Dylan Barnes, Ryan Kilbride, Cameron Kerbaugh

CSci 463, Spring 2016, Dr. Hassan Reza, Ph.D.

DC# Converter Software Management Plan

DC# Converter Software Management Plan Table of Contents

[Revision History 2](#_Toc446438699)

[1 Introduction 3](#_Toc446438700)

[2 Methodology 3](#_Toc446438701)

[3 Work Planning 3](#_Toc446438702)

[4 Estimation 3](#_Toc446438703)

[5 Code Diagrams 4](#_Toc446438704)

[5.1 UML Class Diagram 4](#_Toc446438705)

[5.2 UML Use Case Diagram 5](#_Toc446438706)

[6 Possible Problems 6](#_Toc446438707)

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VERSION | TEAM MEMBER | DESCRIPTION | | DATE |
| 0.1.0a | Dylan Barnes | Initial Document Layout and Introduction | 01/29/2016 | |
|  |  | Add Sub-Clause 5.1 *UML Class Diagram*  Add Sub-Clause 5.2 *UML Use Case Diagram*  Add UML Class Diagram 5-1 to *Code Diagrams*  Add UML Use Case Diagram 5-2 to *Code Diagrams* |  | |
| 0.2.0a | Ryan Kilbride | Add FP Cost Estimate 4-1 to Main-Clause 4 | 02/12/2016 | |
| 0.3.0a | Dylan Barnes | Add Page Numbers  Add Table of Contents  Correct ambiguous grammar | 02/18/2016 | |
| 0.4.0a | Ryan Kilbride | Add Figure Caption to Figure 5-1  Add Figure Caption to Figure 5-2  Updated Main-Clause 6  Updated Ambiguous Grammar Throughout Document | 03/07/2016 | |
| 0.5.0a | Dylan Barnes | Add Main-Clause *Revision History* | 03/19/2016 | |
| 0.6.0a | Cameron Kerbaugh | Corrected Grammar and Punctuation | 03/22/2016 | |
|  |  |  |  | |

# Introduction

DC# Converter is a portable, multi-function conversion tool. It is part of a larger open source software suite, known as Data Control Sharp (DC#). Main features include: simple file conversion using a graphical user interface (GUI), C# classes, or command line arguments; a list view display of the data contained within converted files; and various options to modify converted data. These features are further explained in the DC# Converter Requirements Analysis and Specification document.

This document will explain the planning and development methodologies we intend to use throughout the development of DC# Converter. In addition to this, a rough estimate will be calculated to determine how much work will be required, as well as how that workload will be distributed between the group members. We will be using Visual Studio software for all of our software planning and development.

# Methodology

Contrary to the original project requirements, we will be using an Agile methodology throughout the development of our software. Therefore, instead of completed documentation for every project deliverable, we will be submitting a prototype along with partial documentation. Final documentation will be presented at the end of the semester along with our release candidate software suite.

# Work Planning

In order to implement the Agile methodology, we are using Visual Studio Team Services. This allows us to easily link our change sets, models, and work tasks together to form a more unified approach to software development.

Instead of listing the individual tasks in an offline document, we have decided to take a more modern approach by using a web-based iteration tracker. This Microsoft application allows us to create stories with various tasks. Below you will find all relevant links required to review our work planning and progress. Due to the nature of the Agile methodology, sprints are only planned out a few weeks in advance to allow for flexibility.

|  |  |
| --- | --- |
| Location | Link |
| Task Board | <https://barndyla.visualstudio.com/DefaultCollection/SE%20Project/_backlogs/taskboard?_a=requirements> |
| All Tasks Query | <https://barndyla.visualstudio.com/DefaultCollection/SE%20Project/_workitems?path=My%20Queries%2FAll%20Tasks&_a=query> |

Table 3-1: Links for work planning and project tracking

***Note:*** *If you are unable to access the link, you may need to reach out to* [*Dylan.a.barnes@und.edu*](mailto:Dylan.a.barnes@und.edu) *to request access.*

# Estimation

For the DC# Converter we have elected to use function points (FPs) as a method of cost estimation. This estimate is based on an adjusted FP weighting. This weighting was supplied by our CSCI 463 lecture notes. There are three categories that subdivide the complexity of the components. The project will be broken down into five sections, as shown below in Table 4-1. Each row shows the number of components and their complexity multiplier, as well as a brief description of those components:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software Complexity: | Simple | Average | Complex | Total |
| Inputs | **2** x 3  (JSON, CSV file parsing) | **1** x 4  (XML file parsing) | **0** x 6 | 10 |
| Outputs | **2** x 4  (JSON, CSV out files) | **1** x 5  (XML out file) | **0** x 7 | 13 |
| Internal Files | **0** x 7 | **0** x 10 | **0** x 15 | 0 |
| Queries | **1** x 3  (Interface displays file in list view) | **3** x 5  (Internal validation of correct conversion for each file type) | **0** x 7 | 18 |
| Interfaces | **2** x 5  (graphical user interface and C# library) | **1** x 7  (command line usage) | **0** x 10 | 17 |
| **Combined** |  |  |  | **58** |

Table 4-1: Function Point cost estimation. Combined total in the lower right indicates computed function point total for this project.

From the table above we obtain an adjusted function point number of 58. Assuming a productivity factor of 6.5 Function Points per Person-Month (FP/PM), project effort can be calculated:

At three months per team member, we are expected to finish within the semester.

***Note:*** *Due to our agile methodology and the nature of cost/effort estimation, requirements are subject to change.*

# Code Diagrams

## UML Class Diagram

Figure 5-1 contains the current UML Class Diagram of DC# Converter. Due to the use of an Agile methodology, this diagram will continue to evolve as future versions are released.

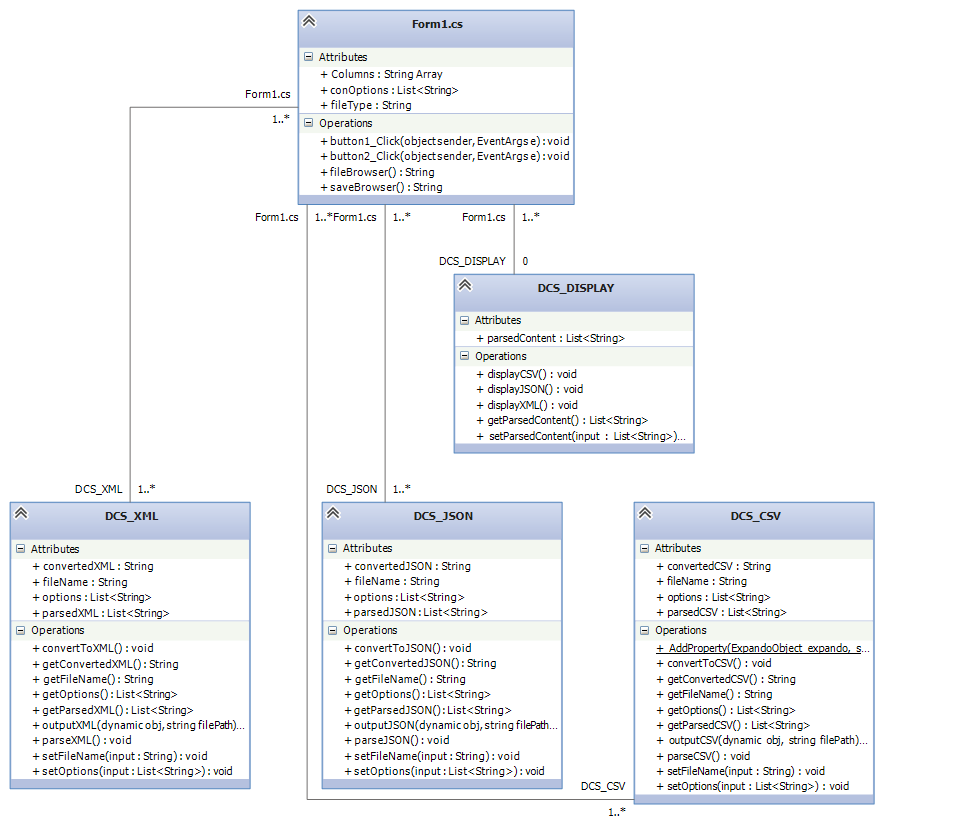
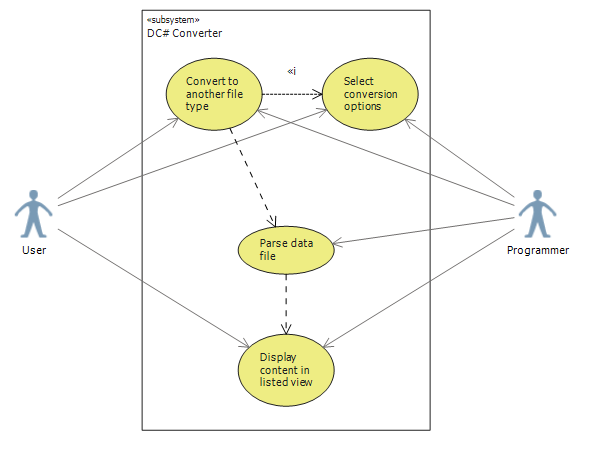


Figure 5-1: UML Class Diagram for DC# Converter

## UML Use Case Diagram

Figure 5-2 contains the current UML Use Case Diagram of DC# Converter. Due to the use of an Agile methodology, new diagrams may be added as features become available.

Figure 5-2: UML Use-Case Diagram for the GUI and C# library. Through the GUI, a user has the ability to convert between files, specify options and settings, and display file contents. The programmer or other external systems, through use of our API, will have the same capabilities, through the use of our functions and code.

# Possible Problems

The following issues may arise:

1. Lack of support/commitment from a team member
2. Underestimating the complexity of a task and/or the time required to complete it
3. Application-breaking bugs that halt progress of the application

The following solutions will help mitigate risk:

1. Team members have agreed to pick up the slack of others if required to ensure the project has a timely finish
2. Tasks may be broken up into an individual story and re-assigned to the following sprint if they are overly ambitious
3. With Visual Studio, we are able to roll back change sets that may have induced a bug