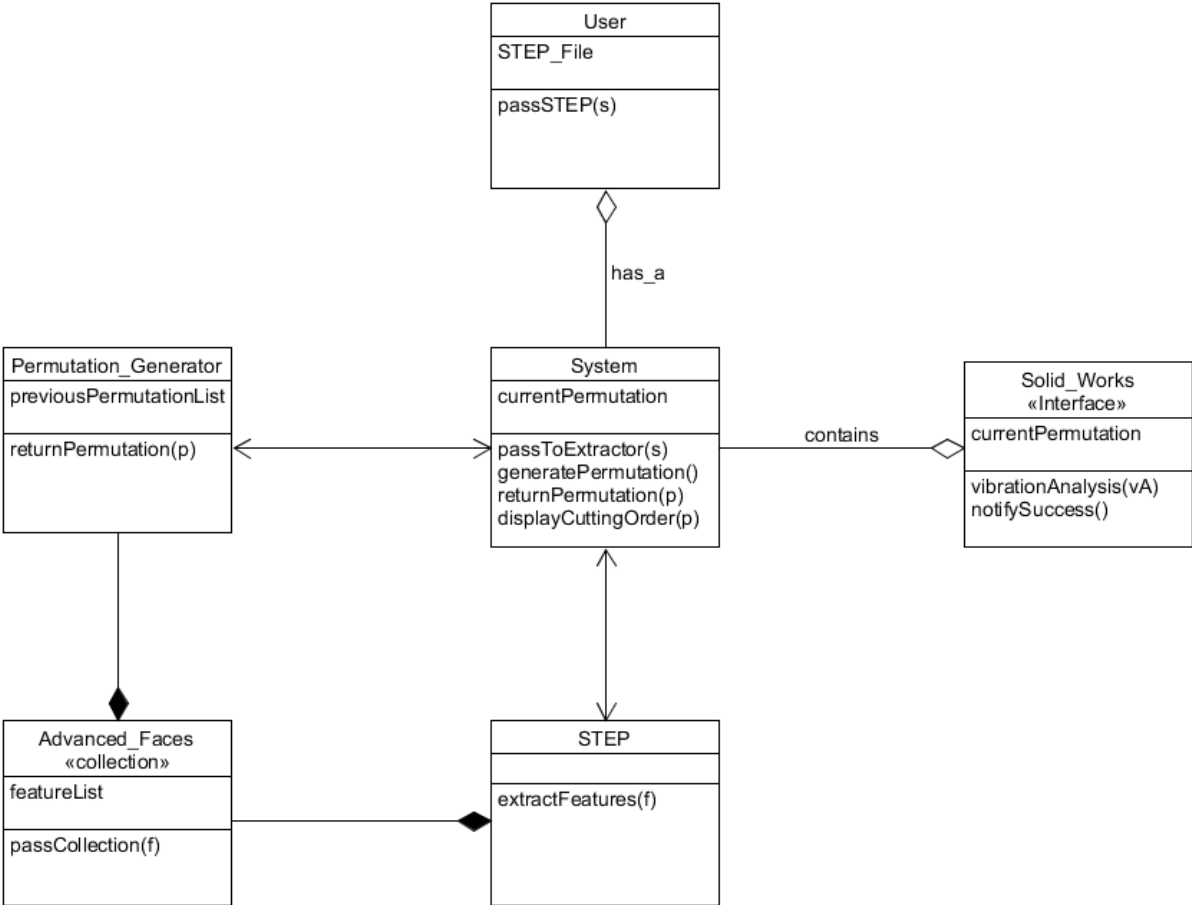


Figure 2: Class Diagram

Domain Model

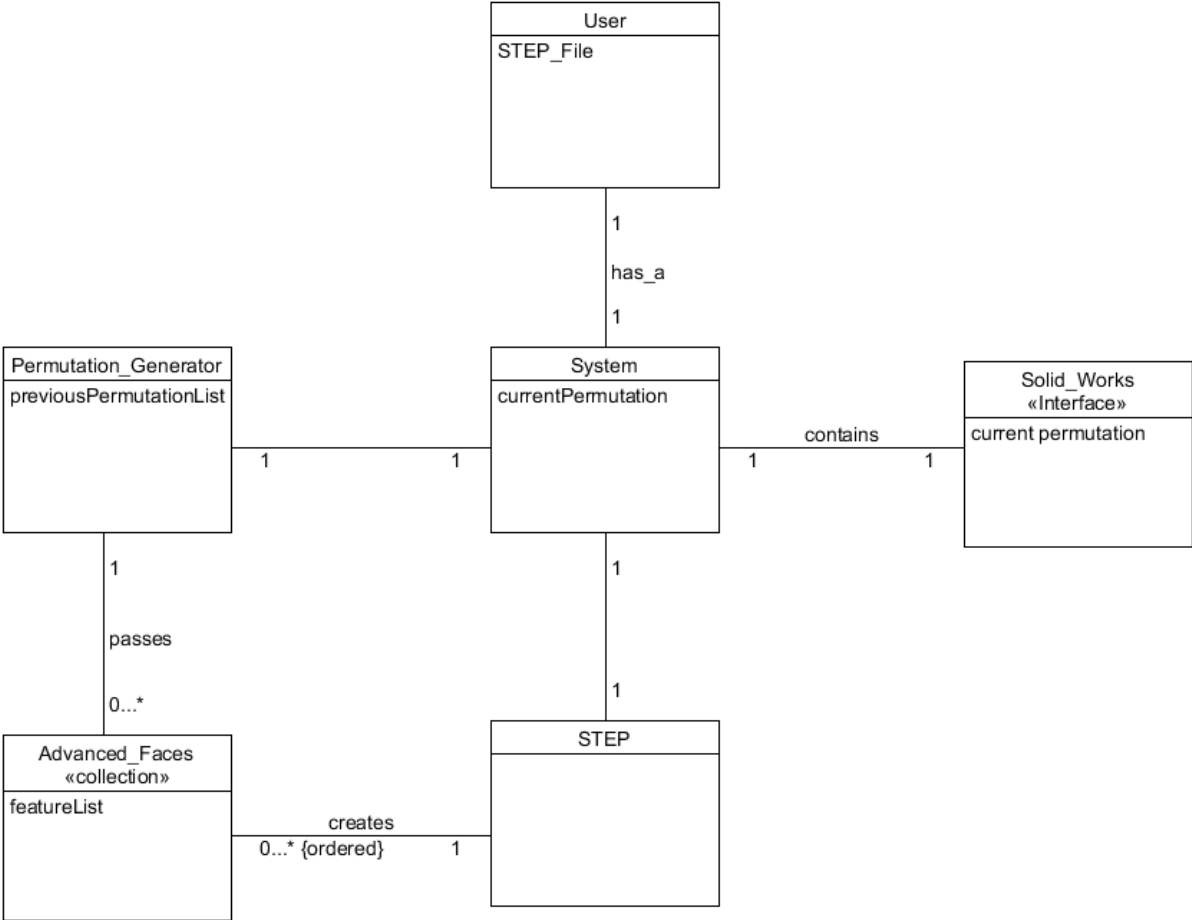


Figure 3: Domain Model

Sequence Diagram

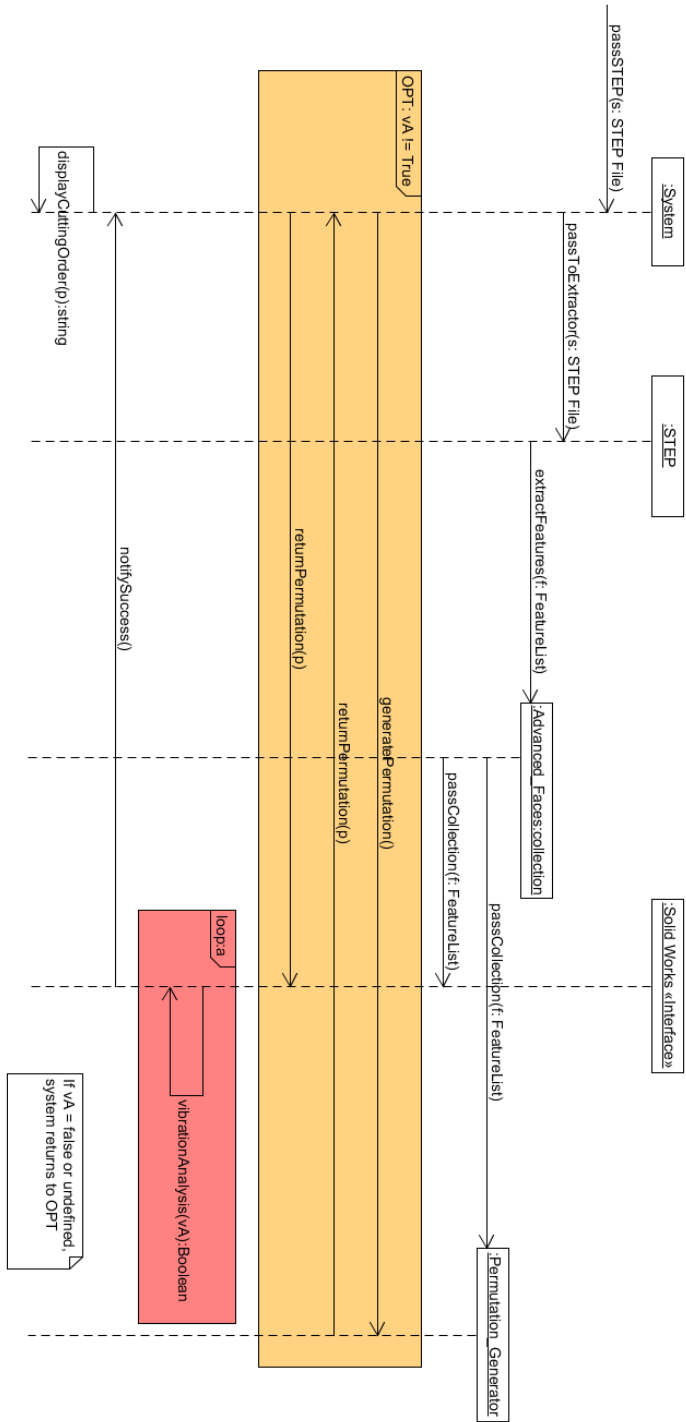


Figure 4: System Sequence Diagram

STEP Data Structure

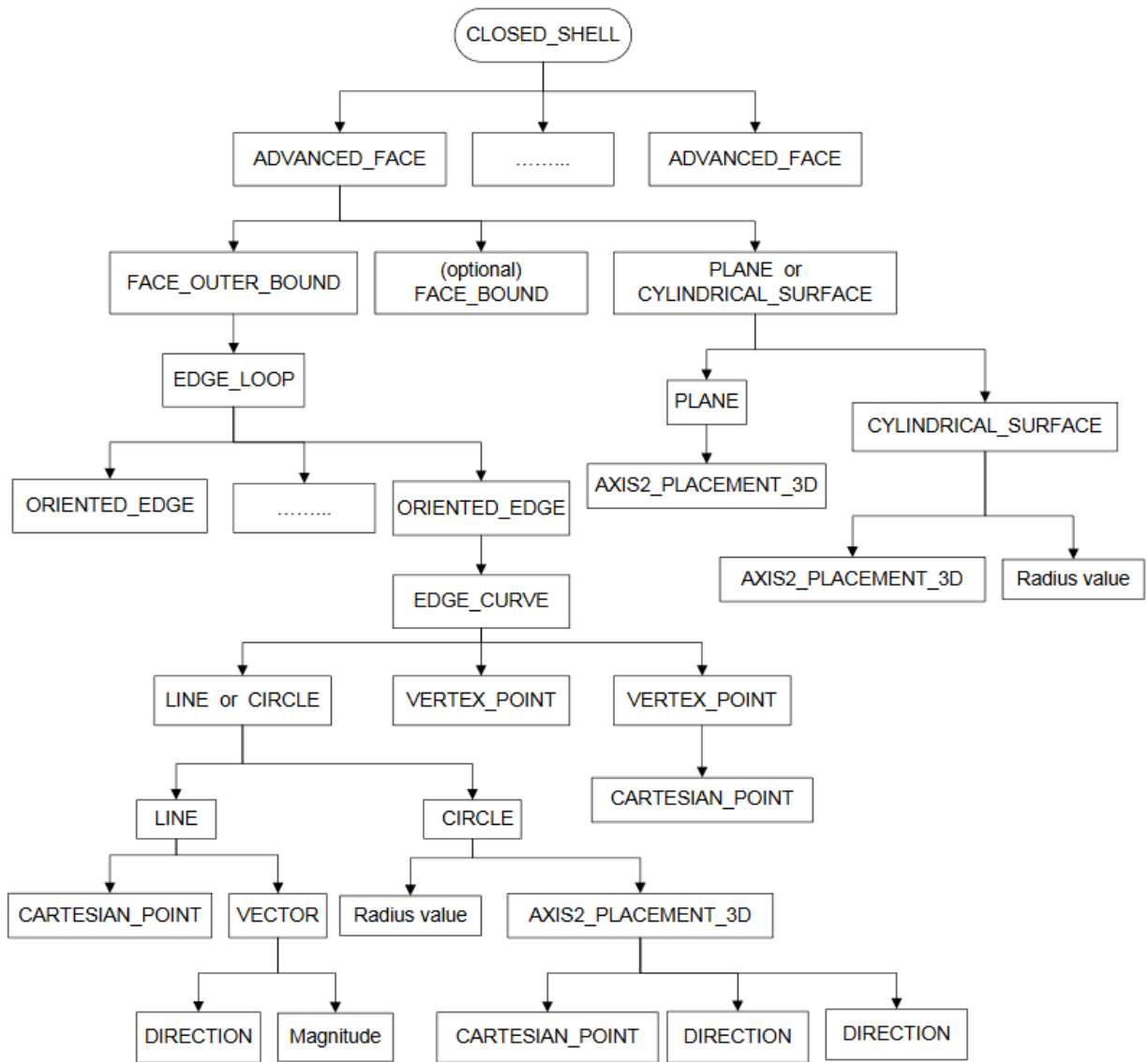


Figure 5: STEP AP203 Data Structure

[Source](#)

```

#18 = ADVANCED_FACE ( 'NONE', ( #108 ), #181, .F. );
  #108 = FACE_OUTER_BOUND ( 'NONE', #99, .T. );
    #99 = EDGE_LOOP ( 'NONE', ( #137, #58, #160, #69 ) );
      #181 = PLANE ( 'NONE', #140 );
        #140 = AXIS2_PLACEMENT_3D ( 'NONE', #222, #207, #205 );
  
```

ETC...

A tree data structure will be required to store this information at run time.

Use Cases

Name:	Extract Features
Actors:	Engineer (User)
Description	Use case for extracting geometrical features from STEP file.
Preconditions	A valid STEP file has been uploaded.
Main success scenario:	<ol style="list-style-type: none"> 1. The user tells the system to begin extracting features. 2. The STEP file is read into memory and catalogued. 3. A recursive algorithm is used to find advanced faces and their sub-properties. 4. Advanced face is found. 5. Sub-property(ies) is found and stored locally. 6. All properties have been located, and the system is informed of success.
Post Conditions	The features have successfully been stored.
Alternative	<ol style="list-style-type: none"> 4a. Advanced face is not found. <ol style="list-style-type: none"> 1. System is informed of failure. 2. Application is restarted. 5a. No sub properties found. <ol style="list-style-type: none"> 1. System searches for the next advanced face.

Name:	Generate Permutation
Actors:	NULL
Description	Use case for generating permutations of potential cutting order. The lexicographic order algorithm is used for permutations.
Preconditions	Features have been stored locally
Main success scenario:	1.
Post Conditions	Permutation is produced.

Name:	Vibration Analysis
Actors:	Solid Works API
Description	This use case involves exporting faces one by one to the solid works API in order to perform vibration analysis. The system will pass a feature and wait for a result.
Preconditions	A permutation has been generated.
Main success scenario:	<ol style="list-style-type: none"> 1. Feature is passed to Solid Works. 2. Feature is subtracted from a solid block piece. 3. Vibration analysis is performed using Solid Works. 4. Results are sent back to the system. 5. Results are received. 6. Results pass, next feature is passed. 7. All features have been passed and results have all passed. 8. A cutting order has been solved.
Post Conditions	The user is displayed the cutting order.
Alternative	<ol style="list-style-type: none"> 1a. No features available <ol style="list-style-type: none"> 1. System is informed of failure. 2. Application is restarted. 6a. Results do not pass requirements <ol style="list-style-type: none"> 1. New permutation is requested from permutation class. 2. New permutation is received. 3. Next feature is passed.

Pseudo-Code Algorithms

Extract Features - Searching Algorithm

```
2  var foundLines = new List<string>();
3  var nextLines = new List<string>();
4  bool numberFound = false;
5  bool lastNumberFound = false;
6  string number = "";
7
8  foreach (string line in dataObj.faces) // cycles through advanced face locations
9  {
10     nextLines.Add(line);
11     while (lastNumberFound != true) // runs until last number is found
12     {
13         foreach (string nextLine in nextLines) // cycles through next lines
14         {
15             char[] characters = nextLine.ToCharArray(); // Convert current line to char array
16             foreach (char ch in characters) // cycles through char array
17             {
18                 if (numberFound == true)
19                 {
20                     if (Char.IsDigit(ch))
21                     {
22                         number += ch;
23                     }
24                     else
25                     {
26                         // save number
27                         foundLines.Add(number);
28                         numberFound = false;
29                         number = "";
30                     }
31                 }
32                 else if (ch == '#')
33                 {
34                     numberFound = true;
35                 }
36             } // ch for ends
37             if(foundLines.Count == 0)
38             {
39                 lastNumberFound = true;
40             }
41         } // next line ends
42         nextLines.Clear();
43         nextLines.AddRange(foundLines); // Add the lines found to next line list
44         foundLines.Clear();
45     } // while ends
46     // Add steps found to Features List here
47     lastNumberFound = false;
48 }
```

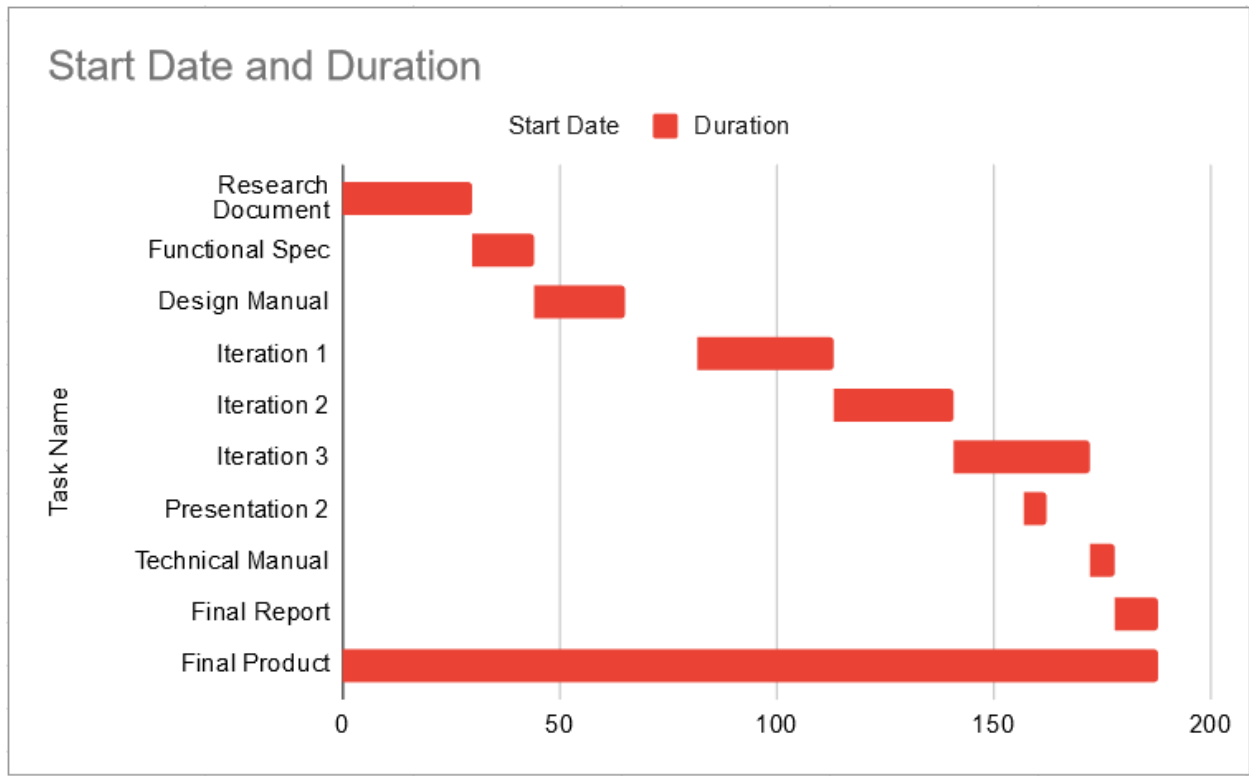
Generate Permutation

(Heap's Algorithm)

```
void permutator(int[]a , int size, int n)
{
    if(size==1)
    {
        printArr(a,n);
        return;
    }
    for(int i=0; i<size; i++)
    {
        permutator(a, size-1, n);

        if(size%2==1)
        {
            swap(a[0], a[size-1]);
        }
        else
        {
            swap(a[i], a[size-1]);
        }
    }
}
```

Project Plan



Start Date: 14/10/2020

End Date: 19/04/2021

Project Duration: 188 Days

Plagiarism Declaration

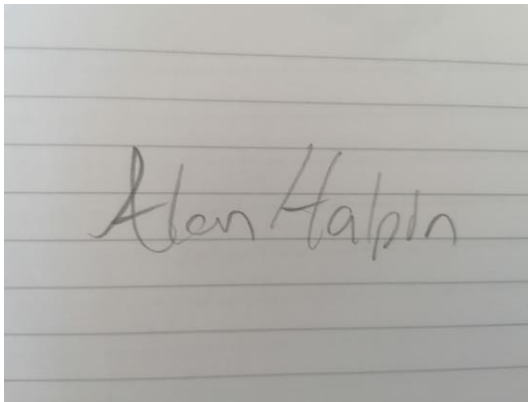
Declaration

- I declare that all material in this submission e.g.thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name: Alan Halpin

Student Number: C00229361

Signature:

A photograph of a handwritten signature 'Alan Halpin' on a piece of lined paper. The signature is written in dark ink and is centered on the page.

Date: 30/04/2021