CMPT 473 ASSIGNMENT 1

Edwin Gao <yga22> • Geoff Groos <ggroos>

Table of Contents

Specification of the Program Under Test

The program under test, Xml to Csv Conversion Tool has a fairly simple input/output requirement, owing to its narrowly focused nature. Its input and output are closely analyzed below.

Input

This program takes a standard, validated XML file as its sole input.
Supporting documentation is as follows.

XML

A valid XML file is required for input into this program. The XML standard is defined by the W3C Consortium.
Any XML file(s) found to be in violation of this standard is rejected by the program under test during the file input stage.

Output

This program outputs file(s) in the CSV-format only.
Supporting documentation is as follows.

CSV

CSV lacks a formal standard; however the output of this program is RFC 4180 compliant; that is, the output format is compatible with the *de facto* standard implementation of CSV.
This output file is created at the end of execution.

Category-Partition Method

In this section the chosen component will be analyzed for its input domain and constraints.

Component Specification

The following is the specification of the function under test.

Function	ConvertTables
Syntax	private static void ConvertTables(<parameter1>, <parameter2>)
Details	The ConvertTables function encapsulates all of the tasks involved in the conversion a XML file to a CSV file. Upon being called with a path to the input file (specified with parameter1) the ConvertTables function will attempt to open, validate and parse the XML file at the path specified. If the path is invalid for any reason, program execution is halted. If there have been no errors, the XML file is processed and the internal data structure is written to a CSV file located at the path specified in parameter2. Indirectly, this function relies on multiple other parameters in order to correctly function. A valid XML file is one such example of an indirect parameter.

Category Partition

The function ‘OpenXmlFile’ has the following characteristics.

Parameters
File Name File exists File does not exist Not given

Environmental Variables	
Execution Window (WPF) Command Line XML Content List formatted Nested documents Escape-coded elements Non-printable characters Empty Corrupt	XML Validity Valid Invalid Tables None One More than one Duplicate Columns None Exists

Constraints

Parameters

File Name

File exists	-
File does not exist	[Error]
Not given	[Error]

Environment

Execution

Window (WPF)	-
Command Line	-

XML Content

List formatted	[if NonEmpty]
Nested documents	[if NonEmpty]
Escape-coded elements	[if NonEmpty]
Non-printable characters	[error]
Empty	[error]
Corrupt	[error]

XML Validity

Valid	[if NonEmpty]
Invalid	[error]

Tables

None	[if Empty]
One	[if NonEmpty]
More than one	[if NonEmpty] [property Match]

Duplicate Columns

None	[if NonEmpty]
Exists	[if NonEmpty] [property Match]

Test Report

Our group has developed a test suite using the Visual C# unit test framework, based on the input domains identified during the Category-Partitioning process. These unit tests were carefully designed to cover all input domains.

Please note that the original developer has included in the source code a set of unit tests; we have chosen to leverage some of the tests to provide additional coverage for our chosen program.

The results are as follows:

Test Category	Test Pass Rate
File Name	
File exists	
File does not exist	
Not given	
Execution	
Window (WPF)	
Command Line	
XML Content	
List formatted	100%
Nested documents	
Escape-coded elements	
Non-printable characters	
Empty	
Corrupt	100%
XML Validity	
Valid	100%
Invalid	100%
Tables	
None	
One	100%
More than one	100%

Duplicate Columns	
None	100%
Exists	100%